PENGEMBANGAN APLIKASI PEMBELAJARAN UNTUK MENDUKUNG MAHASISWA DALAM BELAJAR PEMROGRAMAN JAVA BERBASIS ANDROID (STUDI KASUS : FILKOM UB)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Fadhyl Farhan Alghifari
NIM: 155150207111146

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2020

PENGESAHAN

PENGEMBANGAN APLIKASI PEMBELAJARAN UNTUK MENDUKUNG MAHASISWA DALAM BELAJAR PEMROGRAMAN JAVA BERBASIS ANDROID (STUDI KASUS : FILKOM UB)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Fadhyl Farhan Alghifari
NIM: 155150207111146

Skripsi ini telah diuji dan dinyatakan lulus pada
15 Juli 2020
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I

Komang Candra Brata, S.Kom., M.T., M.Sc.
NIK: 201607 890711 1 001

Dosen Pembimbing II

Dr.Eng. Ahmad Afif Supianto, S.Si., M.Kom.
NIK: 201201 820623 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 30 Juni 2020

Fadhyl Farhan Alghifari

NIM: 155150207111146

ABSTRAK

Fadhyl Farhan Alghifari, Pengembangan Aplikasi Pembelajaran Untuk Mendukung Mahasiswa Dalam Belajar Pemrograman Java Berbasis Android (Studi Kasus : FILKOM UB)

Pembimbing: Komang Candra Brata, S.Kom., M.T. dan Dr. Eng. Ahmad Afif Supianto, S.Si., M.Kom.

Kemajuan bidang teknologi informasi di Indonesia mengakibatkan kebutuhan terhadap alumni perguruan tinggi berbasis IT tinggi. Namun, masih banyak lulusan yang dianggap mengecewakan. Salah satu faktor yang menyebabkannya adalah tidak disertakannya kurikulum pemrograman di tingkat sekolah SD, SMP, dan SMA serta terbatasnya akses pembelajaran pemrograman. Kemudian, penulis melakukan pengumpulan data dengan cara kuisioner kepada 58 Mahasiswa Fakultas Ilmu Komputer Universitas Brawijaya (FILKOM UB) angkatan 2018 dimana 75,9% responden berpendapat bahwa mempelajari pemrograman di perangkat *mobile* akan mempermudah mereka belajar Pemrograman karena akan mempermudah akses belajar. Lalu, sebesar 91,4% responden sebelum menjadi mahasiswa belum pernah mempelajari Bahasa Java, padahal Java ialah satu bahasa pemrograman yang sering dipakai. Oleh karena itu, aplikasi *M-Learning* untuk mempelajari Pemrograman Java dirasa perlu untuk mempermudah mahasiswa dalam mempelajari Pemrograman Java. Penelitian ini akan menggunakan metode *Mobile-D* dimana aplikasi *M-Learning* sifatnya fleksibel dan metode *Mobile-D* dipakai karena dapat menyesuaikan dengan kebutuhan pengguna yang selalu berubah.

Berdasarkan hasil penelitian yang telah dilakukan, penilaian pengguna terhadap aplikasi dengan nilai skor rata-rata SUS sebesar 84 pada mahasiswa dan 70 pada dosen, maka dapat disimpulkan bahwa Aplikasi Pembelajaran untuk Mendukung Mahasiswa dalam Belajar Pemrograman Java Berbasis Android mudah digunakan dan dapat diterima oleh pengguna.

Kata kunci: Java, android, *m-learning, mobile-d*

ABSTRACT

Fadhyl Farhan Alghifari, Development of Android Based Learning Applications to Support Students in Learning Java Programming (Case Study : FILKOM UB)

Supervisors: Komang Candra Brata, S.Kom., M.T. dan Dr. Eng. Ahmad Afif Supianto, S.Si., M.Kom.

The advancement of information technology in Indonesia have resulted IT Graduated Students are necessarily needed. However, there are still many graduates who are considered disappointing. One of the factors causing this problem is the exlusion of programming curricula at the elementary, junior, and senior high school level and the limited access to programming course. Then, the authors collected data by questionnaire to 58 students of the Faculty of Computer Science Universitas Brawijaya (FILKOM UB) in 2018 where 75.9% of respondents thought that learning programming on mobile devices would make it easier for them to learn programming because it would facilitate access to learning. Then, 91.4% of respondents never learned the Java Language before becoming a college student, even though Java is a programming language that is often used. Therefore, the M-Learning application to learn Java Programming is necessary to facilitate students in learning Java Programming. This research will use the Mobile-D method in which the M-Learning application is flexible and the Mobile-D method is used because it can adapt to the users need.

Based on the results of research conducted, user ratings of applications with an average SUS score of 84 for students and 70 for lecturers, it can be concluded that Java-Based Learning Applications to Support Students in Learning Java Programming are easy to use and can be accepted by users.

Keyword: Java, android, *m-learning, mobile-d*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Kemajuan bidang teknologi informasi atau *Information Technology* (IT) di Indonesia mengakibatkan kebutuhan terhadap alumni perguruan tinggi dengan jurusan IT tinggi. Walaupun alumni perguruan tinggi di jurusan IT tinggi, terdapat masalah yang dihadapi, salah satunya adalah banyak lulusan yang dianggap mengecewakan. Masalah yang terjadi diakibatkan salah satunya karena faktor tidak disertakannya kurikulum pemrograman di sekolah tingkat SD, SMP, dan SMA. Hal ini menyebabkan para mahasiswa IT yang minim pengetahuan terhadap pembelajaran pemrograman sebelumnya susah untuk beradaptasi karena mereka hanya mendapatkan pembelajaran pemrograman saat waktu kuliah saja dengan akses yang terbatas (Koran Sindo, 2018).

Dari masalah tersebut, penulis kemudian melakukan pengumpulan data dengan cara menyebarkan kuisioner kepada 58 Mahasiswa Fakultas Ilmu Komputer Universitas Brawijaya (FILKOM UB) angkatan 2018 untuk mengetahui masalah yang mereka hadapi dalam mempelajari materi pemrograman di mana mereka wajib untuk mempelajari Pemrograman Java. Sebanyak 91,4% responden belum mempelajari Pemrograman Java sebelum menjadi Mahasiswa FILKOM UB, dan hanya 5,2% responden yang menganggap bahwa mempelajari pemrograman Java itu mudah. Sebagian besar responden berpendapat bahwa materi Pemrograman Java yang mereka anggap sulit ialah materi perulangan *for*, perulangan *while*, dan perulangan *do-while*. Kemudian 67,2% responden menjawab bahwa mereka tidak pernah menggunakan *tools/software* untuk mempelajari Pemrograman Java selama ini. Terakhir, 75,9% responden berpendapat bahwa mempelajari pemrograman di perangkat *mobile* akan mempermudah mereka dalam belajar Pemrograman.

Pemrograman Java banyak dipelajari oleh masyarakat termasuk Mahasiswa FILKOM UB dikarenakan Java merupakan salah satu bahasa pemrograman yang paling banyak dipakai. Pada tahun 2016 berdasarkan *TIOBE Index*, Bahasa Pemrograman Java menduduki posisi pertama dan menduduki posisi kedua untuk jumlah repositori di Github dibawahnya *Javascript*. Bahasa Pemrograman Java sering digunakan karena seringkali dipakai untuk aplikasi *web*, robotika, dan aplikasi *Native* Android yang sering digunakan saat ini. (Fajar, 2017). Aplikasi *Native* adalah aplikasi yang dibangun untuk target platform tertentu dengan menggunakan bahasa pemrograman tertentu juga, contohnya Bahasa Java atau Kotlin yang digunakan untuk membangun aplikasi dengan Platform Android. Aplikasi *Native* memiliki *User Experience*, kualitas, keamanan yang lebih baik daripada aplikasi *Hybrid* serta memiliki akses penuh ke perangkat bergerak (Desantha, 2018).

Walaupun mempelajari pemrograman itu susah, namun mempelajarinya dengan perangkat *mobile* seperti *smartphone* akan menciptakan pengalaman yang baru dan menarik bagi para pelajar itu sendiri. Mereka juga berpendapat

bahwa mempelajari pemrograman di perangkat mobile dapat memudahkan para pelajar yang akan mempelajari pemrograman karena dapat diakses dengan mudah serta akan membuat mempelajari pemrograman semakin menarik (Tillmann, Moskal & Halleux, 2012).

Salah satu metode untuk belajar melalui *smartphone* ialah dengan metode *Mobile Learning* (*M-Learning*). *M-Learning* ialah suatu metode *E-Learning* yang dibatasi oleh fungsionalitas, ukuran layar, kecepatan prosesor, dan daya tahan baterai (Crompton, 2013). Menurut Traxler (2005), karakteristik *M-Learning* ialah spontan, bersifat pribadi, *portable*, situational, informal, ringan, *context-aware*, *connected* dan interaktif.

Selain menyebarkan kuisioner ke mahasiswa, penulis juga menyebarkan kuisioner kepada sebagian dosen Fakultas Ilmu Komputer Universitas Brawijaya (FILKOM UB) yang pernah atau sedang mengajarkan perkuliahan dasar-dasar Pemrograman Java. 75% responden berpendapat bahwa sebagian besar mahasiswa yang mereka ajarkan juga mengalami kesulitan saat mempelajari materi pemrograman Java, yakni materi perulangan *for*, perulangan *while*, dan perulangan *do while*. Selain itu, 75% responden berpendapat pula jika pembelajaran dengan metode *M-Learning* dapat mendukung mahasiswa dalam proses pembelajaran Pemrograman Java.

Sedangkan untuk pengembangan aplikasi *M-Learning* akan menggunakan metode *Mobile-D*. Metode *Mobile-D* adalah suatu metode yang dikembangkan dari beberapa *framework* seperti *Crystal*, *Rational Unified Process*, dan *Extreme Programming* (XP). Metode *Mobile-D* digunakan karena sifatnya yang fleksibel dan *agile*. Metode *Mobile-D* juga cocok digunakan karena jumlah pengembang yang kecil, berorientasi pada objek, dan waktu pengembangan yang cepat (Abrahamsson, Hanhineva, Hulkko, 2004).

Setelah aplikasi selesai dikembangkan, maka akan dilakukan pengujian *usability* pada pengguna. Tujuan dilakukannya pengujian *usability* adalah sebagai bahan pertimbangan apakah pengguna akan menggunakan perangkat lunak (Brooke, 2011). Pengujian *usability* akan menggunakan skala *System Usability Scale* (SUS). Skala SUS digunakan untuk mendapatkan *range acceptability* apakah aplikasi dapat diterima atau tidak oleh pengguna sebelum aplikasi dapat disebarkan kepada calon pengguna nantinya (Bangor, Kortum & Miller, 2009).

Penelitian ini bertujuan untuk mengembangkan aplikasi *M-Learning* Belajar Java berbasis *Android* membantu mahasiswa dalam kemudahan akses pembelajaran Pemrograman Java. Dengan aplikasi ini, penulis berharap para mahasiswa dapat mempelajari pemrograman Java hanya melalui *smartphone*.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, maka dirumuskanlah beberapa rumusan masalah, yaitu :

1. Bagaimana tingkat validitas aplikasi *BelajarJava*.
2. Apakah aplikasi *BelajarJava* mudah untuk digunakan pengguna.

1.3 Tujuan

Penelitian ini bertujuan untuk mengimplementasikan aplikasi BelajarJava untuk membantu mahasiswa belajar Bahasa Pemrograman Java dimanapun dan kapanpun.

1.4 Manfaat

Manfaat yang diharapkan dari penelitian ini adalah :

1. Membantu para mahasiswa untuk mempelajari Pemrograman Java melalui *smartphone* berbasis Android.
2. Sebagai sarana bagi pengajar di bidang pemrograman Java untuk menerapkan *M-Learning*.

1.5 Batasan Masalah

Dalam penelitian ini, peneliti memiliki batasan sebagai berikut :

1. Aplikasi yang akan dikembangkan hanya berisi konten materi perulangan *for*, perulangan *while*, dan perulangan *do-while*.
2. Penelitian ini tidak mengadopsi metodologi *Mobile-D* yang berhubungan dengan manajemen tim.
3. Iterasi yang dilakukan pada penelitian ini sebanyak dua kali.

1.6 Sistematika Pembahasan

Skripsi ini disusun dalam tujuh bab pembahasan sebagai pedoman berfikir secara sistematis. Sistematika skripsi ini sebagai berikut :

BAB I PENDAHULUAN

Bab ini menjelaskan gambaran umum pada penelitian yang berupa latar belakang dilakukannya penelitian, rumusan masalah, tujuan, manfaat, batasan masalah, dan sistematika pembahasan

BAB II LANDASAN KEPUSTAKAAN

Bab ini berisi dasar teori dan pustaka yang digunakan penulis dalam melakukan penelitian. Pada bab ini juga menjelaskan kajian pustaka terkait penelitian yang sebelumnya telah dilakukan dan terdapat hubungannya dengan penelitian yang akan dilakukan.

BAB III METODOLOGI PENELITIAN

Bab ini menjelaskan tahapan penelitian dan metode yang akan dilakukan dalam penelitian.

BAB IV REKAYASA KEBUTUHAN

Bab ini membahas tentang penggalian kebutuhan serta analisis terhadap kebutuhan terkait dengan aplikasi yang akan dikembangkan.

BAB V PERANCANGAN DAN IMPLEMENTASI

Bab ini membahas tentang analisis kebutuhan dan perancangan terkait dengan aplikasi pembelajaran untuk mendukung mahasiswa dalam belajar pemrograman Java berbasis Android. Selain itu, bab ini juga membahas tentang implementasi pada aplikasi pembelajaran untuk mendukung mahasiswa dalam belajar Pemrograman Java berbasis Android.

BAB VI PENGUJIAN

Bab ini membahas tentang hasil pengujian dari hasil implementasi aplikasi pembelajaran untuk mendukung mahasiswa dalam belajar pemrograman Java berbasis Android.

BAB VII PENUTUP

Bab ini membahas tentang kesimpulan yang didapatkan dari hasil penelitian yang telah dilakukan pada skripsi serta saran untuk pengembangan aplikasi pembelajaran untuk mendukung mahasiswa dalam belajar pemrograman Java lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Penelitian terkait pembelajaran Java berbasis perangkat *mobile* pernah dilakukan oleh T. Jordine, et. al (2015). Penelitian ini membahas metode pembelajaran tentang pemrograman Java berbasis perangkat *mobile* menggunakan media game. Sistem yang dikembangkan berupa game berbasis *mobile* untuk menyelesaikan masalah mahasiswa dimana mereka kesulitan dengan materi pembelajaran pemrograman berorientasi objek.

Penelitian terkait metode pengembangan untuk perangkat *mobile* pernah dilakukan oleh Abrahamsson, et. al (2004). Penelitian ini membahas sebuah metode *agile* untuk pengembangan aplikasi perangkat bergerak yang menantang karena tuntutan spesifik dan kendala teknis pengembangan perangkat bergerak. Metode pengembangan ini disebut *Mobile-D* dan telah mendapatkan sertifikat CMMI level 2.

Penelitian terkait pengembangan aplikasi *M-Learning* dilakukan oleh Brata, et. al (2018). Penelitian ini membahas pengembangan aplikasi *M-Learning* dengan media suara berbasis Android menggunakan *Firebase* sebagai database-nya. Hasil yang didapatkan dari aplikasi *M-Learning* ini mendapatkan skor *usability* sebesar 81 yang berarti performa dari aplikasi *M-Learning* yang dikembangkan sangat bagus.

Penelitian terkait pengembangan aplikasi *mobile* berbasis Android dilakukan oleh Firmansyah, et. al (2018). Penelitian ini membahas pengembangan aplikasi *mobile* dengan menggunakan metode *Mobile-D* sehingga diharapkan menghasilkan aplikasi yang dibutuhkan. Aplikasi yang dikembangkan adalah aplikasi komplain kebersihan pesawat dalam penerbangan komersial berbasis Android. Metode *Mobile-D* digunakan dalam penelitian ini karena bersifat *agile* dan fleksibel. Kesimpulan dari penelitian ini adalah metode perancangan *mobile-d* cocok digunakan untuk aplikasi yang dikembangkan karena dokumentasi yang dihasilkan tidak minim seperti metode *scrum*.

2.2 Dasar Teori

2.2.1 Aplikasi *Mobile*

Aplikasi *mobile* adalah sebuah perangkat lunak yang dibangun untuk perangkat bergerak seperti *Smartphone* dan *Tablet*. Para *developer* aplikasi *mobile* membangun aplikasi untuk beberapa sistem operasi yang kompatibel untuk perangkat bergerak yakni *iOS* milik Apple Inc. dan *Android* milik Google.

Aplikasi *mobile* mempunyai perbedaan dari aplikasi biasa, yakni aplikasi dapat menyesuaikan dengan lebar layar yang dimiliki oleh *Smartphone* pengguna, kapasitas memori yang cukup ringan, tampilan yang berbeda dengan aplikasi pada

biasanya, *button*, dan fungsi layar sentuh yang telah dirancang oleh *developer* (Viswanathan, 2017).

2.2.2 *Mobile Learning* (*M-Learning*)

Menurut Crompton (2013), *Mobile Learning* (M-Learning) mengadaptasi metode *E-Learning* yang memakai teknologi seperti *World Wide Web* (WWW). Namun, *M-Learning* dibatasi oleh fungsionalitas, ukuran layar, kecepatan prosesor, dan daya tahan baterai. Seiring dengan kemajuan teknologi, terdapat beberapa keunggulan yang ditawarkan oleh pembelajaran dengan metode *M-Learning*, yakni :

1. *Contingent Learning*, dimana pengajar merespon dan bereaksi terhadap lingkungan belajar dan menciptakan pengalaman yang berbeda.
2. *Situated Learning*, dimana pembelajaran dapat dilakukan di lokasi yang dapat dijadikan tempat untuk belajar.
3. *Authentic Learning*, dimana tugas-tugas yang berkaitan dapat langsung diberikan dengan cepat.
4. *Context-Aware Learning*, dimana pembelajaran dapat disesuaikan dengan riwayat pembelajaran dan lingkungan.
5. *Personalized Learning*, dimana pembelajaran dapat disesuaikan dengan kemampuan, preferensi, dan minat para pengajar.

Menurut Al-Fahad (2009), *M-Learning* dapat meningkatkan hasil pembelajaran para siswa. *M-Learning* bisa menjadi metode belajar yang efektif dikarenakan metode ini dapat memberi dukungan pembelajaran secara langsung, membawa pengalaman baru dalam pembelajaran, sifatnya yang fleksibel karena dapat dengan mudah untuk diakses di mana saja dan kapan saja, dan dapat meningkatkan komunikasi antara pengajar dan pelajar.

2.2.3 *Firebase*

Firebase adalah sebuah teknologi yang dikembangkan oleh Google untuk membuat aplikasi *web*, *mobile* atau *cross-platform* untuk mengontrol database dan melakukan sinkronisasi kepada pengguna tanpa menggunakan *server-side programming*. Jika aplikasi yang memakai *Firebase* dirancang untuk *platform* Android, maka *Operating System* (OS) harus berjalan pada perangkat yang menggunakan OS Android 4.0 atau diatasnya (Kumar, Akhi, Gunti & Reddy, 2016).

2.2.3.1 *Firebase Authentication*

Salah satu fitur *Firebase* adalah *Firebase Authentication* yang digunakan untuk menyimpan data pengguna dan disimpan di *cloud*. Fitur ini menyediakan layanan backend, SDK dan *library* UI yang siap digunakan untuk autentikasi pengguna ke aplikasi. Aplikasi yang menggunakan fitur ini dapat memanfaatkan fitur autentikasi dengan *password*, nomor telpon, atau autentikasi dengan menggunakan sosial media seperti Google, Facebook, Twitter, ataupun GitHub (Google, 2018).

2.2.3.2 *Firebase Realtime Database*

Firebase Realtime Database adalah *database* yang disimpan di *cloud* dan disimpan sebagai tipe data JSON. *Database* disinkronisasi secara langsung kepada pengguna yang terhubung sehingga pengguna secara otomatis dapat menerima *update* data terbaru. Walaupun pengguna dalam keadaan tidak terkoneksi ke internet, aplikasi yang memakai *Firebase* tetap responsif karena SDK yang digunakan *Firebase Realtime Database* menyimpan data ke *disk*. Saat pengguna terkoneksi ke internet, aplikasi melakukan sinkronisasi dengan status server yang terbaru dan memperbarui perubahan yang terlewat (Google, 2018).

2.2.3.3 *Firebase Cloud Storage*

Firebase Cloud Storage digunakan oleh para *developer* aplikasi baik aplikasi berbasis *mobile* ataupun aplikasi berbasis *web* untuk menyimpan data konten yang dibuat atau disimpan oleh pengguna, seperti foto dan video. *Firebase* SDK untuk penyimpanan *cloud* menambahkan keamanan Google ke *file* unggahan dan unduhan untuk aplikasi yang menggunakan *Firebase*, terlepas dari kualitas jaringan.

Pengembang menggunakan *Firebase* SDK untuk penyimpanan *cloud* untuk mengunggah dan mengunduh *file* langsung dari klien. Klien dapat mencoba lagi operasi tepat di tempat yang ditinggalkannya meski koneksi yang dipakai pengguna buruk. *Firebase* SDK untuk penyimpanan *cloud* terintegrasi dengan *Firebase Authentication* untuk mengidentifikasi pengguna, dan pengembang dapat mengontrol akses pada *file* individual atau grup *file*, sehingga pengembang dapat membuat *file* sebagai publik atau pribadi (Google, 2018).

2.2.4 Metode *Mobile-D*

Menurut Abrahamsson (2005) yang dikutip oleh Spataru (2010), sebuah metodologi untuk pengembangan aplikasi perangkat bergerak yang sukses adalah yang memprioritaskan hal sebagai berikut :

1. Ketangkasan (*Agility*).
2. Permintaan pasar.
3. Spesifikasi awal arsitektur fisik.
4. Dukungan umpan balik *end-user*.
5. Dukungan produk perangkat lunak.
6. Dukungan faktor *reusability*.
7. Pengembangan berbasis arsitektur yang ada.
8. Ulasan dan sesi belajar.

Pekerjaan ini berfokus pada peningkatan pada metode pengembangan aplikasi perangkat bergerak secara umum, yang menyeimbangkan antara konsumen, kebutuhan, harapan *end-user*, dan kendala vendor *platform*. Metode *Mobile-D* menggunakan daftar tersebut sebagai panduan, dan menyelidiki kemungkinan perbaikan di sub wilayah tertentu. Dengan memeriksa kategori aplikasi perangkat bergerak yang sukses, dapat diambil langkah-langkah spesifik untuk meningkatkan kualitas dan meminimalkan risiko aplikasi yang sedang

dikembangkan. Prosedur penyelarasan dapat dimasukkan sebagai tugas dalam tahap *Mobile-D* tertentu. Prinsip-prinsip metodologi pengembangan yang lain juga dapat diintegrasikan ke dalam metode *Mobile-D* ketika skenario tersebut akan mendukung penggunanya.

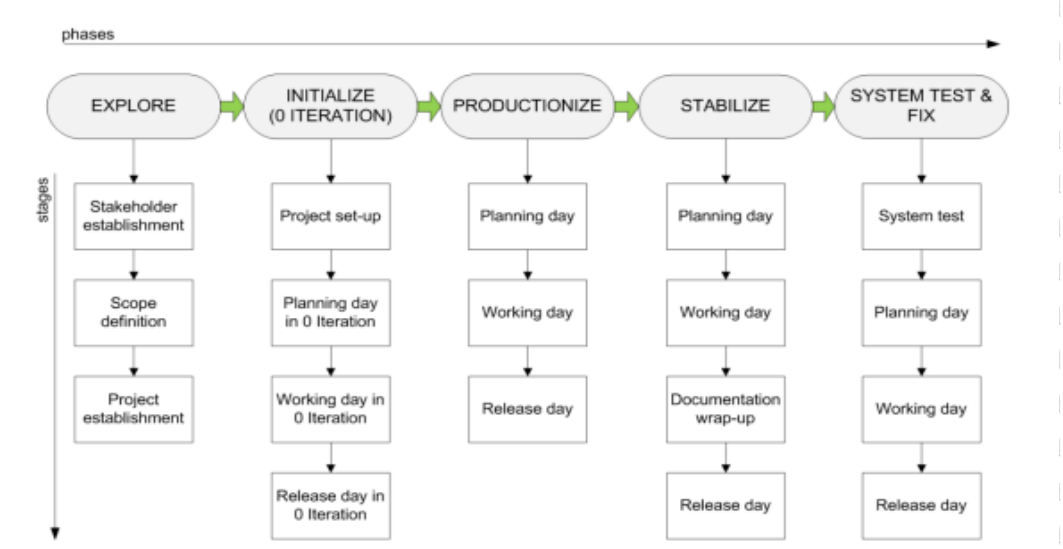
Metode *Mobile-D* adalah suatu metode yang dikembangkan dari beberapa *framework* metode *agile*, yaitu *Rationale Unified Process*, *Crystal* dan *Extreme Programming* (Flora, K.H. & Chande, S.V., 2013).

Beberapa fase metode *Mobile-D* terdapat pada Gambar 2.2.

Gambar 2.1 Fase *Mobile-D*

(Sumber : Flora, K.H. & Chande, S.V., 2013)

Menurut Sapataru (2010), metode *Mobile-D* yang diterapkan dalam beberapa proyek pembangunan, seperti deteksi kesalahan lebih awal dan perbaikan secara teknis, tingkat kecacatan produk akhir rendah, dan kemajuan terus-menerus dalam pembangunan. Masing-masing fase pada metode *Mobile-D* memiliki sejumlah tahapan yang digambarkan dalam Gambar 2.3.



Gambar 2.2 Fase dan Tahap *Mobile-D*

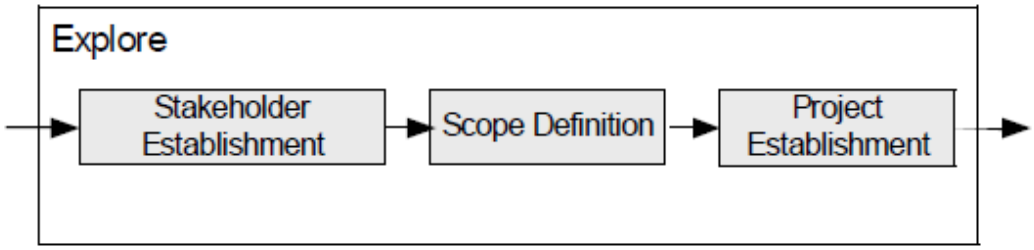
(Sumber : Spataru, 2010)

2.2.4.1 *Explore*

Fase *explore* adalah fase untuk merencanakan proyek yang baru akan dikembangkan. Fase ini digunakan sebagai dasar awal untuk membangun sebuah sistem. Contohnya arsitektur sistem dan pemilihan lingkungan perangkat lunak. Fase *explore* dapat dilakukan tanpa terikat waktu dalam fase *Mobile-D*. Fase *explore* ialah fase yang penting untuk menetapkan landasan bagi implementasi

perangkat lunak, seperti arsitektur perangkat lunak, proses pengembangan perangkat lunak, dan pemilihan lingkungan perangkat lunak. (Salo & Hulkko, 2004).

Pada fase *explore*, dibagi menjadi tiga tahap yang terdapat pada Gambar 2.4.



Gambar 2.3 Fase *Explore*

(Sumber : http://virtual.vtt.fi/virtual/agile/mobiled.html)

1. *Stakeholder Establishment*

Tahap ini bertujuan untuk menentukan *stakeholder* atau aktor pada aplikasi. Kemudian, semua aktor dan pemilik produk yang terlibat dalam pengembangan aplikasi didefinisikan perannya. Penentuan *stakeholder* mencakup tugas mengidentifikasi, melakukan, dan membentuk berbagai *stakeholder* yang diperlukan dalam proses pengembangan perangkat lunak. Penentuan peran *stakeholder* pada tahap ini yakni sebagai berikut :

a. Kelompok pengarah

Kelompok pengarah pada fase *explore* memiliki peran untuk mengelola proyek, membuat keputusan, mengamati dan mengarahkan proyek.

b. Tim proyek

Tim proyek terdiri dari beberapa kelompok dengan peran yang berbeda seperti pemimpin proyek, *programmer*, pelacak metrik, dan penguji.

c. Kelompok pengguna

Kelompok pengguna memiliki peran untuk identifikasi, pengujian, dan menerima rilis produk.

d. Representasi kelompok pengguna

Representasi dari kelompok pengguna memiliki peran terkait dengan persyaratan identifikasi, pengujian, dan menerima rilis produk. Representasi kelompok pengguna juga diperlukan dalam kelompok pengarah.

e. Kelompok pendukung

Kelompok pendukung memiliki peran yang dibutuhkan sepanjang proyek pengembangan, seperti spesialis untuk membantu menyesuaikan dan meningkatkan proses pengembangan proyek

perangkat lunak, pelatihan personel, atau ahli dari berbagai aspek teknis untuk membantu pengembangan perangkat lunak.

f. Kelompok eksplorasi

Kelompok eksplorasi untuk menangani inisiasi proyek sebelum keberadaan tim proyek.

Pada tahap *stakeholder establishment* memiliki dua *task*, yakni *customer establishment* dan *stakeholder group establishment*.

a. *Customer establishment*

Task ini membentuk kelompok kepentingan pelaku pengembangan perangkat lunak. *Task customer establishment* bertujuan untuk identifikasi pengguna yang berpartisipasi dalam pengembangan perangkat lunak, mendapatkan komitmen dari pengguna yang diidentifikasi, dan menentukan mode (*offsite/onsite*), tugas, peran dan tanggung jawab untuk kelompok pengguna.

b. *Stakeholder group establishment*

Task ini bertujuan untuk menentukan kelompok *stakeholder* yang akan berpartisipasi dalam proyek yang akan diimplementasi.

Penulis mengadopsi tahap *customer establishment* yang dijelaskan pada Sub bab 4.2.1.

2. *Scope Definition*

Tahap ini bertujuan untuk menjelaskan tujuan atau ruang lingkup dari aplikasi yang akan dikembangkan. Pengumpulan kebutuhan dilakukan yang akan dianalisis di fase selanjutnya.

Terdapat dua *task* yang terdapat pada tahap *scope definition*, yaitu :

a. *Initial project planning*

Pada *task initial project planning*, dilakukan pembuatan *timeline* pada proyek dan pendefinisian ritme iterasi. Tujuan dibuat *timeline* yakni untuk menyiapkan waktu yang dibutuhkan untuk pengembangan aplikasi dan alokasi waktu yang dibutuhkan tiap fase agar pengembangan aplikasi dapat selesai sesuai alokasi waktu yang direncanakan.

b. *Initial requirements collection*

Task initial requirements collection adalah *task* dimana kelayakan untuk perangkat lunak diatur sedemikian rupa. Persyaratan harus mencakup persyaratan fungsional seperti *platform hardware* dan *software* dan non-fungsional seperti masalah peliharaan pada sistem. Kebutuhan dapat diperbaiki pada saat ini, namun terkadang terdapat kebutuhan atau persyaratan baru yang muncul pada proyek secara berulang. Dalam keadaan kebutuhan atau persyatan baru yang akan muncul pada tiap fase yang akan datang, beberapa persyaratan atau kebutuhan awal tetap harus

ditentukan untuk mendefinisikan fungsi utama dari produk yang akan menjadi dasar perangkat lunak dari proses metode *Mobile-D*.

Penulis mengadopsi tahap *initial requirements collection* yang dijelaskan pada Bab 4, dimana penulis menggali kebutuhan dan menuliskannya dalam spesifikasi kebutuhan. Kemudian, kebutuhan tersebut dimodelkan dalam bentuk *use case*.

3. *Project Establishment*

Tahap ini dilakukan untuk menyiapkan kebutuhan untuk mengembangkan sistem, seperti sumber daya manusia, dan alat pendukung untuk mengembangkan aplikasi.

Pada tahap *project establishment* terdapat empat *task*, yaitu :

a. *Environment selection*

Pada *task* ini proyek direncanakan secara teknis mengenai lingkungan, misalnya target perangkat, alat pengembangan, *platform* perangkat keras produk serta SDM dan lingkungan kerja.

b. *Personnel allocation*

Pada *task* ini tim pengembang perangkat lunak serta tim pendukung didefinisikan dan dialokasikan untuk proyek.

c. *Architecture line definition*

Task ini dilakukan untuk membantu pengembang dalam merencanakan arsitektur perangkat lunak yang menggunakan metode *Mobile-D*. Masalah yang terkait dengan arsitektur dieksplorasi sebelum produksi perangkat lunak dilakukan, seperti konteks sistem, *platform* perangkat lunak, kualitas produk internal pada sistem, integrasi perancangan sistem, dokumentasi arsitektur perangkat lunak serta rancangan dokumentasi akhir, serta keterampilan arsitektur yang dibutuhkan.

d. *Process establishment*

Task ini meliputi identifikasi dan pendefinisian pelatihan yang diperlukan diantara tim proyek mengenai proses serta masalah teknis, proses dasar yang sesuai dengan proyek, dan pemantauan proyek (termasuk masalah jaminan kualitas, proyek dokumentasi, praktik, metrik, dan alat untuk memantau).

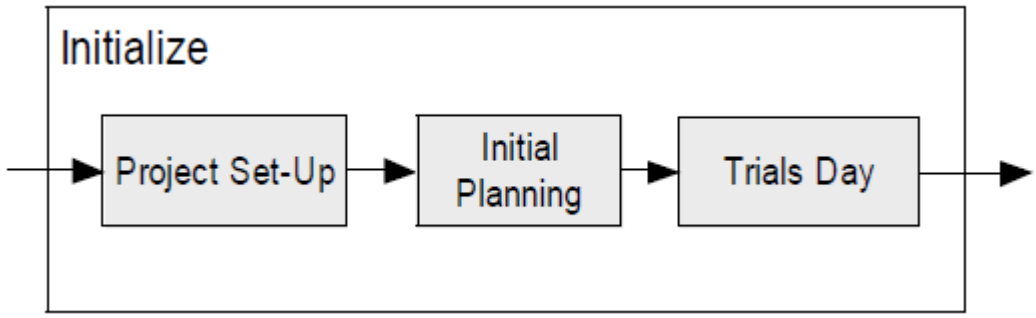
Penulis mengadopsi tahap *environment selection* yang dijelaskan pada bab 4 bagian pemilihan lingkungan pengembangan, serta tahap *architecture line definition* yang dijelaskan pada bab 4 bagian deskripsi perangkat lunak.

2.2.4.2 *Initialize*

Fase *Initialize* adalah fase untuk menyiapkan dan memverifikasi semua masalah terkait pengembangan. Fase *initialize* juga dikenal dengan sebutan iterasi 0 untuk menentukan pola pada pengembangan. Tujuan dari fase *initialize* adalah

untuk menambahkan kemungkinan kesuksesan dalam pembangunan proyek dengan memverifikasi semua masalah terkait pengembangan sehingga pengembang siap untuk mengimplementasikan kebutuhan yang telah didefinisikan oleh *stakeholder* (Ihme, 2005).

Pada fase *Initialize*, dibagi menjadi tiga tahap yang terdapat pada Gambar 2.5.



Gambar 2.4 Fase *Initialize*

(Sumber : http://virtual.vtt.fi/virtual/agile/mobiled.html)

1. *Project Set-Up*

Tahap ini dilakukan persiapan untuk mengembangkan perangkat lunak seperti OS, *hardware*, dan *software* pendukung yang akan digunakan untuk mengembangkan aplikasi. Tahap ini juga dijelaskan bagaimana cara antar anggota tim berkomunikasi dengan anggota tim pengembang lainnya.

2. *Initial Planning*

Tahap *initial planning* adalah tahap uji coba yang mengatur pengembangan teknis lingkungan dan untuk memastikan semua pengembang yang bergabung dalam proyek telah siap untuk mengimplementasi sistem yang akan dikembangkan. Tahap ini bertujuan untuk mendapatkan pemahaman yang baik terhadap aplikasi yang akan dikembangkan, menyiapkan dan mencari rencana proyek selanjutnya. Tahap ini juga bertujuan untuk mengimplementasikan beberapa fungsionalitas inti dari sistem tanpa menghasilkan kode program. Tahap ini membentuk pra-fase untuk waktu aktual pengembangan produk.

Pada tahap *initial planning* terdapat dua *task*, yaitu :

a. *Architecture line planning*

Task ini dilakukan untuk menyiapkan dasar yang akan dijadian acuan dalam mengimplementasi sistem. *Task* ini juga dilakukan sebagai langkah awal persiapan untuk mereka yang bergabung dalam tim proyek pengembangan perangkat lunak. Persiapan dilakukan dengan menyiapkan arsitektur perangkat lunak agar sesuai dengan kebutuhan yang telah didefinisikan oleh pengguna.

b. *Initial requirement analysis*

Tujuan dari *task ini* adalah menentukan prioritas kebutuhan secara teliti dari hasil analisis kebutuhan untuk menentukan komponen kebutuhan yang paling penting dan akan dijadikan dasar dalam mengimplementasi perangkat lunak. Kerangka arsitektural harus ditemukan paling lambat pada akhir iterasi. Hasil dari *task* ini merupakan sebuah *product backlog*. *Product Backlog* ialah daftar kebutuhan yang diinginkan stakeholder dalam produknya yang akan dikembangkan beserta prioritas kebutuhan yang akan diimplementasikan terlebih dahulu. Contoh *product backlog* ditunjukkan pada Gambar 2.6.

ToDo List

ID	Story	Estimation	Priority
7	As an unauthorized User I want to create a new account	3	1
1	As an unauthorized User I want to login	1	2
10	As an authorized User I want to logout	1	3
9	Create script to purge database	1	4
2	As an authorized User I want to see the list of items so that I can select one	2	5
4	As an authorized User I want to add a new item so that it appears in the list	5	6
3	As an authorized User I want to delete the selected item	2	7
5	As an authorized User I want to edit the selected item	5	8
6	As an authorized User I want to set a reminder for a selected item so that I am reminded when item is due	8	9
8	As an administrator I want to see the list of accounts on login	2	10
Total		**30**	

Gambar 2.5 Contoh *Product Backlog*

(Sumber : https://www.scrum-institute.org/The_Scrum_Product_Backlog.php)

Penulis mengadopsi tahap *architecture line planning* yang dijelaskan pada bab 5 bagian perancangan arsitektur sistem dan tahap *initial requirement analysis* yang dijelaskan pada bab 4 bagian elisitasi kebutuhan yang menghasilkan tabel 4.4 dan tabel 4.8.

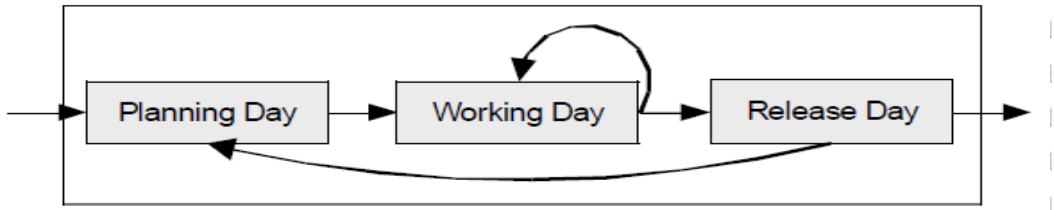
3. *Trial Day*

Tahap ini dilakukan untuk menguji coba lingkungan pengembangan serta memastikan apakah semua rencana dan kebutuhan yang telah dijelaskan telah siap untuk diimplementasikan dalam bentuk produk.

2.2.4.3 *Productionize*

Fase *Productionize* adalah fase dimana kebutuhan yang telah didefinisikan setelah iterasi 0 diubah ke dalam bentuk produk. Peran pengguna harus ditekankan karena *feedback* yang didapatkan dari pengguna bermanfaat untuk mencapai hasil produk yang memuaskan. Tujuan utama dari fase *productionize* ialah mengimplementasikan fungsionalitas inti. (Koskela & Kyllönen, 2004).

Pada fase *Productionize*, dibagi menjadi tiga tahap yang terdapat pada Gambar 2.7.



Gambar 2.6 Fase *Productionize*

(Sumber : http://virtual.vtt.fi/virtual/agile/mobiled.html)

1. *Planning Day*

Tahap ini digunakan untuk memilih dan merencanakan konten pekerjaan untuk iterasi. Pengguna yang berpartisipasi aktif untuk kegiatan perencanaan, memastikan kebutuhan dapat memberikan nilai bisnis paling menguntungkan dan kebutuhan yang telah didefinisikan telah sesuai.

Pada tahap *planning day* fase *productionize* terdapat 5 *task*, yaitu :

a. *Post-iteration workshop*

Tujuan dari *task* ini adalah untuk meningkatkan proses pengembangan perangkat lunak agar lebih sesuai dengan kebutuhan tim proyek perangkat lunak. Ini termasuk identifikasi kelebihan dan kekurangan dalam proses serta tindakan perbaikan untuk iterasi berikutnya.

b. *Requirement analysis*

Tujuan dari *task* ini adalah untuk menentukan prioritas kebutuhan yang akan diimplementasikan terlebih dahulu. Selama *task* ini kebutuhan akan dianalisis mana yang memiliki nilai bisnis yang paling besar, dan menjelaskan secara spesifik tentang kebutuhan tersebut agar dapat dengan mudah dimengerti.

c. *Iteration planning*

Tujuan dari *task* ini adalah untuk menghasilkan jadwal dan konten untuk iterasi yang dieksekusi. Isi dari konten didefinisikan dalam tugas yang merupakan perintah kerja untuk tim proyek.

d. *Acceptance test generation*

Tujuan dari *task* ini adalah melakukan verifikasi terhadap kebutuhan pengguna agar hasil dari produk nanti telah sesuai dengan hasil perancangan dan kebutuhan yang didefinisikan. *Acceptance test* dihasilkan selama *planning day*, dan dilaksanakan dengan pengguna untuk menemukan dan membuat dokumentasi tentang masalah yang ditemukan seperti *error* dan *bug* pada perangkat lunak.

e. *Acceptance test review*

Tujuan dari *task* ini adalah untuk melakukan *review* terhadap *acceptance test* yang dilakukan untuk keseluruhan tim dan mengizinkan anggota tim untuk mengomentari *acceptance test* untuk meningkatkan kualitas perangkat lunak.

Penulis mengadopsi tahap *requirement analysis* yang dijelaskan pada bagian bab 4 dan bab 5. Pada tahap *requirement analysis*, penulis melakukan *update* terhadap kebutuhan perangkat lunak berdasarkan fase *initialize* atau hasil dari iterasi 0 yang telah dilakukan.

2. Working Day

Tahap ini bertujuan untuk mengimplementasikan fungsionalitas yang akan diterapkan di produk yang dikembangkan dari kebutuhan yang telah didefinisikan. Pengembang berfokus pada fungsionalitas dengan prioritas paling tinggi yang telah didefinisikan oleh pengguna.

Pada tahap *working day* terdapat enam *task*, yaitu :

a. *Wrap-up*

Task ini merupakan sesi interaktif untuk mengkomunikasikan kemajuan dan penemuan masalah selama pengembangan dalam tim. *Task wrap-up* biasanya dilakukan sebagai aktivitas pertama atau terakhir pada hari kerja.

b. *Test driven development* (TDD)

Dalam TDD, tes unit ditulis terlebih dahulu sebelum mengimplementasikan dalam kode program. Tujuan TDD adalah untuk memberikan acuan kepada pengembang terhadap hasil kode mereka bekerja dan dapat dijadikan desain untuk memandu pengembang dalam mengimplementasikan kode program nantinya.

c. *Pair programming*

Pair programming adalah gaya pemrograman dimana dua *programmer* mengembangkan kode secara bersama. Tujuan dari *pair programming* adalah untuk meningkatkan komunikasi, menyebarkan pengetahuan dalam tim dan memastikan kualitas kode program.

d. *Continuous integration*

Tujuan dari *continuous integration* adalah untuk terus mengintegrasikan kode baru dengan kode yang ada dalam *repository*.

e. *Refactoring*

Refactoring ialah proses meningkatkan internal struktur perangkat lunak tanpa mengubah perilaku eksternal. Dengan perbaikan kecil pada kode program, *refactoring* menjamin bahwa perangkat lunak dapat dimodifikasi dan dibaca.

f. *Inform customer*

Tujuan dari *task* ini adalah memberitahu pengguna terhadap kemajuan proyek, dan untuk meminta *feedback* dari pengguna tentang fitur yang telah diimplementasikan dan untuk memandu pengembangan perangkat lunak.

Penulis mengadopsi tahap *wrap-up* yang merupakan pembaruan penulisan kebutuhan sesuai hasil iterasi 0, dan tahap TDD yang dijelaskan pada Sub bab 6.1.

3. Release Day

Pada tahap ini, fungsionalitas yang diimplementasikan pada tahap sebelumnya diverifikasi dan divalidasi. Produk yang diimplementasikan dievaluasi oleh *stakeholder* utama pada proyek.

Pada tahap *release day* terdapat empat *task*, yaitu :

a. *System integration*

Task ini dilakukan dalam proyek multi-tim, yang bertujuan untuk mengintegrasikan sub sistem yang dihasilkan dalam tim yang terpisah menjadi satu produk.

b. *Pre-release testing*

Pre-release testing dilakukan untuk merilis hasil aplikasi awal yang telah diproduksi oleh tim pengembang kepada pengguna secara terbatas dan untuk menginformasikan *stakeholder* bahwa aplikasi telah siap untuk pengujian dan rilis hari penerimaan.

c. *Acceptance testing*

Acceptance Testing dilakukan untuk memastikan perangkat lunak yang dikembangkan telah sesuai dengan kebutuhan yang telah didefinisikan oleh pengguna. Masalah yang ditemukan seperti *error* dan *bug* didokumentasikan.

d. *Release ceremonies*

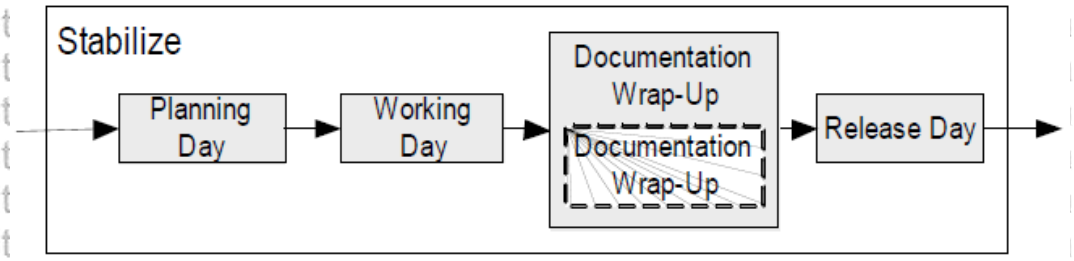
Task ini merupakan *task* terakhir dalam fase *productionize* sebelum merilis perangkat lunak. Dalam *task* ini terdiri dari dua kegiatan penting, yaitu pelepasan audit dan pembuatan *baseline*. Kegiatan ini dilakukan mengkonfirmasi bahwa semuanya telah dilakukan dengan benar dalam iterasi saat ini dan dijadikan dasar untuk pengembangan lebih lanjut.

Penulis mengadopsi tahap *acceptance testing* yang dijelaskan pada bab 6. Pada *task* ini dilakukan iterasi 1 untuk mendapatkan *feedback* pengguna terhadap implementasi awal yang telah dilakukan pengembang. Selanjutnya hasil dari iterasi 1 akan disempurnakan kembali untuk menjamin kualitas dari perangkat lunak yang dikembangkan.

2.2.4.4 *Stabilize*

Tujuan dari fase *Stabilize* adalah menyempurnakan perangkat lunak yang sebelumnya telah memasuki rilis awal dan telah mendapatkan *feedback* dari pengguna untuk memastikan kualitas produk yang dikembangkan (Ihme, 2004).

Pada fase *Stabilize*, dibagi menjadi empat tahap yang terdapat pada Gambar 2.8.



Gambar 2.7 Fase *Stabilize*
(Sumber : http://virtual.vtt.fi/virtual/agile/mobiled.html)

1. *Planning Day*

Tahap ini dilakukan untuk mendefinisikan konten kembali dan mengimplementasikan fitur tambahan ataupun fitur yang diperlukan untuk menambah kualitas dari produk yang dikembangkan. Pengguna berpartisipasi aktif dalam menjamin kebutuhan yang telah diimplementasikan merupakan kebutuyan yang paling menguntungkan dan memastikan kebutuhan telah sesuai dan dapat dimengerti.

Task yang terdapat pada fase *stabilize* tahap *planning day* sama seperti *task planning day* pada fase *productionize*. Penulis juga melakukan adopsi *task* yang sama seperti pada fase *productionize*, yaitu *requirement analysis* yang dijelaskan pada Bab 4 dan Bab 5 pada kebutuhan sistem berdasarkan hasil iterasi 1 yang telah dilakukan.

2. *Working Day*

Implementasi kebutuhan kemudian diselesaikan pada tahap *working* day dan memastikan produk yang dikembangkan terjamin kualitasnya.

Task yang terdapat pada fase *stabilize* tahap *working day* sama seperti *task working day* pada fase *productionize*. Penulis menyempurnakan implementasi perangkat lunak berdasarkan kebutuhan yang telah didefinisikan hasil dari iterasi 1.

3. *Documentation Wrap-Up*

Tahap ini bertujuan untuk menyelesaikan dokumen arsitektur perangkat lunak, desain, dan dokumen yang terkait dengan *User Interface* (UI). Dokumentasi dibutuhkan pasca pengembangan untuk ditujukan kepada orang lain dan dokumen harus lebih lengkap, konsisten, dan bersifat resmi.

Mereka harus membantu memahami apa yang dikembangkan, mengapa perangkat lunak dibangun seperti itu, dan bagaimana cara mengoperasikannya.

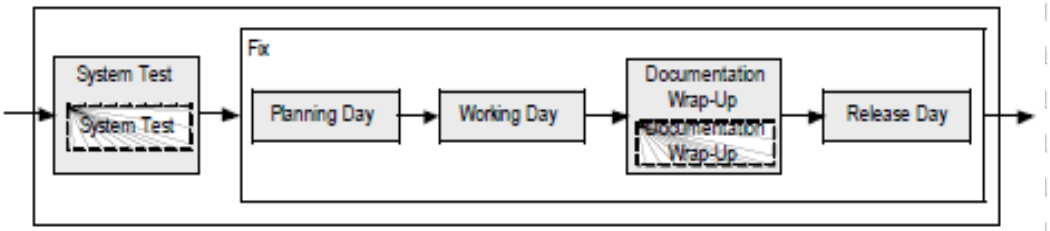
4. *Release Day*

Pada tahap ini, pengembang melakukan validasi keseluruhan implementasi untuk menjamin kualitas perangkat lunak dan melakukan *update* terhadap dokumentasi terkait perangkat lunak yang dikembangkan.

2.2.4.5 *System Test & Fix*

Pada fase *system test & fix* dilakukan dua tahap, yaitu *system test* dan *fix*. Tujuan dari fase ini adalah untuk menguji perangkat lunak. Jika terjadi *error*/*bug* pada perangkat lunak, *error*/*bug* yang dijelaskan pada dokumen diperbaiki pada fase *fix* dan dilakukan berulang sampai tidak terdapat *error*/*bug* pada perangkat lunak. Tidak ada fungsi baru yang diterapkan selama fase *fix* dilakukan (Jäälinoja, Hanhineva & Koskela, 2004).

Pada fase *Fix*, dibagi menjadi empat tahap yang terdapat pada Gambar 2.9.



Gambar 2.8 Fase *System Test & Fix*

(Sumber : http://virtual.vtt.fi/virtual/agile/mobiled.html)

1. *Planning Day*

Tujuan dari tahap ini dilakukan untuk mendefinisikan kebutuhan kembali dan mengimplementasikan fitur yang terdapat *error*/*bug* untuk menambah kualitas dari produk yang dikembangkan.

2. *Working Day*

Tujuan dari tahap ini adalah untuk memperbaiki *error*/*bug* pada perangkat lunak dan menyelesaikan implementasi perangkat lunak.

3. *Documentation Wrap-Up*

Tahap ini melakukan *update* dokumen arsitektur perangkat lunak, desain perangkat lunak, dan dokumen yang terkait dengan *User Interface* (UI) setelah menemukan *error* ataupun *bug* pada perangkat lunak yang telah diperbaiki pada perangkat lunak untuk tujuan iterasi perbaikan.

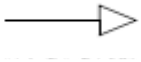
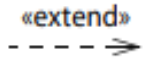
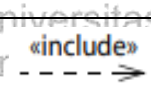
4. *Release Day*

Pada tahap ini, pengembang melakukan validasi dan memastikan keseluruhan implementasi untuk menjamin kualitas perangkat lunak. Setelah dilakukan validasi terhadap perangkat lunak, maka perangkat lunak siap untuk dirilis kepada calon pengguna.

2.2.5 *Use Case Diagram*

Use case menggambarkan perilaku dari sebuah sistem, subsistem, *class* atau komponen yang terdapat pada perangkat lunak. Sebuah *use case* menggambarkan fungsi sistem dan aktor yang terlibat didalamnya (Rumbaugh, 2005). Simbol yang terdapat pada *use case diagram* dijelaskan pada Tabel 2.1.

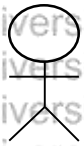
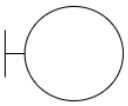
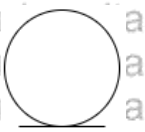
Tabel 2.1 Simbol pada *Use Case Diagram*

Simbol	Nama	Deskripsi
	Use Case	Skenario dari fungsi yang terdapat pada sistem. Terjadi interaksi antara sistem dan aktor
	Aktor	Peran orang, sistem lain, atau alat yang dapat berinteraksi dengan *use case*
	Asosiasi	Penghubung antar objek
	Generalisasi	Penghubung *use case* induk dengan *use case* yang mewarisi *use case* induk dan terdapat penambahan fitur didalamnya
	Extend	Penambahan perilaku *use case* tambahan dari *use case* dasar dan *use case* tersebut dapat berdiri sendiri
	Include	Menspesifikasikan *use case* sumber secara eksplisit

2.2.6 *Sequence Diagram*

Sequence diagram merupakan diagram kolaborasi dinamis antar objek dalam sebuah grafik dua dimensi. *Sequence diagram* digunakan untuk menggambarkan alur interaksi yang terjadi pada sebuah sistem dalam bentuk pesan antara objek serta interaksi antar objek (Rumbaugh, 2005). Simbol yang terdapat pada *sequence diagram* dijelaskan pada Tabel 2.2.

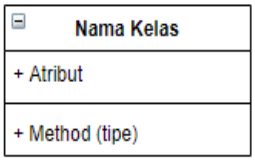
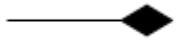
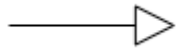
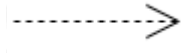
Tabel 2.2 Simbol pada *Sequence Diagram*

Simbol	Nama	Deskripsi
	Aktor	Peran orang, sistem lain, atau alat yang dapat berinteraksi dengan *use case*
	Boundary	Menggambarkan antarmuka yang merupakan interaksi antara aktor dengan sistem
	Entity	Klas dari domain sistem
	Controller	Penghubung antara boundary dengan entity
	Activation	Simbol yang menunjukkan bahwa suatu objek sedang berinteraksi dengan objek lainnya
	Lifeline	Menyatakan kehidupan sebuah objek
	Pesan	Menggambarkan pengiriman pesan antar objek
	Return	Menyatakan nilai kembalian sebuah objek

2.2.7 *Class Diagram*

Class diagram merupakan diagram yang mendeskripsikan struktur sistem dalam segi kelas serta hubungan antar kelas. Pada *class diagram* terdapat atribut dan operasi beserta *method* yang terdapat pada suatu objek atau kelas (Rumbaugh, 2005). Simbol-simbol yang terdapat pada *class diagram* dijelaskan pada Tabel 2.3.

Tabel 2.3 Simbol pada *Class Diagram*

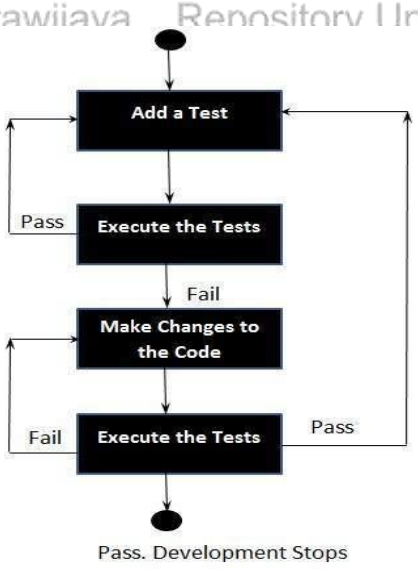
Simbol	Nama	Deskripsi
	Kelas	Objek pada sistem yang memiliki atribut serta *method*
	Atribut	Karakteristik dari sebuah kelas
	Method	Operasi dari sebuah kelas
	Asosiasi	Penghubung antar kelas
	Komposisi	Penghubung antar kelas dimana kelas tersebut bergantung satu sama lain
	Agregasi	Penghubung antar kelas dimana suatu kelas merupakan bagian dari kelas lain namun tidak bergantung satu sama lain
	Generalisasi	Penghubung kelas induk dengan kelas yang mewarisi perilaku kelas induk
	Dependency	Penghubung suatu kelas dimana *method* suatu kelas membutuhkan objek dari kelas lain

2.2.8 Pengujian Validasi

Pengujian validasi mengacu pada proses evaluasi perangkat lunak. Pengujian validasi dilakukan untuk menjamin bahwa perangkat lunak yang telah dikembangkan telah bebas dari *error* dan *bug* serta memastikan bahwa kebutuhan telah sesuai dengan sistem yang dikembangkan. Jika terdapat *error* atau *bug* dalam sistem, maka fungsi yang diimplementasikan dinyatakan gagal. Perangkat lunak juga dapat divalidasi kebutuhannya dengan memanfaatkan prototipe (Collofello, 1988).

2.2.9 *Test Driven Development* (TDD)

Test Driven Development (TDD) menurut Sommerville (2011) adalah pengembangan program dimana pengembang melakukan pengujian terhadap algoritma dan pengembangan kodenya. Pada *task* TDD, kode dikembangkan secara bertahap, dan dilakukan tes secara bersamaan untuk kode tersebut. Gambar 2.10 menunjukkan prosedur dalam melakukan pengujian TDD.



Gambar 2.9 Contoh Pengujian Unit pada TDD

(Sumber : tutorialspoint.com)

Pengujian TDD pada perangkat Android sendiri dilakukan pengujian terhadap unit kode. Pengujian unit biasanya menguji fungsionalitas unit kode terkecil (*method*, *class*, atau komponen) dengan cara yang bisa diulang (Google Developer).

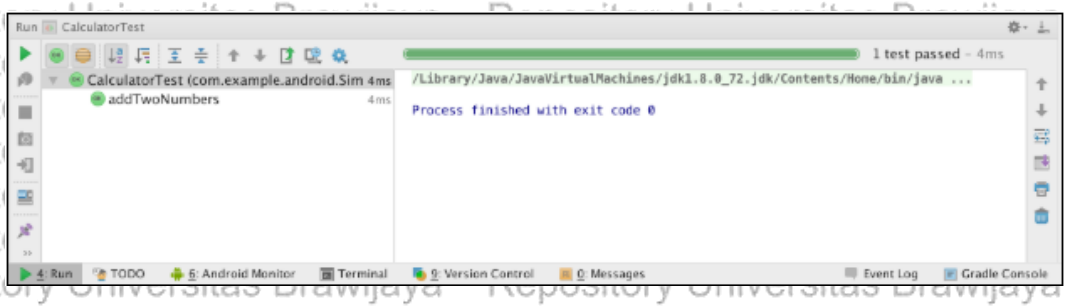
Contoh pengujian terhadap unit kode program pada aplikasi *Android Studio* terdapat pada Gambar 2.11.

```
@RunWith(JUnit4.class)
@SmallTest
public class CalculatorTest {
    private Calculator mCalculator;
    // Set up the environment for testing
    @Before
    public void setUp() {
        mCalculator = new Calculator();
    }

    // test for simple addition
    @Test
    public void addTwoNumbers() {
        double resultAdd = mCalculator.add(1d, 1d);
        assertThat(resultAdd, is(equalTo(2d)));
    }
}
```

Gambar 2.10 Contoh Pengujian Unit pada TDD

(Sumber : https://google-developer-training.gitbooks.io/android-developer-fundamentals-course-concepts/content/idn/Unit%201/32_c_testing_your_app.html)

Hasil terhadap pengujian unit kode program pada aplikasi *Android Studio* terdapat pada Gambar 2.12.



Gambar 2.11 Hasil Pengujian Unit pada TDD

(Sumber : https://google-developer-training.gitbooks.io/android-developer-fundamentals-course-concepts/content/idn/Unit%201/32_c_testing_your_app.html)

2.2.10 *White Box Testing*

White box testing ialah sebuah metode pengujian berdasarkan kasus uji yang diperoleh dari struktur kode dan logika internal. Pilihan kasus uji didasarkan pada implementasi perangkat lunak yang menguji sampai bagian terkecil dari sebuah program. *White box* testing dilakukan ketika pengembang mengetahui tentang struktur program, sehingga *white box testing* hanya dapat dilakukan oleh pengembang yang mengenali programnya beserta struktur kodenya. Dengan teknik ini, dimungkinkan untuk menguji setiap cabang dan keputusan dalam program. Ada beberapa teknik yang dipakai dalam salah satunya yaitu *white box testing* (Nidhra, 2012). Pada penelitian ini digunakan *basis path testing* untuk menguji unit kode program.

Pada *basis path testing*, *flow graph* digunakan untuk menentukan *cyclomatic complexity* sebuah kode program. *Flow graph* adalah grafik berarah yang terdiri dari node dan *control flow*. Node diekspresikan oleh sebuah lingkaran, yang mewakili satu atau lebih pernyataan, kondisi, prosedur program, atau konvergensi dua atau lebih node. *Control flow* dinyatakan oleh busur dengan panah atau garis, bisa disebut tepi, yang mewakili aliran control program. Dalam *flow graph*, sebuah simpul termasuk kondisi yang disebut simpul predikat, dan tepi dari simpul predikat harus menyatu pada sebuah simpul tertentu. Area yang didefinisikan oleh *edge* dan node disebut sebagai *region*. Untuk menghitung sebuah *cyclomatic complexity*, digunakan rumus V(G) = E-N+2 dengan V(G) sebagai *cyclomatic complexity*, E sebagai *edge*, dan N sebagai node.

2.2.11 *Black Box Testing*

Black box testing didasarkan pada kebutuhan atau spesifikasi desain perangkat lunak yang diuji. Contoh hasil yang diharapkan seperti spesifikasi kebutuhan atau desain, nilai yang dihitung oleh tangan, dan hasil simulasi. Pengujian fungsional difokuskan pada perilaku eksternal dari perangkat lunak sehingga *black box testing* bisa disebut juga dengan *behavioral testing*. Tipe pengujian yang dipakai dalam *black box testing* adalah pengujian non fungsional, pengujian sistem, pengujian *acceptance*, dan pengujian beta (Nidhra, 2012).

2.2.12 *Acceptance Test*

Acceptance test adalah pengujian untuk menentukan apakah suatu sistem memenuhi kriteria kebutuhan dan untuk menentukan apakah pengguna dapat menerima atau tidaknya sebuah sistem yang telah dirancang. Tujuan dari *acceptance test* ialah untuk memberi keyakinan bahwa sistem yang dikembangkan telah memenuhi kebutuhan bisnis pengguna. Salah satu metode yang dapat dilakukan dalam melakukan *acceptance* ialah *test driven development* (TDD). TDD merupakan bagian dari metode *agile* seperti *extreme programming* (XP). Dengan dilakukan TDD, dapat memungkinkan pengguna dan pengembang menghindari kesalahpahaman dalam mengembangkan perangkat lunak yang akhirnya mengurangi risiko kegagalan dalam mengembangkan perangkat lunak (Melnik, et. al., 2006).

Acceptance test penting dilakukan karena kebutuhan pengguna yang sering berubah-ubah. *Acceptance test* dilakukan sampai pengguna dan pengembang telah sepakat akan kebutuhan yang akan diimplementasikan. Tim pengembang terlebih dahulu menuliskan *acceptance test* sebelum kebutuhan diimplementasikan ke dalam kode program. *Acceptance test* dapat dijalankan secara otomatis atau manual sesuai dengan kebutuhan. Setelah pengujian dilakukan, tim harus sering menjalankannya sampai pengguna dan tim pengembang telah sepakat akan kebutuhan yang akan diimplementasikan. Manfaat dari dilakukannya *acceptance test* ialah menjaga desain tetap sederhana, dan menjadi acuan dalam melakukan implementasi kode program (Miller & Collins, 2002).

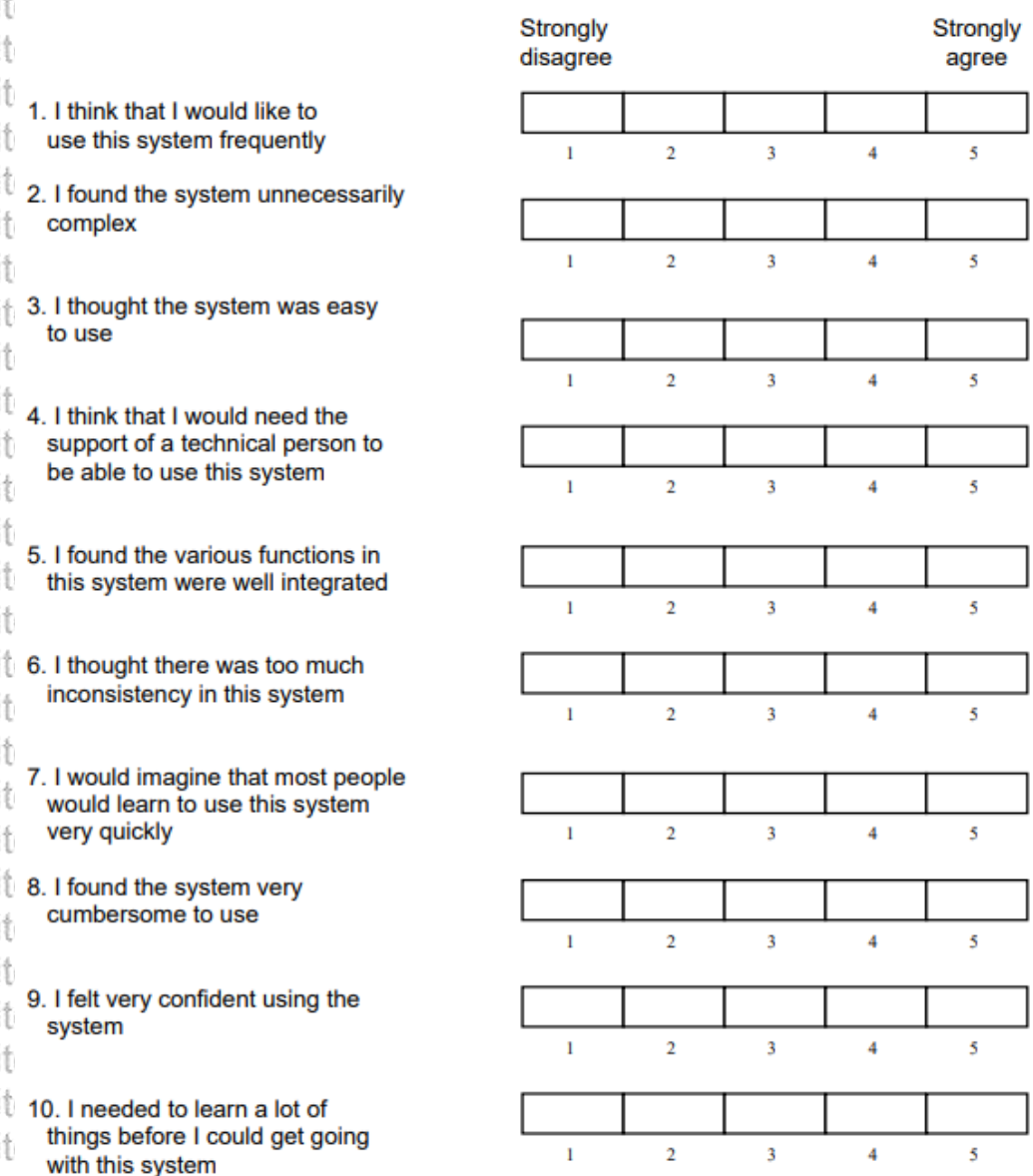
2.2.13 Pengujian *Usability*

Usability dalam sebuah perangkat lunak bukan kualitas dalam artian mutlak, melainkan *usability* disimpulkan sebagai pertimbangan dari pengguna dalam menggunakan sebuah perangkat lunak. *Usability* harus dilihat dalam hal konteks di mana ia digunakan, dan kesesuaiannya dengan konteks itu (Brooke, 2011). Ukuran *usability* sesuai dengan ISO 9241-11 harus memenuhi kriteria sebagai berikut :

1. Efektivitas (kemampuan pengguna untuk menyelesaikan tugas menggunakan sistem dan kualitas output)
2. Efisiensi (tingkat sumber daya yang dikonsumsi)
3. Kepuasan (reaksi subjektif pengguna untuk menggunakan sistem)

Untuk memenuhi kriteria tesebut, skala *usability* sederhana dikembangkan, salah satunya *System Usability Scale* (SUS). SUS terdiri dari sepuluh item sederhana yang memberikan pandangan globak tentang subjektif dari penilaian *usability* sebuah perangkat lunak. SUS adalah skala Likert, sering diasumsikan bahwa skala likert ialah skala berdasarkan pernyataan atau kuisioner pada skala poin 5 atau 7. Namun, skala likert disajikan dalam pernyataan yang responden

menunjukkan kesetujuan atau ketidaksetujuan yang dipilih dengan cermat. *Item* dari SUS digambarkan pada Gambar 2.13.



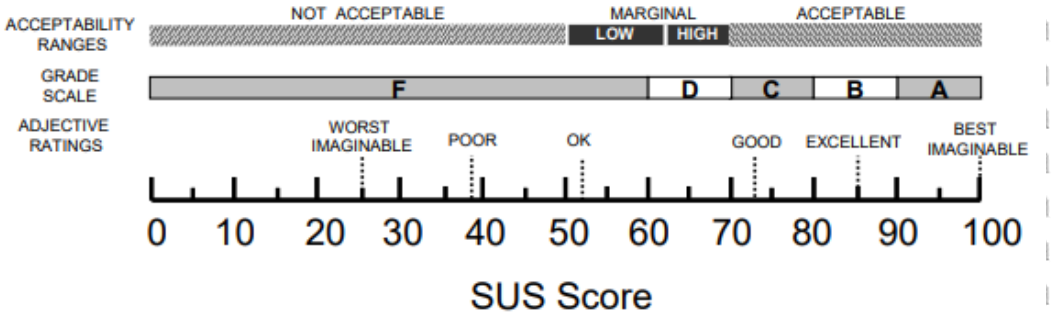
Gambar 2.12 *Item* dari SUS

Sumber : (Digital Equipment Corporation, 1986)

Skala SUS umumnya digunakan setelah responden memiliki kesempatan untuk menggunakan sistem yang sedang dievaluasi. Responden diminta untuk mencatat tanggapan langsung mereka untuk setiap item. Untuk menghitung skor SUS, yang dilakukan pertama kali ialah menghitung total dari setiap *item*. Setiap item berkontribusi antara 0 sampai 4. Untuk *item* nomor ganjil, skor kontribusinya adalah skor dikurangi 1. Untuk *item* nomor genap, skor kontribusinya adalah 5

dikurangi skor yang didapatkan. Setelah skor didapatkan, selanjutnya skor dikali dengan 2,5 untuk mendapatkan nilai SUS secara keseluruhan.

Menurut Bangor, Kortum & Miller (2009), terdapat *range* dalam mengetahui arti dari skala nilai SUS yang didapatkan yang ditunjukkan pada Gambar 2.14.



Gambar 2.13 Skala Penilaian SUS

Sumber : (Bangor, Kortum & Miller, 2009)

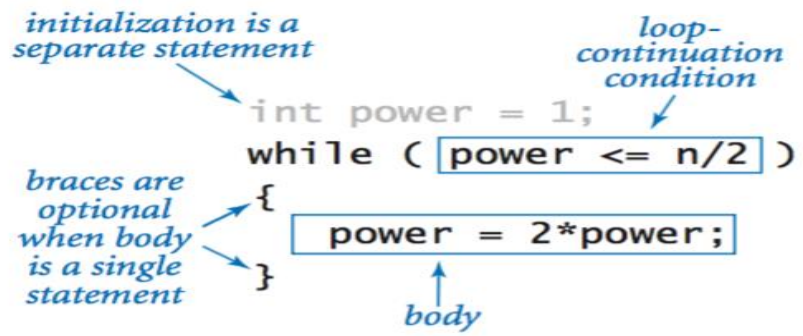
Pada Gambar 2.15, ditunjukkan bahwa ada 3 tingkatan *range acceptability* pada SUS untuk mengukur apakah sebuah aplikasi dapat diterima atau tidak. Termasuk dalam kategori tidak diterima jika skor SUS yang didapatkan sebesar 0-50. Kategori selanjutnya yaitu cukup, apabila skor SUS yang didapatkan sebesar 51-70. Kategori terakhir, yaitu diterima apabila skor SUS yang didapatkan nilainya sebesar 71-100.

2.2.14 Java

Beberapa aplikasi atau situs web mengharuskan penggunanya telah meng-*install* Java dalam komputernya. Java adalah salah satu bahasa pemrograman yang cepat, aman, dan bersifat dapat diandalkan (Oracle).

2.2.14.1 Perulangan - *while*

Perulangan *while* adalah salah satu cara untuk memungkinkan kita mengeksekusi sekelompok pernyataan berkali-kali sehingga kita bisa mengekspresikan perhitungan panjang tanpa menulis banyak kode (Sedgewick, R. dan Wayne, K., 2017). Contoh penggunaan perulangan *while* digambarkan pada Gambar 2.15.

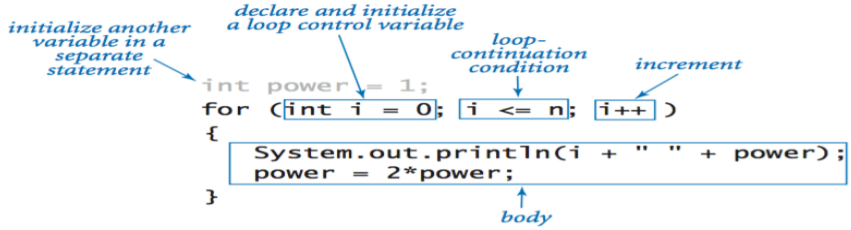


Gambar 2.14 Contoh Penggunaan Perulangan *while*

(Sumber : https://introcs.cs.princeton.edu/java/13flow/)

2.2.14.2 Perulangan - *for*

Perulangan for adalah susunan alternatif di Java yang lebih fleksibel saat menulis kode perulangan. Untuk notasi, banyak perulangan yang mengikuti skema dasar yang sama, yakni menginisialisasi variabel indeks ke nilai tertentu dan menggunakan perulangan *while* untuk menguji kondisi *output* yang melibatkan variabel indeks menggunakan pernyataan terakhir di perulangan sementara untuk memodifikasi variabel indeks (Sedgewick, R. dan Wayne, K., 2017). Perulangan *for* adalah cara langsung untuk mengekspresikan perulangan seperti itu yang digambarkan pada Gambar 2.16.



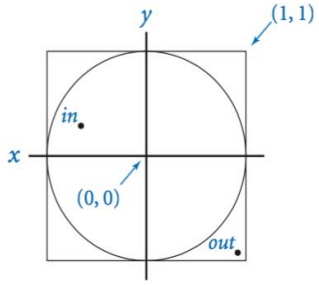
Gambar 2.15 Contoh Penggunaan Perulangan *for*

(Sumber : https://introcs.cs.princeton.edu/java/13flow/)

2.2.14.3 Perulangan – *do while*

Perulangan *do while* hampir sama dengan perulangan *while*. Bedanya, perulangan *do while* melakukan perulangan terlebih dahulu, lalu memeriksa kondisi atau syarat untuk melakukan perulangan. Contoh penggunaan perulangan *do while* digambarkan pada Gambar 2.17.

```
do
{  // Scale x and y to be random in (-1, 1).
   x = 2.0*Math.random() - 1.0;
   y = 2.0*Math.random() - 1.0;
} while (Math.sqrt(x*x + y*y) > 1.0);
```

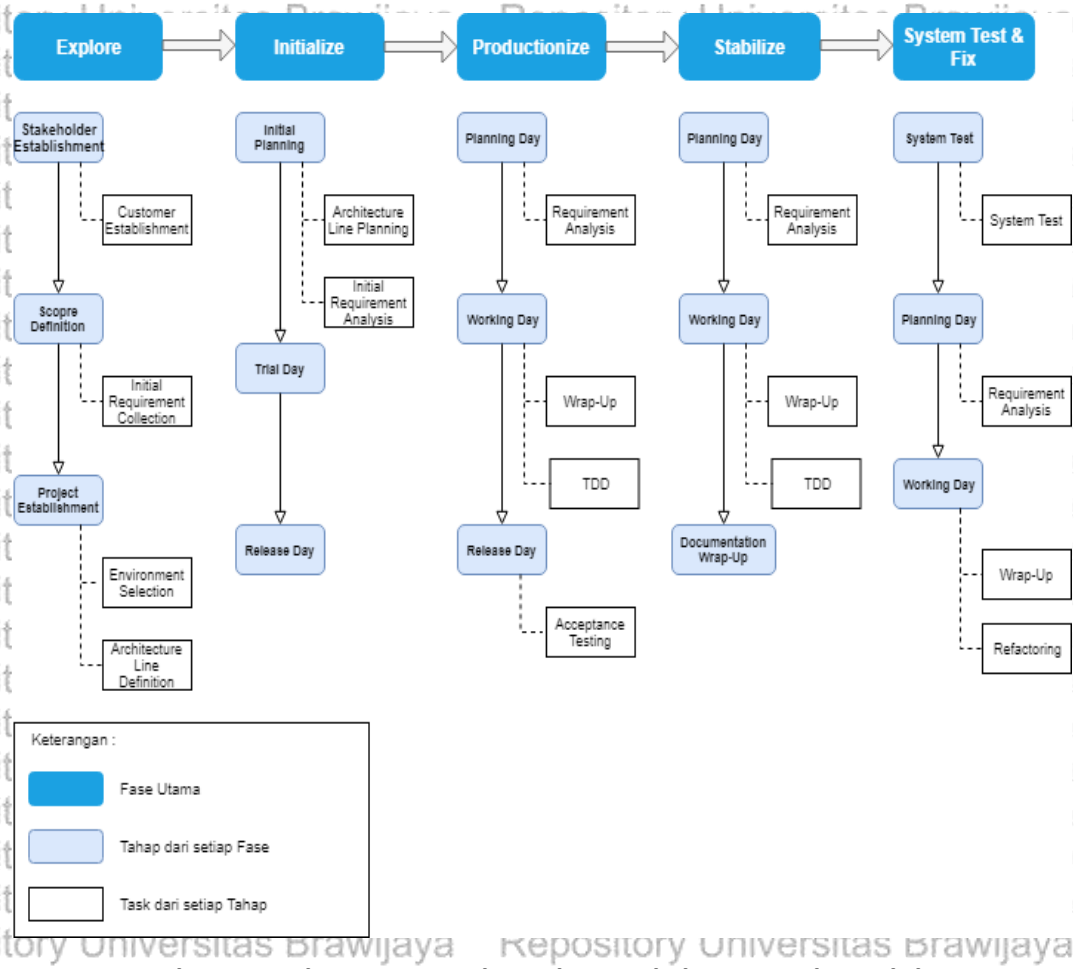


Gambar 2.16 Contoh Penggunaan Perulangan *do while*

(Sumber : https://introcs.cs.princeton.edu/java/13flow/)

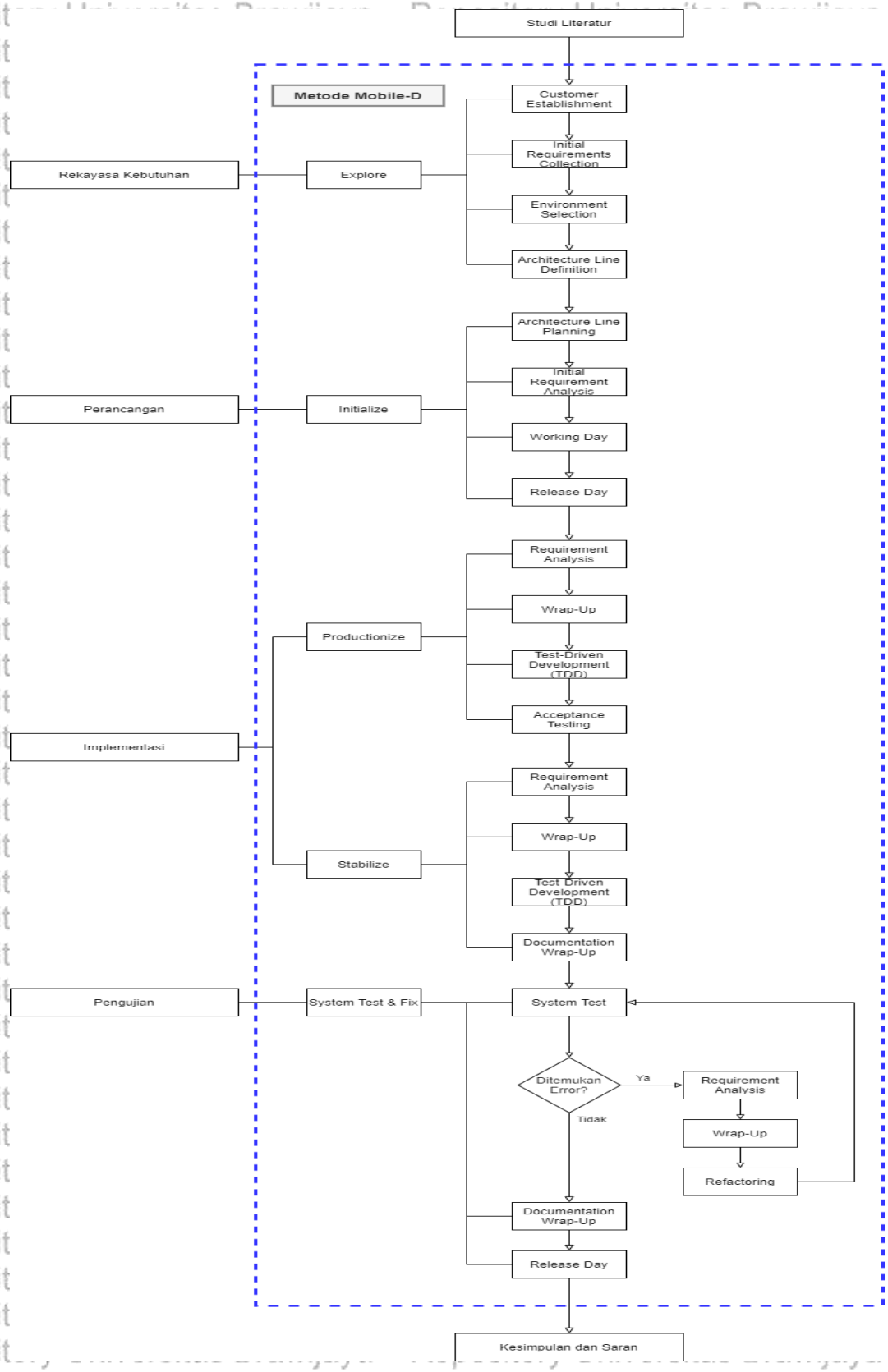
BAB 3 METODOLOGI

Penelitian ini mengadopsi hanya beberapa bagian dari semua fase dan *task* yang terdapat pada metode *Mobile-D*. Penulis melakukan adopsi dari beberapa fase, tahap dan *task* yang terdapat pada *Mobile-D* dan digambarkan pada Gambar 3.1.



Gambar 3.1 Adopsi Fase, Tahap, dan *Task* dari Metode Mobile-D

Sedangkan tahapan penelitian yang akan dilakukan pada penelitian implementatif pada pengembangan aplikasi BelajarJava terdapat pada Gambar 3.2.



Gambar 3.2 Tahapan Penelitian

3.1 Studi Literatur

Tahap ini dilakukan untuk mempelajari dan menerapkan konsep-konsep pada penelitian terkait yang telah dilakukan sebelumnya. Penelitian yang telah dikaji kemudian akan didapatkan metode yang sesuai untuk menyelesaikan permasalahan yang telah dirumuskan. Untuk mendukung penulisan penelitian ini, dikumpulkan beberapa sumber terpercaya seperti jurnal, buku atau laporan penelitian yang terdapat hubungannya dengan penelitian yang akan dilakukan. Tahap ini terdapat pembahasan kajian pustaka, *M-Learning*, *Firebase*, Metode Mobile-D, Java, dan *usability testing*.

3.2 Rekayasa Kebutuhan

Tahap ini mengadopsi fase *explore*. Pada fase *Explore*, pengembang harus memahami produk yang akan dikembangkan dan mempersiapkan segala kebutuhan yang akan dijadikan batasan dalam mengembangkan produk, mulai dari menentukan gambaran umum dan alur kerja aplikasi yang akan dikembangkan, identifikasi aktor, pemilihan lingkungan pengembangan sistem, menggali kebutuhan fungsional dan kebutuhan non-fungsional sistem, analisa data dan memodelkan kebutuhan ke dalam *use case*.

Pada tahap rekayasa kebutuhan, penulis mengadopsi fase *explore* yang terdiri dari beberapa tugas (*task*). *Task* pertama yang dilakukan ialah *customer establishment* dimana penulis melakukan identifikasi aktor dan menentukan mode elisitasi untuk menentukan kebutuhan sistem, dimana penulis menggunakan metode elisitasi secara tidak langsung (*off-site customer*) menggunakan survei kuisioner. Kemudian *task* kedua yang dilakukan ialah *initial requirements collection*, dimana penulis menuliskan kebutuhan awal dari sistem baik kebutuhan fungsional yang dimodelkan ke diagram *use* case dan kebutuhan non-fungsional. *Task* selanjutnya yang dilakukan ialah *environment selection* dimana penulis menentukan lingkungan pengembangan. *Task* terakhir ialah *architecture line definition* dimana penulis menggambarkan sistem secara umum lalu menentukan alur kerja dari sistem yang akan dikembangkan.

3.3 Perancangan Sistem

Tahap ini mengadopsi fase *initialize*. Pada fase ini, pengembang mulai membangun karakteristik dari sistem yang kebutuhannya telah didefinisikan pada fase *explore*. Penulis menganalisis dan merumuskan kebutuhan dimana pada fase ini terjadi iterasi 0 untuk mengetahui *feedback* terhadap rancangan awal pada sistem, yakni struktur data dan perancangan antarmuka.

Pada fase ini, penulis mengadopsi tiga fase yang terdapat pada metode *Mobile-D*, yakni *initial planning* yang mengadopsi *task architecture line planning* dan *task initial requirement analysis*, *trial day*, dan *release day*. Pada *task* pertama yakni *architecture line planning*, penulis membuat perancangan arsitektural pada sistem terlebih dahulu untuk menentukan kebutuhan komponen utama dan komponen pendukung penyusun sistem. Kemudian, *task* selanjutnya yang

dilakukan yakni *initial requirement analysis*, analisis dilakukan terhadap kebutuhan yang telah didefinisikan menggunakan *product backlog*, tujuannya yaitu untuk menentukan prioritas kebutuhan yang akan diimplementasi terlebih dahulu. Hasil dari analisis kebutuhan menggunakan *product backlog* terdapat pada Tabel 4.3.

Tahap selanjutnya yaitu *working day*, dimana pada tahap ini spesifikasi kebutuhan yang telah ditentukan kemudian dirancang untuk menentukan batasan-batasan dalam implementasi. Pada tahap *working day*, penulis membuat perancangan terhadap *sequence diagram*, *class diagram*, basis data, perancangan algoritma, dan perancangan antarmuka.

Tahap terakhir yaitu *release day*, dimana hasil dari perancangan kemudian diberikan kepada pengguna untuk mengetahui *feedback* dari pengguna. Dari *feedback* tersebut kemudian pengembang bisa menambahkan ataupun menghapus beberapa fungsionalitas sistem agar sistem yang dikembangkan telah sesuai dan dapat diimplementasikan pada tahap berikutnya.

3.4 Implementasi

Pada tahap ini dilakukan dua fase yang diadopsi dari metode *mobile-d*, yaitu fase *productionize* dan *stabilize*. Pada fase *productionize*, penulis mengimplementasikan kebutuhan dan perancangan yang terjadi pada iterasi 0 yang kemudian akan diberikan kepada calon pengguna kembali untuk didapatkan *feedback* pengguna. Di akhir fase *productionize*, akan terjadi iterasi 1 sebelum memasuki fase *stabilize*. Pada fase *stabilize*, aplikasi yang dikembangkan sudah final beserta dokumentasi lengkap dari sistem dan aplikasi yang dikembangkan sudah siap untuk diujikan pada fase *system test & fix*.

Pada fase *productionize*, penulis mengadopsi tahap *planning day* dengan *task* yang dilakukan ialah *requirement analysis*. Selanjutnya tahap *working day* dengan *task* yang dilakukan ialah *wrap-up* dan TDD. Terakhir tahap *release day* dengan *task* yang dilakukan ialah *acceptance testing*.

Task pertama yang dilakukan ialah *requirement analysis*, dimana hasil dari fase *initialize* dijadikan input untuk menentukan perubahan fungsionalitas pada sistem. Hasil dari analisis kebutuhan pada *task* ini terdapat pada Tabel 4.4 dimana terdapat fungsionalitas yang dihapus dari sistem. *Task* selanjutnya ialah *wrap-up* dimana pada *task* ini penulis melakukan *update* terhadap dokumentasi berdasarkan hasil yang didapatkan dari *task requirement analysis*. Perubahan yang dimaksud ialah perubahan terhadap fungsionalitas sistem dimana akan terjadi perubahan pada *use case diagram*, basis data, antarmuka, dan yang lainnya sesusai dengan kebutuhan terbaru. *Task* selanjutnya ialah TDD untuk menguji perancangan algoritma yang telah dilakukan. *Task* terakhir ialah *acceptance test*, dimana terjadi iterasi 1 yang dikarenakan aplikasi versi awal telah dibangun dan siap diberikan kepada pengguna untuk mengetahui *feedback* selanjutnya dari pengguna. Setelah iterasi 1, pengembang tidak bisa merubah fungsionalitas mayor sistem melainkan pengembang hanya dapat penambahan fungsionalitas pada

fungsi-fungsi minor saja. Pengembang tidak bisa merubah fungsionalitas mayor sistem dikarenakan dapat merusak integrasi pada sistem yang berakibat pengembang harus memulai dari awal lagi pengembangan pada sistem.

Pada fase *stabilize*, penulis melakukan persis seperti yang dilakukan pada fase *productionize*. Penulis melakukan *task requirement analysis* dengan input berupa hasil dari iterasi 1 yang telah dilakukan dan penulis mengadopsi *task documentation wrap-up* untuk menggantikan *task release day* karena iterasi hanya dilakukan sebanyak 2 kali. Pada *task documentation wrap-up*, penulis menuliskan hasil dokumentasi lengkap dari sistem dan aplikasi telah final sehingga aplikasi kemudian dapat diujikan ataupun perbaikan pada *error* atau *bug* yang ditemui.

3.5 Pengujian dan Analisis

Tahap ini merupakan fase terakhir, yaitu fase *system test & fix*. Yang pertama kali dilakukan pada fase *system & fix* ialah tahap *system test*. Dimana pada tahap ini sistem dilakukan uji secara keseluruhan. Pada tahap ini penulis melakukan beberapa pengujian, yaitu :

1. Pengujian unit, dimana dilakukan pengujian terhadap klas atau *method* yang penting pada sistem. Pengujian ini dilakukan menggunakan metode *white box testing* dengan teknik *basis path testing*. Pengujian unit dilakukan kepada kode program yang sudah jadi dimana sebelumnya kode program tersebut telah diuji terlebih dahulu menggunakan TDD. Tujuan dari pengujian unit adalah untuk menentukan jumlah *cyclomatic complexity* pada algoritma.
2. Pengujian validasi, dimana aplikasi diuji menggunakan metode *black box testing*. Pengujian ini bertujuan untuk mengetahui apakah aplikasi yang telah dikembangkan sesuai dengan daftar kebutuhan atau tidak.
3. Pengujian *usability*, dimana dilakukan pengujian aplikasi terhadap pengguna untuk mengetahui tingkatan kepuasan pengguna. Pada pengujian ini, penulis menggunakan kuisioner SUS pada pengguna.

Dilakukan juga latihan sebelum menggunakan aplikasi (*pre-test*) dan setelah menggunakan aplikasi (*post-test*) untuk mengukur apakah aplikasi efektif dalam pembelajaran mahasiswa.

Setelah melakukan *system test*, jika pengembang menemukan cacat, maka akan dilakukan iterasi perbaikan yang mencakup *task requirement analysis*, *wrap-up*, dan *task refactoring* untuk memperbaiki cacat yang ditemukan pada sistem. Jika tidak ditemukan cacat, maka berlanjut ke *task documentation wrap-up* untuk mengumpulkan seluruh dokumentasi terhadap perangkat lunak. Terakhir, yakni tahap *release day*, dimana aplikasi sudah siap didistribusikan kepada pengguna.

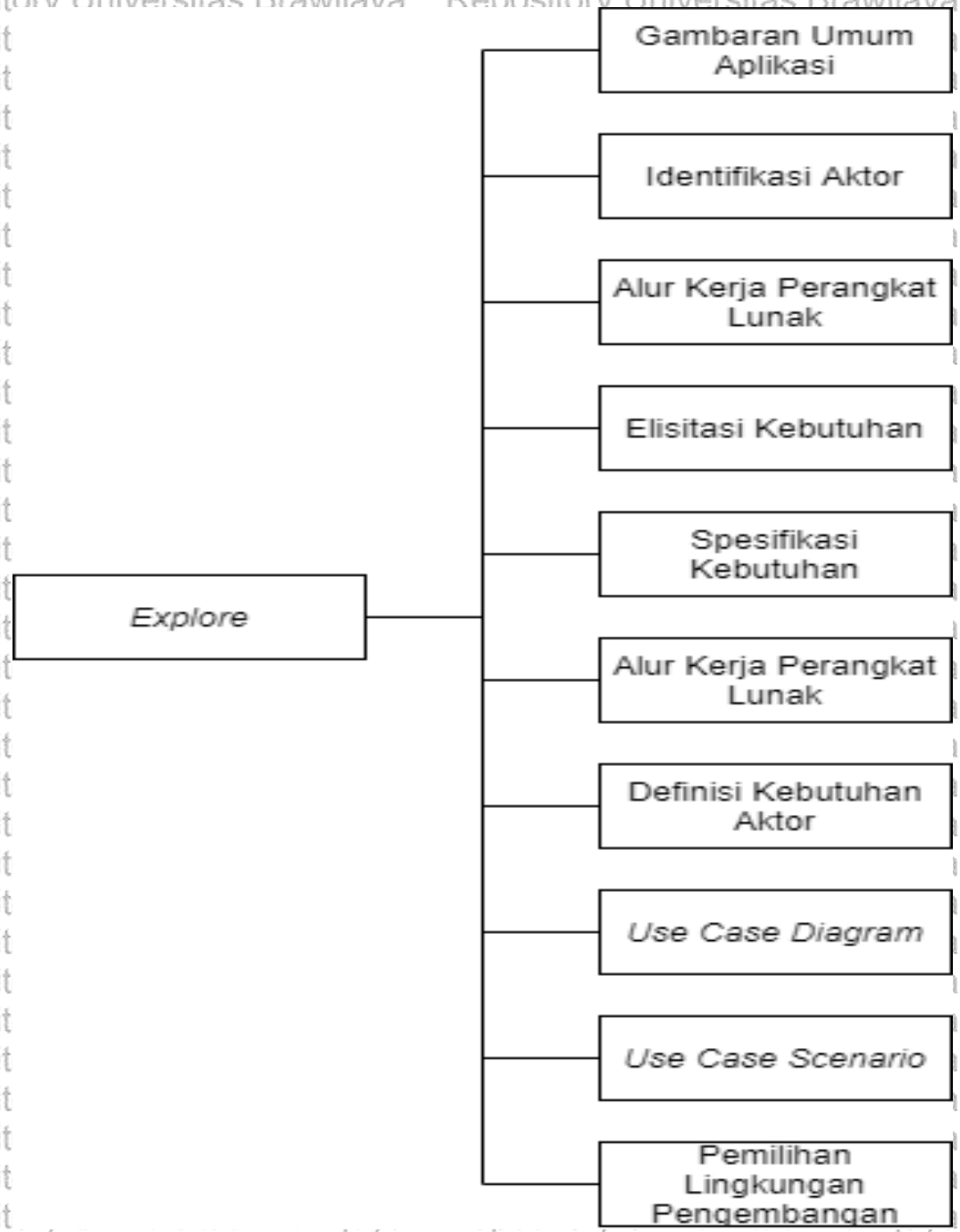
3.6 Kesimpulan dan Saran

Pada tahap ini dilakukan pengambilan kesimpulan setelah semua tahapan *Mobile-D* selesai dilakukan. Kesimpulan diambil berdasarkan pada rumusan

masalah yang telah dibuat. Lalu pemberian saran dilakukan untuk memberi usulan yang bertujuan untuk mengembangkan penelitian atau aplikasi selanjutnya yang berkaitan dengan penelitian ini.

BAB 4 REKAYASA KEBUTUHAN

Pada bab ini akan membahas rekayasa kebutuhan dari aplikasi *BelajarJava*. Tahap rekayasa kebutuhan mengadopsi fase *explore* pada *Mobile-D*. *Output* yang dihasilkan saat fase *explore* dilakukan terdapat pada Gambar 4.1.



Gambar 4.1 Diagram Pohon Rekayasa Kebutuhan

4.1 Analisis Kebutuhan (*Explore*)

Pada fase *explore*, dilakukan penggalian kebutuhan awal dengan memakai metode *off-site customer* atau kuisioner kepada total 62 responden. Responden terbagi menjadi 2 kelompok, yakni mahasiswa sebesar 58 responden dan dosen sebesar 4 responden. Hasil kuisioner terhadap penggalian kebutuhan awal Aplikasi *BelajarJava* terdapat pada Lampiran A.1 dan Lampiran A.2.

4.1.1 Gambaran Umum Perangkat Lunak BelajarJava

Aplikasi *BelajarJava* adalah aplikasi berbasis Android yang bertujuan untuk mahasiswa dalam mengikuti mata kuliah Pemrograman Dasar yang mempelajari dasar-dasar pemrograman Bahasa Java. Aplikasi juga dibangun untuk membantu dosen yang mengajar mata kuliah Pemrograman Dasar dalam bentuk memberi pertanyaan tentang dasar-dasar pemrograman Bahasa Java kepada para mahasiswa-nya. Selain itu, aplikasi ini juga dapat membantu para dosen untuk mengukur seberapa besar kemampuan mahasiswa dalam menjawab pertanyaan tentang dasar-dasar pemrograman Bahasa Java.

4.1.2 Identifikasi Aktor

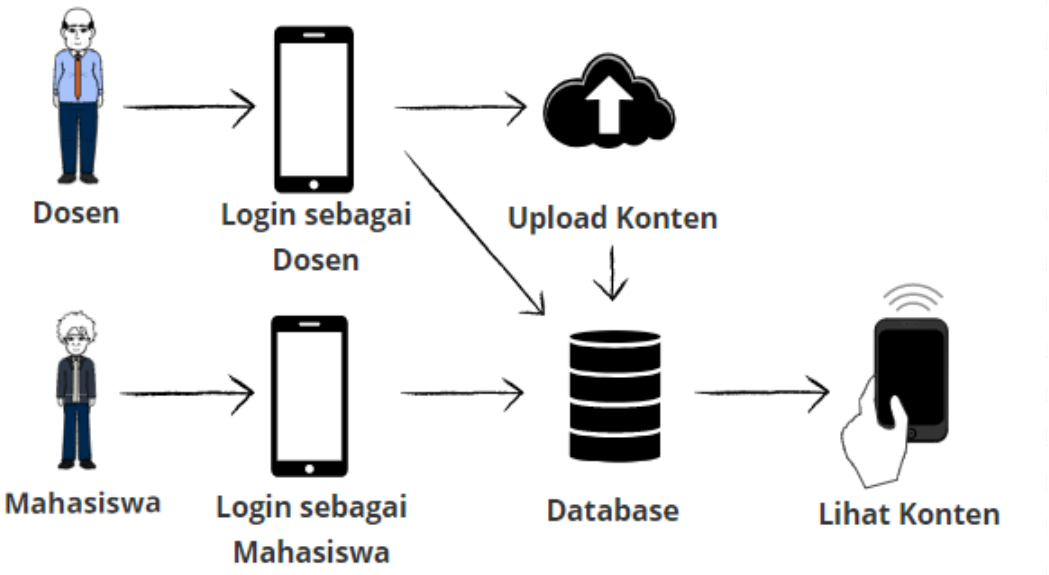
Pada bagian ini, dilakukan identifikasi terhadap aktor pada aplikasi BelajarJava yang akan dilakukan oleh aktor. Penjelasan aktor yang akan berinteraksi pada aplikasi ini yang merupakan hasil dari tahap identifikasi aktor dijelaskan pada Tabel 4.1.

Tabel 4.1 Identifikasi Aktor

Kategori Aktor	Deskripsi
Guest	*Guest* adalah aktor yang dapat melakukan fungsi *register* ataupun *login*
Mahasiswa	Mahasiswa adalah aktor yang dapat melihat pertanyaan, menjawab pertanyaan dan melihat evaluasi hasil jawaban dari pertanyaan yang sudah dijawab
Dosen	Dosen adalah pengguna yang dapat melihat hasil statistik mahasiswa dan mengubah pertanyaan pada program

4.1.3 Alur Kerja Perangkat Lunak

Aplikasi *BelajarJava* dapat diakses oleh dua aktor utama, yaitu mahasiswa dan dosen. *User* dapat mengakses aplikasi sebagai mahasiswa ataupun dosen yang memiliki fungsi berbeda dalam sistem sehingga alur kerja yang terdapat pada aplikasi *BelajarJava* berbeda antara aktor mahasiswa dan dosen. Alur kerja yang terdapat pada aplikasi *BelajarJava* terdapat pada Gambar 4.2.



Gambar 4.2 Alur Kerja Aplikasi *BelajarJava*

Pada Gambar 4.2, digambarkan alur kerja yang berbeda antara aktor mahasiswa dan dosen. Pada aktor dosen, fungsi yang dapat dilakukan adalah fungsi yang terkait dengan melihat konten dan *upload* konten. Sedangkan pada aktor mahasiswa, fungsi yang dapat dilakukan hanya fungsi yang terkait dengan melihat konten.

4.1.4 Definisi Kebutuhan Aktor

Pada bagian ini, kebutuhan aktor yang telah dilakukan berdasarkan kuisioner didefinisikan kebutuhannya. Penulis menyebar kuisioner terhadap kebutuhan awal kepada 25 responden yang dibagi menjadi 20 mahasiswa FILKOM UB dan 5 dosen. Definisi kebutuhan pada aktor mahasiswa terdapat pada Tabel 4.2, sedangkan definisi kebutuhan pada aktor dosen terdapat pada Tabel 4.3.

Tabel 4.2 Definisi Kebutuhan Mahasiswa

No	Nama Kebutuhan
1	Saya ingin sistem menyediakan konten materi singkat
2	Saya butuh sistem disediakan contoh *source code*
3	Saya butuh sistem disediakan latihan
4	Saya butuh sistem disediakan *tutorial*

Tabel 4.2 adalah hasil dari fase *explore* tahap *initial requirements collection.* Kuisioner yang telah disebarkan oleh penulis terhadap mahasiswa yang terlibat dalam sistem. Hasil definisi kebutuhan tersebut dijelaskan secara spesifik dan diberikan lagi kepada pengguna untuk di tentukan prioritas kebutuhan tersebut berdasarkan nilai pengguna.

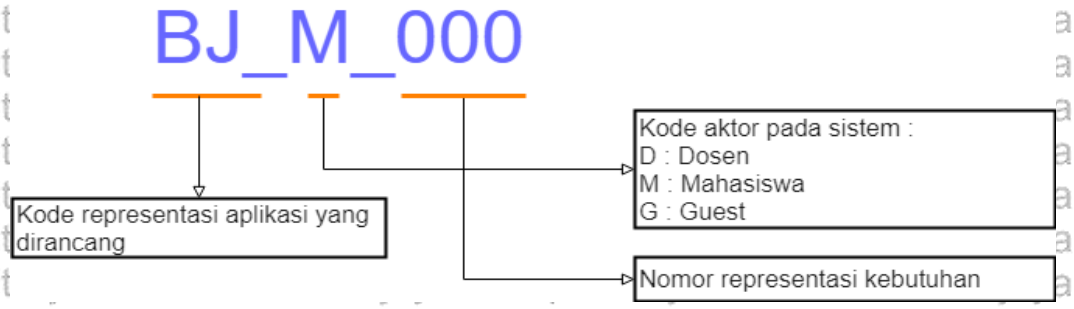
Tabel 4.3 Definisi Kebutuhan Dosen

No	Nama Kebutuhan
1	Saya butuh fitur CRUD pada materi singkat
2	Saya butuh fitur CRUD pada materi *source code*
3	Saya butuh fitur CRUD pada latihan
4	Saya butuh fitur CRUD pada *tutorial*

Tabel 4.3 adalah hasil dari fase *explore* tahap *initial requirements collection.* Kuisioner yang telah disebarkan oleh penulis terhadap dosen yang terlibat dalam sistem. Hasil definisi kebutuhan tersebut dijelaskan secara spesifik dan diberikan lagi kepada pengguna untuk menentukan prioritas kebutuhan tersebut berdasarkan nilai pengguna.

4.1.5 Spesifikasi Kebutuhan Fungsional

Spesifikasi kebutuhan fungsional dikelompokkan menjadi tiga berdasarkan aktornya, yaitu *guest*, mahasiswa, dan dosen. Setiap kebutuhan fungsional diberikan kode seperti yang terdapat pada Gambar 4.3.



Gambar 4.3 Aturan Penomoran Kode Kebutuhan Fungsional

Kebutuhan fungsional *guest* dijelaskan dalam spesifikasi kebutuhan fungsional mahasiswa yang terdapat pada Tabel 4.4.

Tabel 4.4 Spesifikasi Kebutuhan Fungsional *Guest*

No	Kode Fungsi	Nama Fungsi	Deskripsi
1	BJ_G_001	*Login*	Sistem dapat diakses oleh mahasiswa dan dosen melalui proses autentikasi
2	BJ_G_002	*Register*	Sistem menyediakan fungsi *register* kepada *guest*

Kebutuhan fungsional mahasiswa dijelaskan dalam spesifikasi kebutuhan fungsional mahasiswa yang terdapat pada Tabel 4.5.

Tabel 4.5 Spesifikasi Kebutuhan Fungsional Mahasiswa

No	Kode Fungsi	Nama Fungsi	Deskripsi
1	BJ_M_001	Lihat materi	Sistem menyediakan fitur lihat konten materi seputar Bahasa Java kepada mahasiswa
2	BJ_M_002	Lihat contoh *source code*	Sistem menyediakan fitur lihat contoh *source code* seputar Bahasa Java kepada mahasiswa
3	BJ_M_003	Latihan	Sistem menyediakan fitur evaluasi kepada pengguna terhadap pertanyaan yang telah dijawab
4	BJ_M_004	*Tutorial*	Sistem menyediakan fitur *tutorial* kepada pengguna
5	BJ_M_005	Logout	Sistem menyediakan fungsi keluar dari sistem yang akan kembali ke halaman *login*

Kebutuhan fungsional dosen dijelaskan dalam spesifikasi kebutuhan fungsional dosen yang terdapat pada Tabel 4.6.

Tabel 4.6 Spesifikasi Kebutuhan Fungsional Dosen

No	Kode Fungsi	Nama Fungsi	Deskripsi
1	BJ_D_001	Buat materi singkat	Sistem menyediakan fitur untuk membuat konten materi
2	BJ_D_002	Ubah materi singkat	Sistem menyediakan fitur untuk mengubah isi konten materi
3	BJ_D_003	Hapus materi singkat	Sistem menyediakan fitur untuk menghapus konten materi

4	BJ_D_004	Buat contoh *source code*	Sistem menyediakan fitur untuk membuat konten contoh *source code*
5	BJ_D_005	Ubah contoh *source code*	Sistem menyediakan fitur untuk mengubah isi konten contoh *source code*
6	BJ_D_006	Buat latihan	Sistem menyediakan fitur untuk membuat konten latihan
7	BJ_D_007	Ubah latihan	Sistem menyediakan fitur untuk mengubah konten latihan
8	BJ_D_008	Hapus latihan	Sistem menyediakan fitur untuk menghapus konten latihan
9	BJ_D_009	Buat *tutorial*	Sistem menyediakan fitur untuk membuat konten *tutorial*
10	BJ_D_010	Ubah *tutorial*	Sistem menyediakan fitur untuk mengubah konten *tutorial*
11	BJ_D_011	Hapus *tutorial*	Sistem menyediakan fitur untuk menghapus konten *tutorial*
12	BJ_M_007	Logout	Sistem menyediakan fungsi keluar dari sistem yang akan kembali ke halaman login

4.1.6 Spesifikasi Kebutuhan Non-Fungsional

Spesifikasi kebutuhan non-fungsional pada aplikasi BelajarJava terdapat pada Tabel 4.7. Parameter kebutuhan non-fungsional yang dibutuhkan adalah *usability* karena parameter *usability* dapat menentukan apakah perangkat lunak yang dikembangkan mudah digunakan dan dapat diterima oleh pengguna.

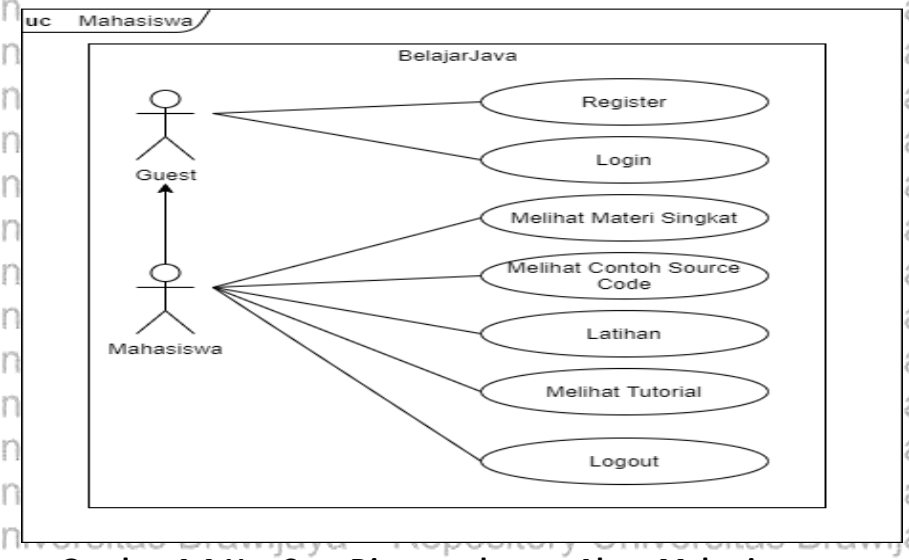
Tabel 4.7 Spesifikasi Kebutuhan Non-Fungsional

No	Parameter	Deskripsi
1	*Usability*	Perangkat lunak dapat dengan mudah digunakan dan dapat diterima oleh pengguna

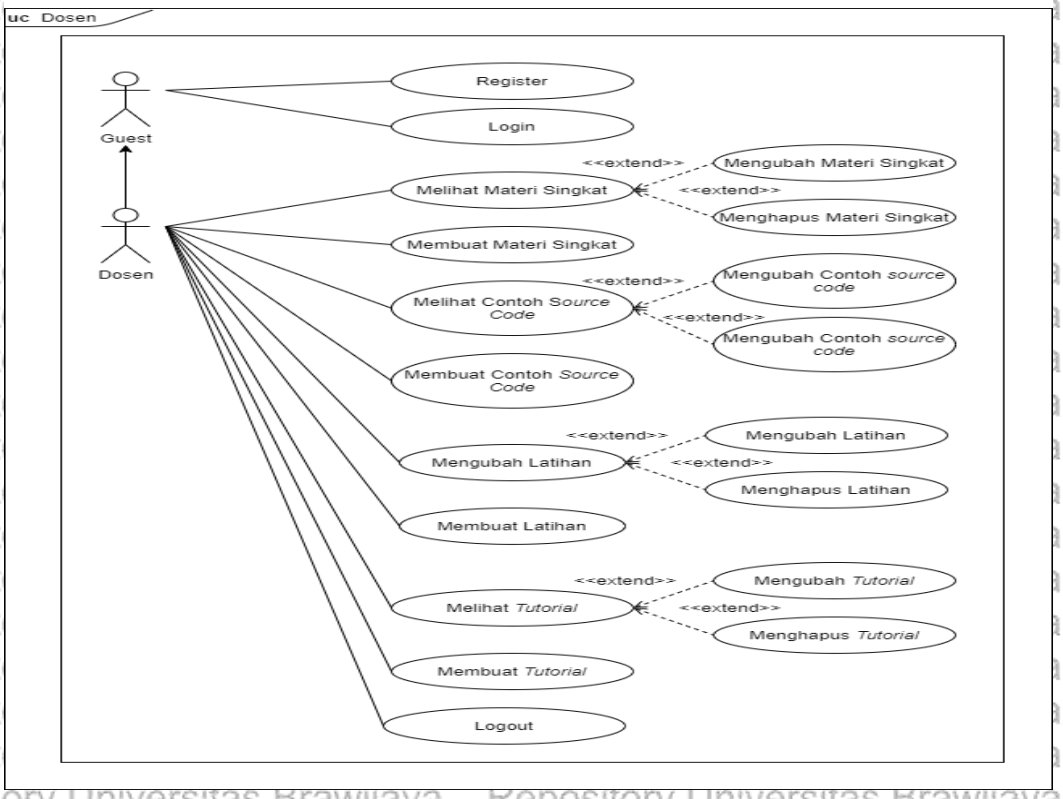
4.1.7 *Use Case Diagram*

Use case diagram pada subbab ini merupakan *use case diagram* hasil rancangan awal pada fase *initialize* dari identifikasi aktor dan definisi kebutuhan aktor yang telah dilakukan.

Pada *Use case diagram* ini *guest* didefinisikan sebagai pengguna yang memiliki hak khusus sebagai mahasiswa ataupun dosen pada aplikasi *BelajarJava*. *Use case diagram* untuk ini adalah hasil dari iterasi 0 yang digambarkan pada Gambar 4.4 dan Gambar 4.5.



Gambar 4.4 *Use Case Diagram* dengan Aktor Mahasiswa



Gambar 4.5 *Use Case Diagram* dengan Aktor Dosen

4.1.8 *Use Case Scenario*

Use case scenario adalah penjelasan lengkap dari *use case diagram* yang telah didefinisikan. *Use case scenario* yang dijelaskan merupakan penjelasan dari *use case diagram* yang telah dibuat pada tahap *initial requirements collection* pada fase *explore*.

4.1.8.1 *Use Case Scenario Guest*

Use case scenario pada Tabel 4.8 menjelaskan alur yang terjadi pada *use case login*. Dalam *use case scenario login*, aktor yang dilibatkan adalah *guest*. Hasil yang akan didapatkan setelah menjalani alur adalah sistem yang menampilkan beranda setelah alur *login* berhasil, lalu terdapat dua alur alternatif jika terdapat proses yang gagal dalam melakukan alur *login*.

Tabel 4.8 *Use Case Scenario Login*

Nama *Use Case*	*Login*
Aktor	*Guest*
Deskripsi	Pengguna melakukan *login* sebagai mahasiswa ataupun dosen
Pra-Kondisi	Sistem menampilkan halaman dan *form* untuk melakukan *login*

Alur	1. Aktor mengisi form yang telah disediakan 2. Aktor mengklik *login* 3. Autentifikasi sistem 4. Pengguna masuk ke dalam aplikasi
Post-Kondisi	Sistem menampilkan halaman Beranda
Alternatif	3a. *email* dan *password* tidak sesuai Pesan *error* ditampilkan sistem dan meminta pengguna untuk mengisi kembali *email* dan *password* 3b. Salah satu atau kedua kolom *email* atau password tidak terisi. Sistem menampilkan pesan *error* dan meminta pengguna untuk mengisi *email* dan *password*

Use case scenario pada Tabel 4.9 menjelaskan alur yang terjadi pada *use case register*. Dalam *use case scenario register*, aktor yang dilibatkan adalah *guest*. Hasil yang akan didapatkan setelah menjalani alur adalah aktor telah terdaftar dalam sistem, lalu terdapat dua alur alternatif jika terdapat proses yang gagal dalam melakukan alur *register*.

Tabel 4.9 *Use Case Scenario Register*

Nama *Use Case*	*Register*
Aktor	*Guest*
Deskripsi	Pengguna melakukan register sebagai mahasiswa agar dapat mengakses sistem
Pra-Kondisi	Aktor memilih menu *register* pada halaman *login*
Alur	1. *Form register* ditampilkan oleh sistem 2. Aktor mengisi form *register* 3. Aplikasi menampilkan notifikasi bahwa registrasi sukses 4. Pengguna masuk ke dalam aplikasi
Post-Kondisi	Aktor terdaftar dalam sistem
Alternatif	2a. *Password* kurang dari 8 digit Sistem menampilkan pesan *error* dan meminta pengguna untuk mengisi kolom *password* sama dengan atau lebih dari 8 digit 2b. Salah satu atau ketiga kolom NIM/NIP, email atau *password* tidak terisi Sistem menampilkan pesan *error* dan meminta pengguna untuk mengisi NIM/NIP, *email* dan *password*

4.1.8.2 *Use Case Scenario* Mahasiswa

Use case scenario pada Tabel 4.10 menjelaskan alur yang terjadi pada *use case* lihat materi. Dalam *use case scenario* lihat materi, aktor yang dilibatkan adalah mahasiswa. Hasil yang akan didapatkan setelah menjalani alur adalah sistem yang menampilkan materi tentang pemrograman Java, lalu tidak terdapat alur alternatif dari alur yang terjadi.

Tabel 4.10 *Use Case Scenario* Melihat Materi Singkat

Nama Use Case	Melihat materi singkat
Aktor	Mahasiswa
Deskripsi	Aktor mempelajari materi seputar Pemrograman Java
Pra-Kondisi	Aktor login sebagai mahasiswa
Alur	1. Sistem menampilkan halaman beranda atau halaman terakhir yang dibuka oleh pengguna 2. Aktor memilih menu materi singkat pada tab menu 3. Aktor memilih materi singkat apa yang diinginkan
Post-Kondisi	Sistem menampilkan konten materi seputar pemrograman Java
Alternatif	-

Use case scenario pada Tabel 4.11 menjelaskan alur yang terjadi pada *use case* lihat contoh *source code*. Dalam *use case scenario* lihat contoh *source code*, aktor yang dilibatkan adalah mahasiswa. Hasil yang akan didapatkan setelah menjalani alur adalah sistem yang menampilkan contoh *source code* tentang pemrograman Java, lalu tidak terdapat alur alternatif dari alur yang terjadi.

Tabel 4.11 *Use Case Scenario* Melihat Contoh *Source Code*

Nama Use Case	Melihat contoh *source code*
Aktor	Mahasiswa
Deskripsi	Aktor mempelajari *source code* seputar pemrograman Java
Pra-Kondisi	Aktor telah login sebagai mahasiswa
Alur	1. Sistem menampilkan halaman beranda atau halaman terakhir yang dibuka oleh pengguna 2. Aktor memilih menu contoh *source code* pada tab menu 3. Aktor memilih konten contoh *source code* apa yang diinginkan
Post-Kondisi	Sistem menampilkan contoh *source code* Bahasa Java
Alternatif	-

Use case scenario pada Tabel 4.12 menjelaskan alur yang terjadi pada *use case* latihan. Dalam *use case scenario* latihan, aktor yang dilibatkan adalah mahasiswa. Hasil yang akan didapatkan setelah menjalani alur adalah sistem yang menampilkan nilai berdasarkan latihan yang telah dikerjakan, lalu tidak terdapat alur alternatif dari alur yang terjadi.

Tabel 4.12 *Use Case Scenario* Latihan

Nama Use Case	Latihan
Aktor	Mahasiswa
Deskripsi	Aktor menjawab pertanyaan seputar pemrograman Java
Pra-Kondisi	Aktor telah *login* sebagai mahasiswa
Alur	1. Sistem menampilkan halaman beranda atau halaman terakhir yang dibuka oleh pengguna 2. Aktor memilih menu latihan pada tab menu 3. Aktor memilih materi latihan apa yang diinginkan 4. Aktor mengerjakan latihan
Post-Kondisi	Sistem menampilkan nilai berdasarkan latihan yang telah dikerjakan
Alternatif	-

Use case scenario pada Tabel 4.13 menjelaskan alur yang terjadi pada *use case* melihat *tutorial*. Dalam *use case scenario* latihan, aktor yang dilibatkan adalah mahasiswa. Hasil yang akan didapatkan setelah menjalani alur adalah sistem yang menampilkan nilai berdasarkan latihan yang telah dikerjakan, lalu tidak terdapat alur alternatif dari alur yang terjadi.

Tabel 4.13 *Use Case Scenario* Melihat *Tutorial*

Nama Use Case	Melihat *tutorial*
Aktor	Mahasiswa
Deskripsi	Aktor melihat *tutorial* Pemrograman Java
Pra-Kondisi	Aktor telah *login* sebagai mahasiswa
Alur	1. Sistem menampilkan halaman beranda atau halaman terakhir yang dibuka oleh pengguna 2. Aktor memilih menu *tutorial* pada *tab menu* 3. Aktor memilih konten *tutorial* apa yang ingin dilihat
Post-Kondisi	Sistem menampilkan konten *tutorial*
Alternatif	-

Use case scenario pada Tabel 4.14 menjelaskan alur yang terjadi pada *use case* melihat *logout*. Dalam *use case scenario logout*, aktor yang dilibatkan adalah mahasiswa dan dosen. Hasil yang akan didapatkan setelah menjalani alur adalah sistem menampilkan halaman *login*.

Tabel 4.14 *Use Case Scenario* Melihat *Logout*

Nama Use Case	*Logout*
Aktor	Mahasiswa dan Dosen
Deskripsi	Aktor keluar dari sistem
Pra-Kondisi	Aktor telah *login* sebagai mahasiswa atau dosen
Alur	1. Aktor memilih menu *logout* 2. Sistem membatalkan autentifikasi aktor
Post-Kondisi	Sistem menampilkan halaman *login.*
Alternatif	-

4.1.8.3 *Use Case Scenario* Dosen

Use case scenario pada Tabel 4.15 menjelaskan alur yang terjadi pada *use case* membuat materi singkat. Dalam *use case scenario* membuat materi singkat, aktor yang dilibatkan adalah dosen. Hasil yang akan didapatkan setelah menjalani alur adalah sistem memberikan notifikasi bahwa konten materi singkat telah dibuat. Alur alternatif yang terjadi ialah sistem memberikan notifikasi jika terdapat *error*.

Tabel 4.15 *Use Case Scenario* Membuat Materi Singkat

Nama Use Case	Membuat materi singkat
Aktor	Dosen
Deskripsi	Sistem menyediakan fitur untuk membuat konten materi singkat
Pra-Kondisi	Pengguna telah *login* sebagai dosen
Alur	1. Aktor memilih menu membuat konten materi singkat 2. Aktor mengisi *form* 3. Aktor menekan tombol ubah 4. *Database* materi singkat di *update* oleh sistem
Post-Kondisi	Sistem memberikan notifikasi bahwa konten telah dibuat
Alternatif	4a. Jika *error*, sistem akan menampilkan notifikasi

Use case scenario pada Tabel 4.16 menjelaskan alur yang terjadi pada *use case* mengubah materi singkat. Dalam *use case scenario* mengubah materi singkat, aktor yang dilibatkan adalah dosen. Hasil yang akan didapatkan setelah menjalani

alur adalah sistem mengubah materi singkat yang diubah oleh aktor. Alur alternatif yang terjadi ialah sistem memberikan notifikasi jika terdapat *error*.

Tabel 4.16 *Use Case Scenario* Mengubah Materi Singkat

Nama Use Case	Mengubah Materi Singkat
Aktor	Dosen
Deskripsi	Sistem menyediakan fitur untuk mengubah konten materi singkat
Pra-Kondisi	Pengguna melihat salah satu konten materi singkat
Alur	1. Aktor memilih menu ubah 2. Dosen mengisi *form* 3. Dosen menekan tombol ubah 4. *Database* materi diubah
Post-Kondisi	Materi diubah
Alternatif	4a. Sistem akan mengirimkan notifikasi jika terdapat *error*

Use case scenario pada Tabel 4.17 menjelaskan alur yang terjadi pada *use case* menghapus materi singkat. Dalam *use case scenario* hapus materi, aktor yang dilibatkan adalah dosen. Hasil yang akan didapatkan setelah menjalani alur adalah sistem menghapus konten materi singkat. Tidak terdapat alur alternatif pada *use case* ubah materi.

Tabel 4.17 *Use Case Scenario* Menghapus Materi Singkat

Nama Use Case	Menghapus materi singkat
Aktor	Dosen
Deskripsi	Sistem menyediakan fitur untuk menghapus konten materi
Pra-Kondisi	Pengguna melihat salah satu konten materi
Alur	1. Aktor memilih menu hapus 2. Dosen menekan tombol ya 3. *Database* dihapus
Post-Kondisi	Konten materi dihapus
Alternatif	-

Use case scenario pada Tabel 4.18 menjelaskan alur yang terjadi pada *use case* membuat contoh *source code*. Dalam *use case scenario* membuat contoh *source code*, aktor yang dilibatkan adalah dosen. Hasil yang akan didapatkan setelah menjalani alur adalah sistem menampilkan notifikasi bahwa konten telah dibuat, dan alur alternatif yang terjadi ialah sistem mengirimkan notifikasi bahwa terdapat *error* ketika sistem melakukan *update* pada *database*.

Tabel 4.18 *Use Case Scenario* Membuat Contoh *Source Code*

Nama Use Case	Membuat contoh *source code*
Aktor	Dosen
Deskripsi	Sistem menyediakan fitur untuk membuat konten contoh *source code*
Pra-Kondisi	Pengguna telah *login* sebagai dosen
Alur	1. Aktor memilih menu membuat contoh *source code* 2. Aktor mengisi *form* 3. Aktor menekan tombol ubah 4. *Database* contoh *source code* diubah
Post-Kondisi	Sistem mengirimkan notifikasi bahwa konten telah dibuat
Alternatif	4a. Sistem akan mengirimkan notifikasi bahwa terdapat *error.*

Use case scenario pada Tabel 4.19 menjelaskan alur yang terjadi pada *use case* mengubah contoh *source code*. Dalam *use case scenario* mengubah contoh *source code*, aktor yang dilibatkan adalah dosen. Hasil yang akan didapatkan setelah menjalani alur adalah sistem mengubah konten contoh *source code*, dan tidak alur alternative yang terjadi ialah sistem mengirimkan notifikasi *error* jika sistem gagal melakukan *update* pada *database*.

Tabel 4.19 *Use Case Scenario* Mengubah Contoh *Source Code*

Nama Use Case	Mengubah contoh *source code*
Aktor	Dosen
Deskripsi	Sistem menyediakan fitur untuk mengubah konten contoh *source code*
Pra-Kondisi	Pengguna melihat salah satu konten contoh *source code*
Alur	1. Aktor memilih menu ubah 2. Dosen mengisi *form* 3. Dosen menekan tombol ubah 4. *Database* contoh *source code* diubah
Post-Kondisi	Sistem menampilkan notifikasi bahwa konten telah diubah
Alternatif	4a. Sistem mengirimkan notifikasi *error* jika sistem gagal melakukan *update* pada *database*.

Use case scenario pada Tabel 4.20 menjelaskan alur yang terjadi pada *use case* menghapus contoh *source code*. Dalam *use case scenario* menghapus contoh *source code*, aktor yang dilibatkan adalah dosen. Hasil yang akan didapatkan setelah menjalani alur adalah sistem menghapus konten contoh *source code*. Tidak terdapat alur alternatif pada *use case* hapus contoh *source code*.

Tabel 4.20 *Use Case Scenario* Menghapus Contoh *Source Code*

Nama Use Case	Menghapus contoh *source code*
Aktor	Dosen
Deskripsi	Sistem menyediakan fitur untuk menghapus konten contoh *source code*
Pra-Kondisi	Pengguna melihat salah satu konten contoh *source code*
Alur	1. Aktor memilih menu hapus 2. Dosen menekan tombol ya 3. *Database* dihapus
Post-Kondisi	Konten contoh *source code* dihapus
Alternatif	-

Use case scenario pada Tabel 4.21 menjelaskan alur yang terjadi pada *use case* membuat latihan. Dalam *use case scenario* membuat latihan, aktor yang dilibatkan adalah dosen. Hasil yang akan didapatkan setelah menjalani alur adalah sistem mengirimkan notifikasi bahwa konten latihan telah dibuat. Alur alternative yang terjadi ialah sistem mengirimkan notifikasi *error* jika gagal melakukan *update* pada *database*.

Tabel 4.21 *Use Case Scenario* Membuat Latihan

Nama Use Case	Membuat latihan
Aktor	Dosen
Deskripsi	Sistem menyediakan fitur untuk membuat latihan
Pra-Kondisi	Pengguna telah *login* sebagai dosen
Alur	1. Aktor memilih menu buat latihan 2. Aktor mengisi *form* 3. Aktor menekan tombol ubah 4. *Database* latihan diubah
Post-Kondisi	Konten latihan diubah
Alternatif	4a. Sistem mengirimkan notifikasi *error* jika *database* gagal melakukan *update*.

Use case scenario pada Tabel 4.22 menjelaskan alur yang terjadi pada *use case* mengubah latihan. Dalam *use case scenario* mengubah latihan, aktor yang dilibatkan adalah dosen. Hasil yang akan didapatkan setelah menjalani alur adalah sistem mengubah isi latihan yang terdapat pada aplikasi. Sedangkan alur alternatif yang terjadi ialah sistem mengirimkan notifikasi *error* jika *database* gagal melakukan *update*.

Tabel 4.22 *Use Case Scenario* Mengubah Latihan

Nama Use Case	Mengubah latihan
Aktor	Dosen
Deskripsi	Sistem menyediakan fitur untuk mengubah latihan
Pra-Kondisi	Pengguna melihat salah satu konten latihan
Alur	1. Aktor memilih menu ubah 2. Dosen mengisi *form* 3. Dosen menekan tombol ubah 4. *Database* latihan diubah
Post-Kondisi	Konten latihan diubah
Alternatif	4a. Sistem mengirimkan notifikasi *error* jika sistem gagal Melakukan *update* pada *database*.

Use case scenario pada Tabel 4.23 menjelaskan alur yang terjadi pada *use case* menghapus latihan. Dalam *use case scenario* menghapus latihan, aktor yang dilibatkan adalah dosen. Hasil yang akan didapatkan setelah menjalani alur adalah sistem menghapus konten latihan. Tidak terdapat alur alternatif pada *use case* hapus latihan.

Tabel 4.23 *Use Case Scenario* Menghapus Latihan

Nama Use Case	Menghapus Latihan
Aktor	Dosen
Deskripsi	Sistem menyediakan fitur untuk menghapus konten latihan
Pra-Kondisi	Pengguna melihat salah satu konten latihan
Alur	1. Aktor memilih menu ubah 2. Dosen menekan tombol ya 3. *Database* dihapus
Post-Kondisi	Konten latihan dihapus
Alternatif	-

Use case scenario pada Tabel 4.24 menjelaskan alur yang terjadi pada *use case* membuat *tutorial*. Dalam *use case scenario* membuat *tutorial*, aktor yang dilibatkan adalah dosen. Hasil yang akan didapatkan setelah menjalani alur adalah sistem membuat *tutorial* yang terdapat pada aplikasi. Alur alternatif yang terjadi ialah sistem mengirimkan notifikasi *error* jika *database* gagal melakukan *update*.

Tabel 4.24 *Use Case Scenario* Membuat *Tutorial*

Nama Use Case	Membuat *tutorial*
Aktor	Dosen

Deskripsi	Sistem menyediakan fitur untuk membuat konten *tutorial*
Pra-Kondisi	Pengguna telah *login* sebagai dosen.
Alur	1. Aktor memilih menu buat *tutorial* 2. Dosen mengisi *form* 3. Dosen menekan tombol ubah 4. *Database tutorial* diubah
Post-Kondisi	Konten *tutorial* diubah
Alternatif	4a. Sistem mengirimkan notifikasi *error* jika sistem gagal melakukan *update*.

Use case scenario pada Tabel 4.25 menjelaskan alur yang terjadi pada *use case* mengubah *tutorial*. Dalam *use case scenario* mengubah *tutorial*, aktor yang dilibatkan adalah dosen. Hasil yang akan didapatkan setelah menjalani alur adalah sistem mengubah *tutorial* yang terdapat pada aplikasi. Alur alternatif yang terjadi adalah sistem mengirimkan notifikasi *error* jika *update* pada *database* gagal.

Tabel 4.25 *Use Case Scenario* Mengubah *Tutorial*

Nama Use Case	Mengubah *tutorial*
Aktor	Dosen
Deskripsi	Sistem menyediakan fitur untuk mengubah konten *tutorial*
Pra-Kondisi	Pengguna melihat salah satu konten *tutorial*
Alur	1. Aktor memilih menu ubah 2. Dosen mengisi *form* 3. Dosen menekan tombol ubah 4. *Database tutorial* diubah
Post-Kondisi	Konten *tutorial* diubah
Alternatif	4a. Sistem menampilkan notifikasi *error* jika sistem gagal melakukan *update* pada *database*.

Use case scenario pada Tabel 4.26 menjelaskan alur yang terjadi pada *use case* menghapus *tutorial*. Dalam *use case scenario* menghapus *tutorial*, aktor yang dilibatkan adalah dosen. Hasil yang akan didapatkan setelah menjalani alur adalah sistem menghapus konten *tutorial*. Tidak terdapat alur alternatif pada *use case* menghapus *tutorial*.

Tabel 4.26 *Use Case Scenario* Menghapus *Tutorial*

Nama Use Case	Menghapus *tutorial*
Aktor	Dosen
Deskripsi	Sistem menyediakan fitur untuk menghapus konten *tutorial*

Pra-Kondisi	Pengguna melihat salah satu konten latihan
Alur	1. Aktor memilih menu hapus 2. Dosen menekan tombol ya 3. *Database* dihapus
Post-Kondisi	Konten *tutorial* dihapus
Alternatif	-

4.1.9 Pemilihan Lingkungan Pengembangan

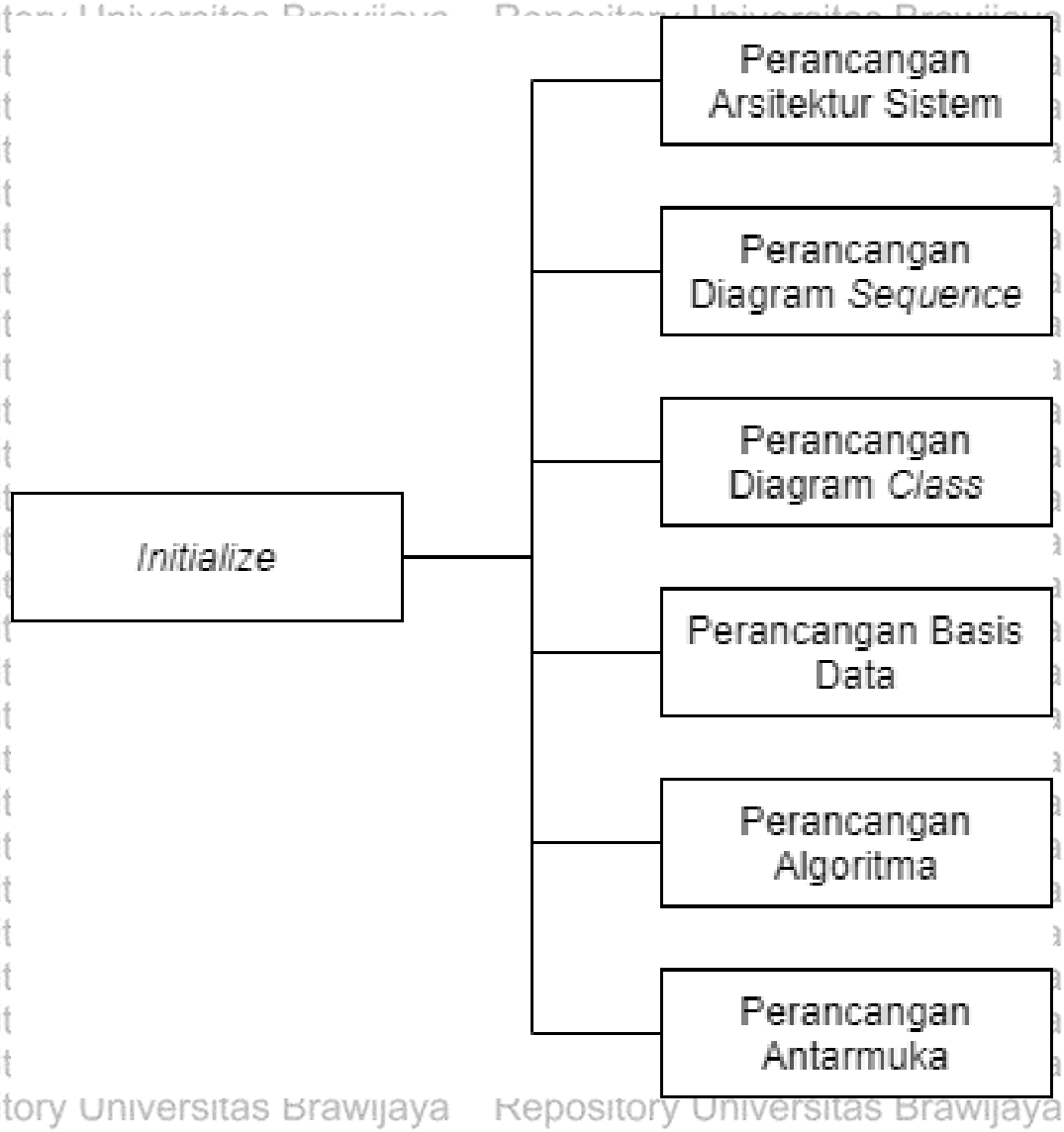
Pengembangan aplikasi *BelajarJava* dilakukan pada OS Android. OS Android dipilih sebagai lingkungan pengembangan dikarenakan dipakai oleh 91% masyarakat di Indonesia pada periode Maret 2018-Maret 2019. Lalu aplikasi dikembangkan pada versi Android 4.4 (*KitKat*) keatas karena dipakai oleh 86% pengguna sesuai dengan *Mobile & Tablet Android Version Market Share* di Indonesia (Statcounter, 2018).

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Bab ini akan membahas fase *initialize*, fase *productionize* dan fase *stabilize* yang terdapat pada metode *Mobile-D* terhadap pengembangan aplikasi *BelajarJava*.

5.1 Perancangan Sistem (*Initialize*)

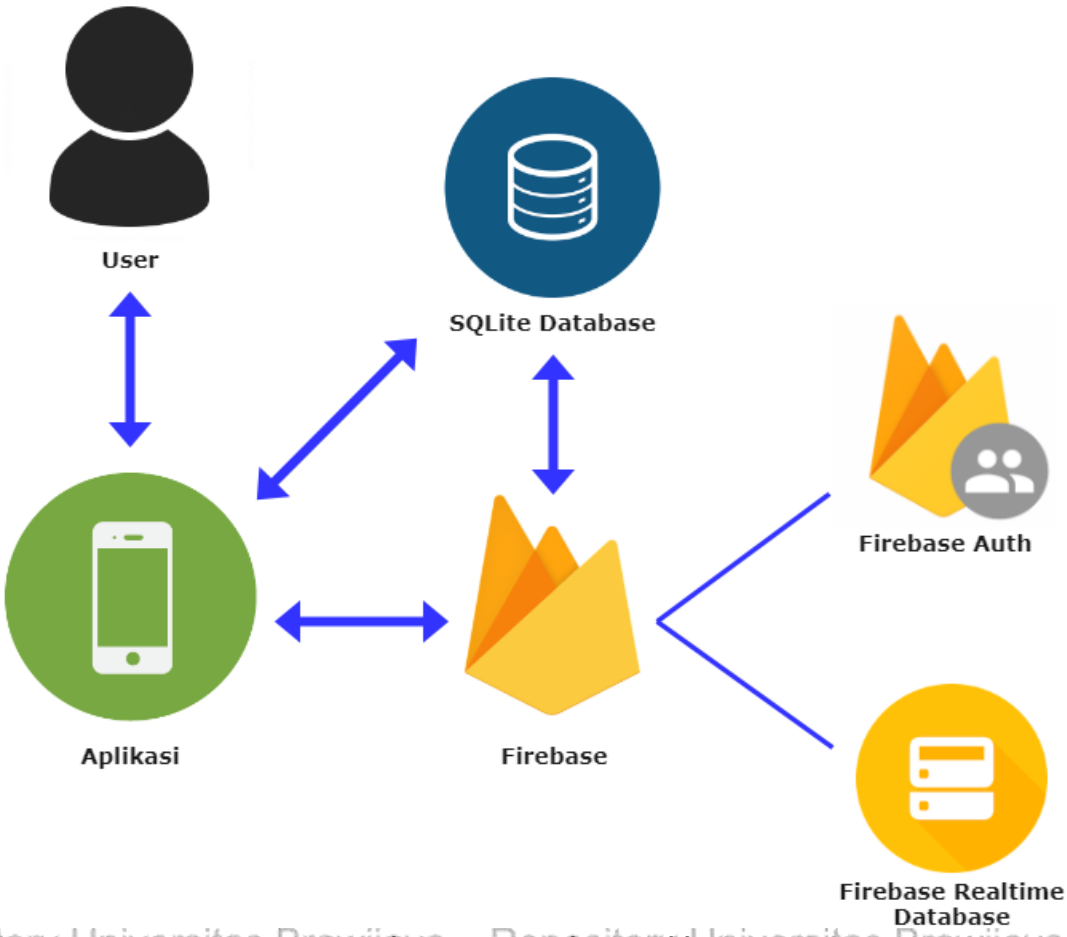
Setelah kebutuhan dianalisis, maka dibuatlah perancangan pada sistem. Pada subbab ini akan membahas perancangan sistem yang mengadopsi fase *initialize* pada tahapan metode *mobile-d*. Tahapan perancangan sistem yang akan dijelaskan pada subbab ini ditampilkan pada Gambar 5.1.



Gambar 5.1 Diagram Pohon Perancangan Sistem

5.1.1 Perancangan Arsitektur Sistem

Sistem aplikasi *BelajarJava* berjalan pada perangkat *mobile* dengan *Operating System* (OS) Android. Cara kerja perangkat lunak digambarkan pada Gambar 5.2.



Gambar 5.2 Arsitektur Aplikasi

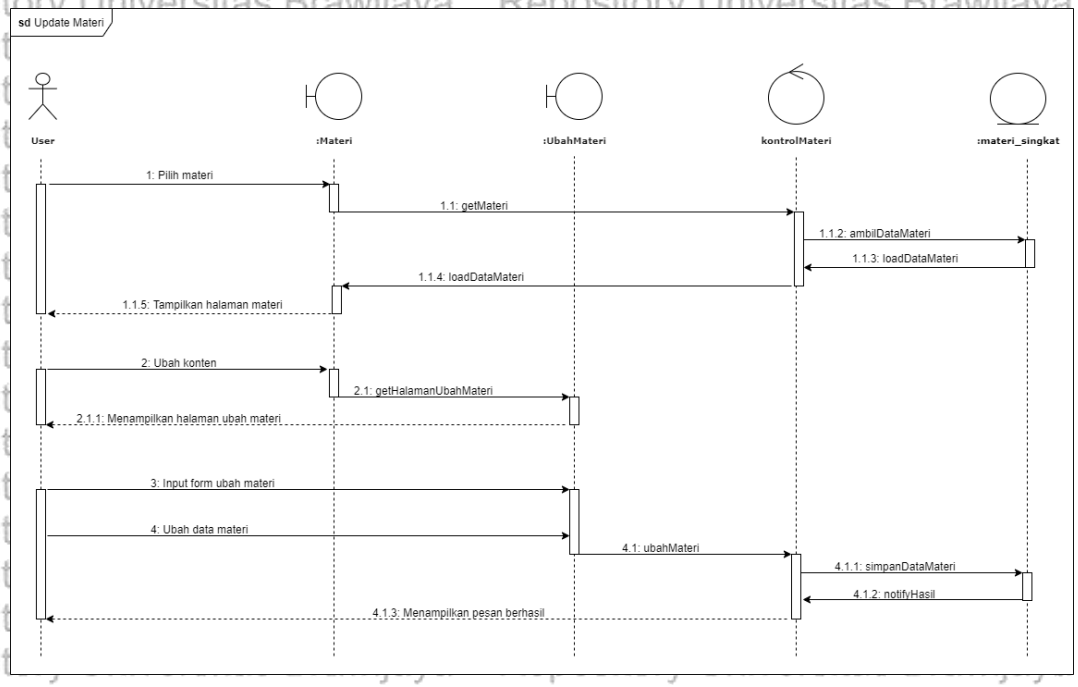
Sebelum menggunakan aplikasi, maka pengguna harus terautentikasi terlebih dahulu dengan *Firebase*. Proses Autentikasi menggunakan *e-mail* untuk login mahasiswa atau *username* untuk dosen dan *password* yang telah terdaftar. Saat melakukan proses *login*, selain autentikasi pengguna, aplikasi akan menyinkronkan data yang terdapat pada data *cloud Firebase* dengan database pada SQLite yang disimpan pada data lokal. Data lokal pada SQLite juga akan disinkronkan dengan data *cloud Firebase* jika aktor mahasiswa melihat menu evaluasi pada sistem.

5.1.2 Perancangan Sequence Diagram

Pada subbab ini akan dijelaskan alur interaksi antara suatu objek dengan objek yang lainnya dalam perangkat lunak. Diagram *sequence* digunakan untuk menunjukkan alur pertukaran serangkaian pesan antar objek. Akan dijelaskan alur-alur pada aplikasi *BelajarJava* yang dianggap penting, yakni ubah konten materi, ubah konten contoh *source code*, dan latihan.

5.1.2.1 *Sequence Diagram Update* Materi

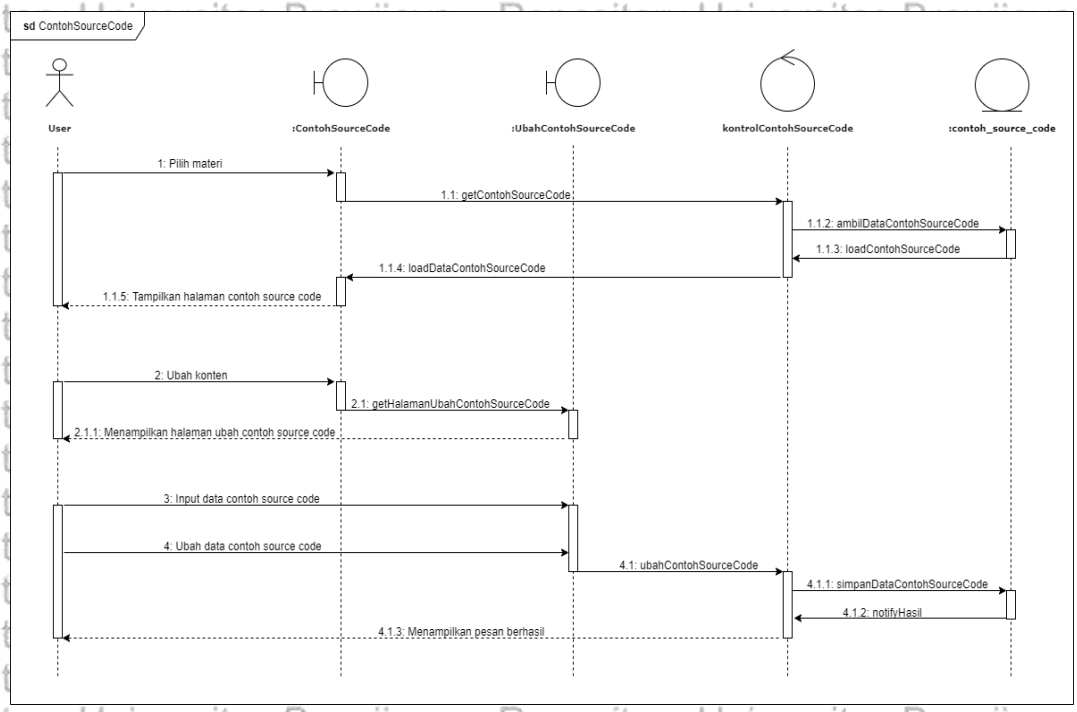
Pada *sequence diagram* ubah materi menjelaskan alur pada sistem *BelajarJava* saat user mengubah konten materi. *Sequence diagram* ubah materi terdapat pada gambar 5.3.



Gambar 5.3 *Sequence Diagram Update* Materi

5.1.2.2 *Sequence Diagram Update Contoh Source Code*

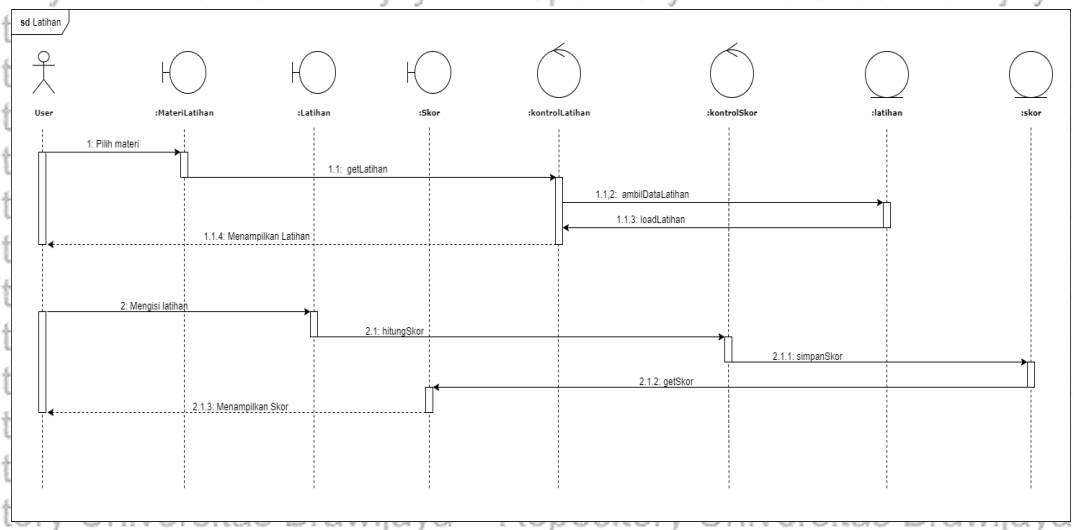
Pada *sequence diagram* ubah contoh *source code* menjelaskan alur pada sistem BelajarJava saat user mengubah konten contoh *source code*. *Sequence diagram update* contoh *source code* terdapat pada Gambar 5.4.



Gambar 5.4 *Sequence Diagram Update* Contoh *Source Code*

5.1.2.3 *Sequence Diagram Latihan*

Pada *sequence diagram* latihan menjelaskan alur pada sistem *BelajarJava* saat *user* saat latihan pemrograman Java. *Sequence diagram* latihan terdapat pada Gambar 5.5.

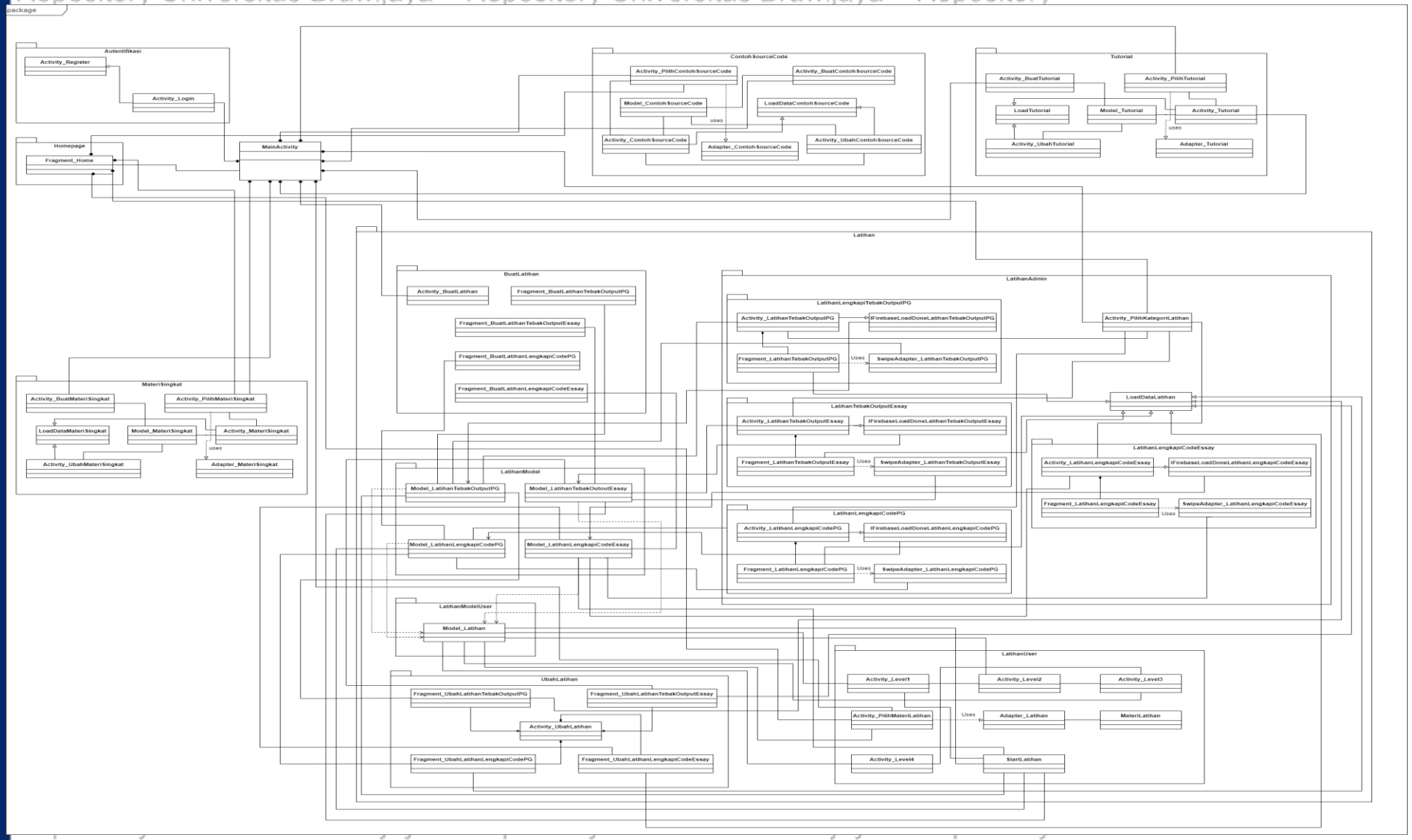


Gambar 5.5 *Sequence Diagram* Ubah Latihan

5.1.3 Perancangan *Class Diagram*

Pada bagian ini akan dijelaskan melalui diagram *class*. Pada sistem ini, kelas di kelompokkan berdasarkan fungsinya yang akan diimplementasi. Pada penelitian ini, akan dijelaskan *class diagram* hasil iterasi 0 dan iterasi 1

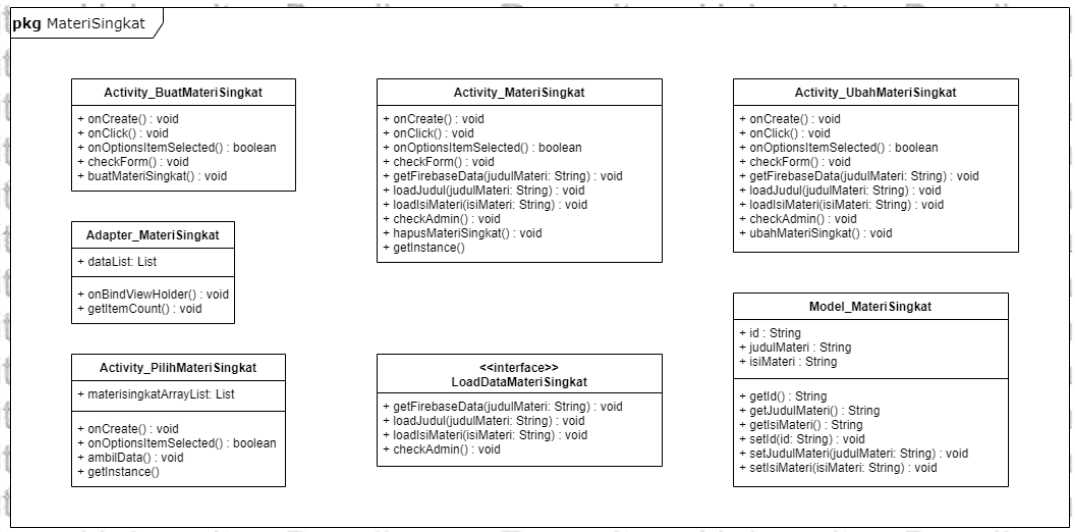
5.1.3.1 *Class Diagram*



Gambar 5.6 *Class Diagram* Aplikasi *BelajarJava*

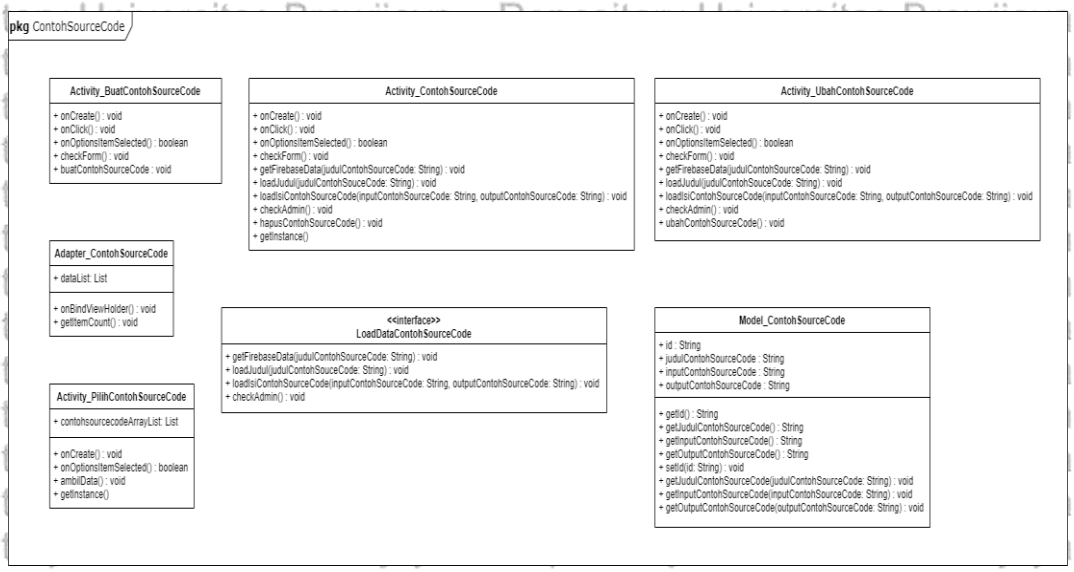
Pada Gambar 5.6 ditunjukkan *class diagram* aplikasi *BelajarJava* secara keseluruhan. Dari Gambar 5.6, tiga *package* utama diambil untuk dijelaskan atribut dan *method* pada setiap kelas. Package utama yang akan dijelaskan adalah paket `MateriSingkat`, paket `ContohSourceCode`, dan paket `LatihanUser`.

Paket `MateriSingkat` merupakan *package* yang berhubungan dengan *use case* pilih materi singkat, lihat materi singkat, ubah materi singkat, dan hapus materi singkat. Paket `MateriSingkat` beserta *class*, atribut dan *method*-nya terdapat pada Gambar 5.7.



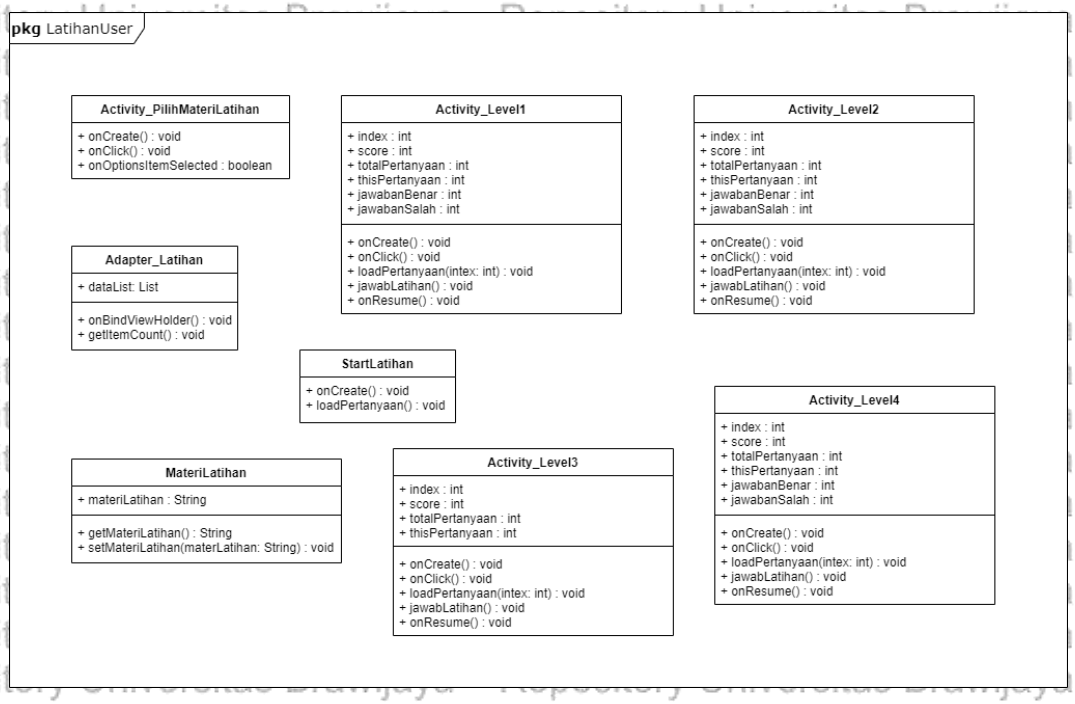
Gambar 5.7 *Class Diagram* Materi Singkat

Paket `ContohSourceCode` merupakan *package* yang menjelaskan kelas yang berhubungan dengan *use case* pilih materi contoh *source code*, lihat contoh *source code*, ubah contoh *source code*, dan hapus contoh *source code*. Paket `ContohSourceCode` beserta *class*, atribut dan *method*-nya terdapat pada Gambar 5.8.



Gambar 5.8 *Class Diagram* Contoh *Source Code*

Paket `LatihanUser` merupakan *package* yang menjelaskan kelas yang berhubungan dengan *use case* latihan. Paket `LatihanUser` beserta *class*, atribut dan *method*-nya terdapat pada Gambar 5.9.



Gambar 5.9 *Class Diagram* Latihan

5.1.4 Perancangan Basis Data

Perancangan basis data pada aplikasi *BelajarJava* menggunakan *Firebase* sebagai *database* yang berbasis noSQL. Karena *database Firebase* bersifat noSQL, maka tidak terdapat relasi antar diagram. Skena *database* pada aplikasi

BelajarJava ditampilkan dalam bentuk JSON sebagaimana ditunjukkan pada gambar 5.10 hingga gambar 5.14 dan dijelaskan pada tabel 5.1 hingga 5.5.

1. Materi Singkat

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "materi_singkat",
    "type": "object",
    "properties": {
        "id": {
            "type": "object",
            "properties": {
                "id": {
                    "type": "string"
                },
                "judulMateri": {
                    "type": "string"
                },
                "isiMateri": {
                    "type": "string"
                },
                "imageUrl": {
                    "type": "string"
                }
            }
        }
    }
}
```

Gambar 5.10 Database *schema* Materi Singkat

Nama tabel : materi_singkat
Jumlah field : 5
Fungsi : untuk menyimpan data materi singkat

Tabel 5.1 Struktur tabel Materi Singkat

No	Nama *Field*	Tipe	Deskripsi
1	id	document	Id materi singkat
2	id	string	Berisi data id materi singkat
3	judulMateri	String	Berisi data judul materi singkat
4	isiMateri	string	Berisi data isi materi singkat
5	imageUrl	string	Berisi data gambar materi singkat

2. Contoh Source Code

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "contoh_source_code",
    "type": "object",
    "properties": {
        "id": {
            "type": "object",
            "properties": {
                "id": {
                    "type": "string"
                },
                "judulContohSourceCode": {
                    "type": "string"
                },
                "inputContohSourceCode": {
                    "type": "string"
                },
                "outputContohSourceCode": {
                    "type": "string"
                },
                "imageUrl": {
                    "type": "string"
                }
            }
        }
    }
}
```

Gambar 5.11 Database *schema* Contoh Source Code

Nama tabel : contoh_source_code
Jumlah field : 6
Fungsi : untuk menyimpan data contoh source code

Tabel 5.2 Struktur tabel Contoh Source Code

No	Nama *Field*	Tipe	Deskripsi
1	id	document	Id contoh source code
2	id	string	Berisi data id contoh source code
3	judulContohSourceCode	String	Berisi data judul contoh source code
4	inputContohSourceCode	string	Berisi data isi input dari contoh source code
5	outputContohSourceCode	string	Berisi data isi input dari contoh source code
6	imageUrl	string	Berisi data gambar contoh source code

3. Latihan Pilihan Ganda

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "latihan",
    "type": "object",
    "properties": {
        "TebakOutput_PG": {
            "type": "object",
            "properties": {
                "id": {
                    "type": "string"
                },
                "materiLatihan": {
                    "type": "string"
                },
                "kategoriLatihan": {
                    "type": "string"
                },
                "gambarLatihan": {
                    "type": "string"
                },
                "pertanyaan": {
                    "type": "string"
                },
                "sourceCode": {
                    "type": "string"
                },
                "kunciJawaban": {
                    "type": "string"
                },
                "jawabanA": {
                    "type": "string"
                },
                "jawabanB": {
                    "type": "string"
                },
                "jawabanC": {
                    "type": "string"
                },
                "jawabanD": {
                    "type": "string"
                }
            }
        },
```

Gambar 5.12 Database *schema* Latihan Pilihan Ganda

Nama tabel : TebakOutput_PG
Jumlah field : 11
Fungsi : menyimpan data latihan tebak output bentuk pilihan ganda

Tabel 5.3 Struktur tabel Latihan Input Output bentuk Pilihan Ganda

No	Nama *Field*	Tipe	Deskripsi
1	id	string	Data id latihan tebak output bentuk pilihan ganda
2	materiLatihan	string	Data materi latihan
3	kategoriLatihan	string	Data kategori latihan
4	gambarLatihan	String	Data url gambar latihan
5	pertanyaan	string	Data pertanyaan dari latihan
6	sourceCode	string	Data isi source code latihan
7	kunciJawaban	string	Data kunci jawaban latihan
8	jawabanA	string	Data jawaban pilihan ganda A
9	jawabanB	string	Data jawaban pilihan ganda B
10	jawabanC	string	Data jawaban pilihan ganda C

11	jawabanD	string	Data jawaban pilihan ganda D

5. Latihan Essay

```
"TebakOutput_Essay": {
    "type": "object",
    "properties": {
        "id": {
            "type": "string"
        },
        "materiLatihan  ": {
            "type": "string"
        },
        "kategoriLatihan": {
            "type": "string"
        },
        "pertanyaan": {
            "type": "string"
        },
        "sourceCode": {
            "type": "string"
        },
        "jawaban": {
            "type": "string"
        }
    }
},
```

Gambar 5.13 Database *schema* Latihan Essay

Nama tabel : TebakOutput_Essay
Jumlah field : 5
Fungsi : untuk menyimpan data latihan tebak output bentuk essay

Tabel 5.4 Struktur tabel Latihan Essay

No	Nama *Field*	Tipe	Deskripsi
1	id	string	Id latihan tebak output bentuk essay
2	materiLatihan	string	Data materi latihan
3	kategoriLatihan	string	Data kategori latihan
4	pertanyaan	string	Data pertanyaan latihan
5	sourceCode	string	Data isi source code latihan
6	jawaban	string	Data jawaban dari latihan

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "materi",
    "type": "object",
    "properties": {
        "id": {
            "id": {
                "type": "string"
            },
            "judulTutorial": {
                "type": "string"
            },
            "isiTutorial": {
                "type": "string"
            }
        }
    }
}
```

Gambar 5.14 Database *schema Tutorial*

Nama tabel : *Tutorial*
Jumlah field : 2
Fungsi : untuk menyimpan data *tutorial*

Tabel 5.5 Struktur Tabel *Tutorial*

No	Nama *Field*	Tipe	Deskripsi
1	id	string	Id latihan lengkapi code bentuk pilihan ganda
2	judulTutorial	string	Data judul *tutorial*
3	isiTutorial	string	Data isi konten *tutorial*

5.1.5 Perancangan *Algoritma*

Pada Subbab ini akan menjelaskan hasil perancangan algoritma dari *method* yang menjadi *core* pada sistem. Perancangan algoritma pada subbab ini merupakan hasil dari tahap *initial requirement analysis* pada tahap *initialize*.

5.1.5.1 Algoritma Buat Materi Singkat

Algoritma buat materi singkat merupakan tahapan proses untuk melakukan fungsi membuat halaman materi pada aplikasi *BelajarJava*. Implementasi algoritma lihat materi singkat dijelaskan pada Gambar 5.15.

```
Algoritma buatMateriSingkat
Deklarasi :

 - String -> id
 - Hashmap -> judulMateri, isiMateri, gambarMateri

Deskripsi :

 - Masukkan : judulMateri, isiMateri, gambarMateri
 - Proses :
     1. Deklarasi variabel judulMateri, isiMateri, gambarMateri
```

```
    2. Mendapatkan key dari Firebase dan disimpan dalam variabel id
    3. Inisialisasi method MateriSingkat
    4. Menyimpan data judulMateri, isiMateri, gambarMateri ke dalam
       bentuk model MateriSingkat dan menyimpannya ke database
       Firebase. Jika sukses maka program akan melakukan :
       a. Memanggil method addPhoto
       b. Set nilai imageUrl pada variabel gambarMateri
       c. Menampilkan pesan Toast bahwa materi telah dibuat.
    5. Jika proses gagal maka akan menampilkan pesan Toast bahwa
       proses buat materi gagal.
```

Gambar 5.15 Algoritma Method Buat Materi Singkat

5.1.5.2 Algoritma Jawab Latihan

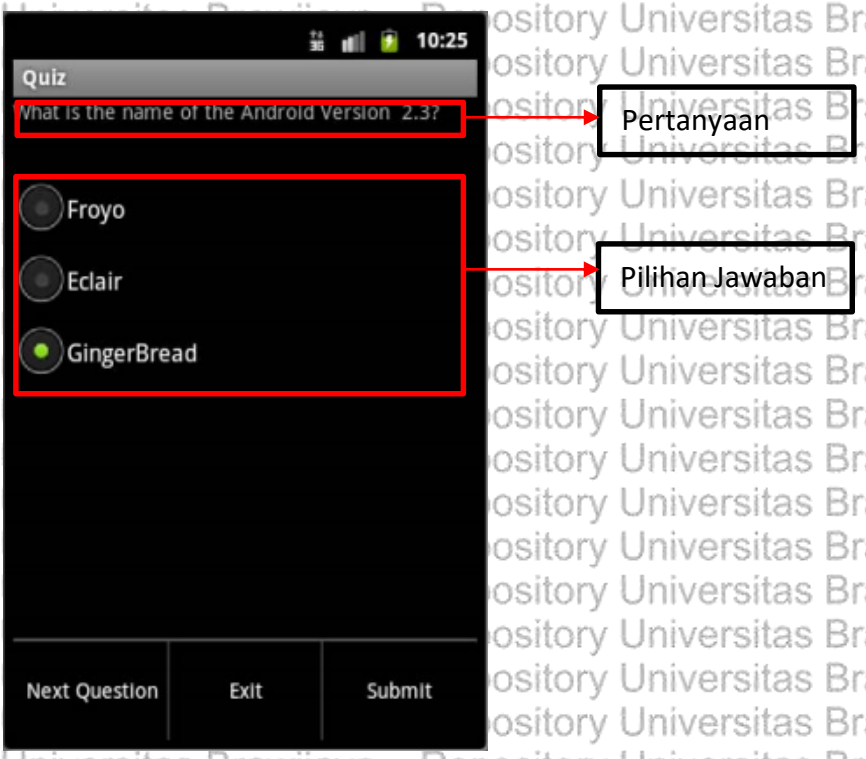
Algoritma jawab latihan merupakan tahapan proses untuk melakukan fungsi menjawab latihan pada aplikasi *BelajarJava*. Implementasi algoritma jawab latihan dijelaskan pada Gambar 5.16.

```
Algoritma jawabLatihan
Deklarasi :
  - Hashmap -> materi, jawabanUser, kunciJawaban, score
Deskripsi :
  - Masukkan : materi, jawabanUser, kunciJawaban, score
  - Proses :
    1. Deklarasi variabel materi, jawabanUser, kunciJawaban, score
    2. Masukkan kondisi if untuk memeriksa materi latihan
    3. Masukkan kondisi if untuk mengecek jawaban user
       a. Jika jawaban benar, maka skor bertambah 1
       b. Jika jawaban salah, maka skor tidak bertambah
```

Gambar 5.16 Algoritma Method Jawab Latihan

5.1.6 Perancangan Antarmuka

Pada Sub bab ini menjelaskan rancangan antarmuka dari sistem aplikasi *BelajarJava*. Pada bagian ini berisi tampilan rancangan antarmuka yang akan digunakan oleh aktor dan juga *screen flow* untuk menggambarkan kebutuhan tiap aktor yang terdiri dari *guest*, mahasiswa, dan dosen. Implementasi antarmuka halaman latihan menurut Shanmugapriya (2011), bagian pertanyaan diletakkan pada bagian atas antarmuka, bagian pilihan jawaban diletakkan di bawah pertanyaan. Contoh implementasi antarmuka halaman latihan pada aplikasi *m-learning* terdapat pada Gambar 5.17.

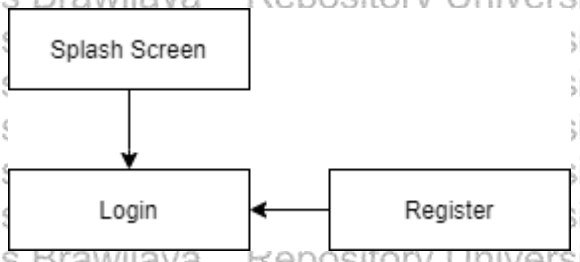


Gambar 5.17 Contoh Implementasi Antarmuka Latihan

(Sumber : Shanmugapriya, 2011)

5.1.6.1 Perancangan Antarmuka pada Aktor *Guest*

Antarmuka pada aktor *guest* merupakan antarmuka yang berfungsi untuk melakukan autentifikasi pada sistem. Alur proses jalannya aplikasi pada aktor *guest* digambarkan pada Gambar 5.18 berikut :



Gambar 5.18 *Screen Flow* Aplikasi pada Aktor *Guest*

Antarmuka pada aktor *guest* sendiri terdiri dari halaman *login*, *register*, dan *reset password*. Aplikasi akan menampilkan antarmuka halaman *splash screen* lalu berpindah ke halaman login. Dari halaman login, aktor dapat menuju halaman *register* ataupun *reset password*.

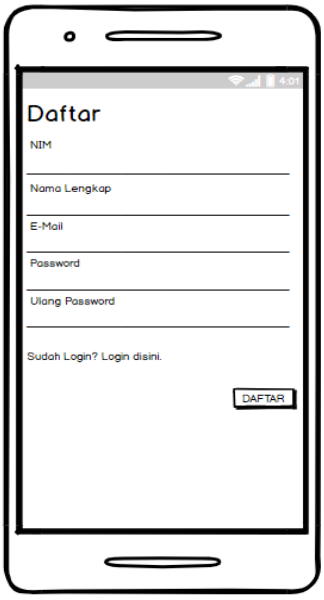
a. Halaman *Login*

Halaman *login* berfungsi sebagai halaman autentifikasi sehingga pengguna dapat menggunakan fitur yang terdapat pada sistem. Perancangan antarmuka halaman *login* terdapat pada Gambar 5.19.

Gambar 5.19 Tampilan Perancangan Antarmuka Halaman *Login*

b. Halaman *Register*

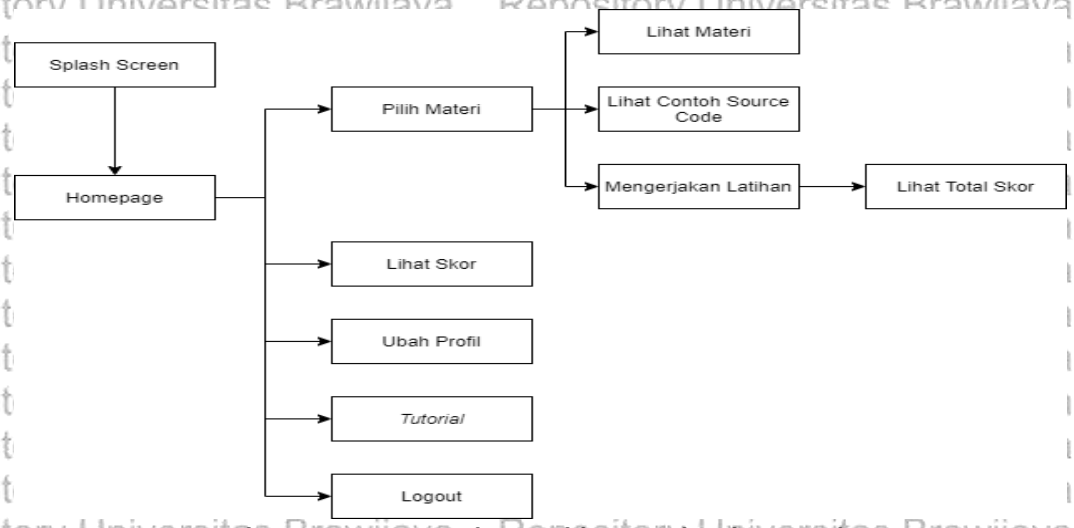
Halaman *register* digunakan untuk mendaftarkan calon pengguna agar dapat menggunakan sistem. Perancangan antarmuka halaman *register* terdapat pada Gambar 5.20.



Gambar 5.20 Tampilan Perancangan Antarmuka Halaman *Register*

5.1.6.2 Perancangan Antarmuka pada Aktor Mahasiswa

Antarmuka pada aktor mahasiswa merupakan halaman yang berfungsi untuk masuk ke dalam sistem. Alur proses jalannya aplikasi pada aktor mahasiswa digambarkan pada Gambar 5.21 berikut :



Gambar 5.21 *Screen Flow* Aplikasi pada Aktor Mahasiswa

Antarmuka pada aktor mahasiswa sendiri terdiri dari halaman materi, contoh *source code*, latihan, pilih materi, *tutorial*, danubah profil. Aplikasi akan menampilkan halaman *splash screen* terlebih dahulu lalu berpindah ke halaman *homepage* yang berisi pilihan menu materi, contoh *source code*, latihan, *tutorial* dan logout. Ketika pengguna memilih menu materi atau latihan, pengguna akan memilih materi-nya terlebih dahulu lalu akan menuju halaman materi, contoh *source code* atau latihan.

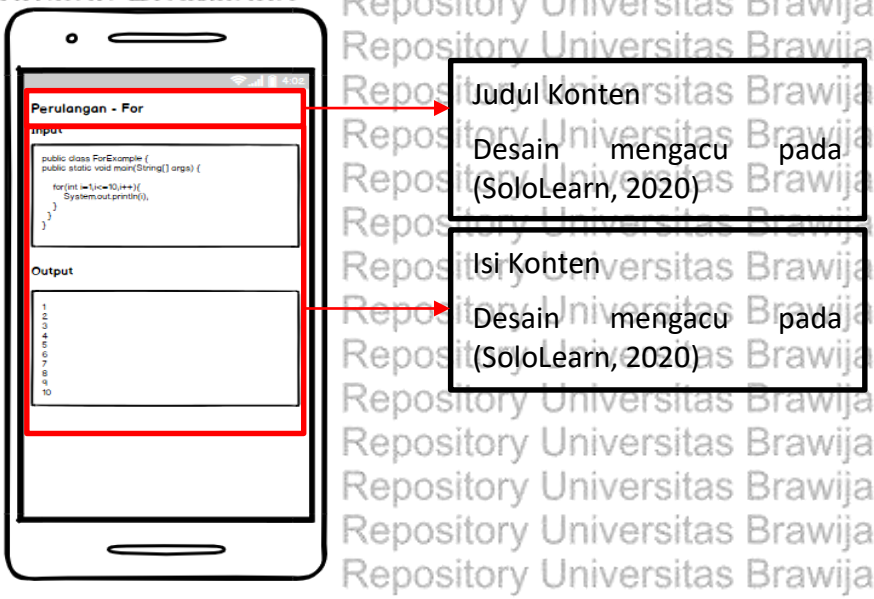
a. Halaman Materi

Halaman materi ialah salah satu antarmuka untuk melihat isi konten materi. Perancangan antarmuka materi terdapat pada Gambar 5.22.

Gambar 5.22 Tampilan Perancangan Antarmuka Halaman Materi Singkat

b. Halaman Contoh *Source Code*

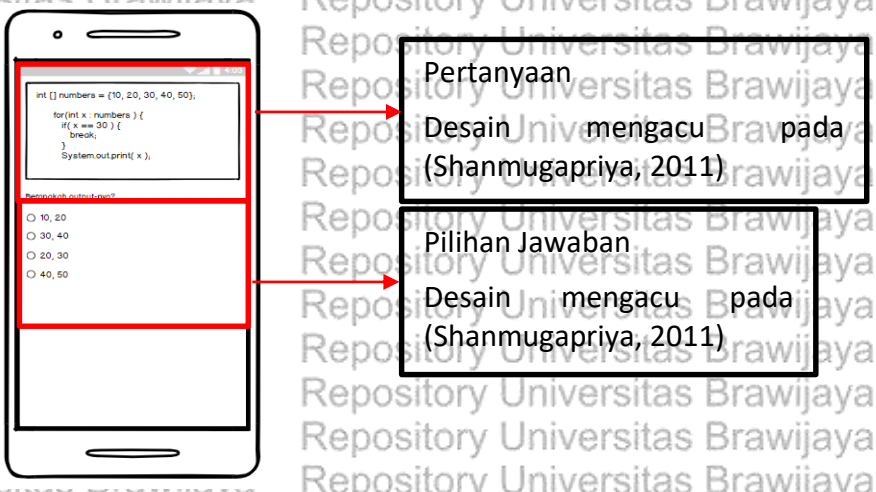
Halaman contoh *source code* digunakan untuk melihat isi konten contoh *source code*. Perancangan antarmuka contoh *source code* terdapat pada Gambar 5.23.



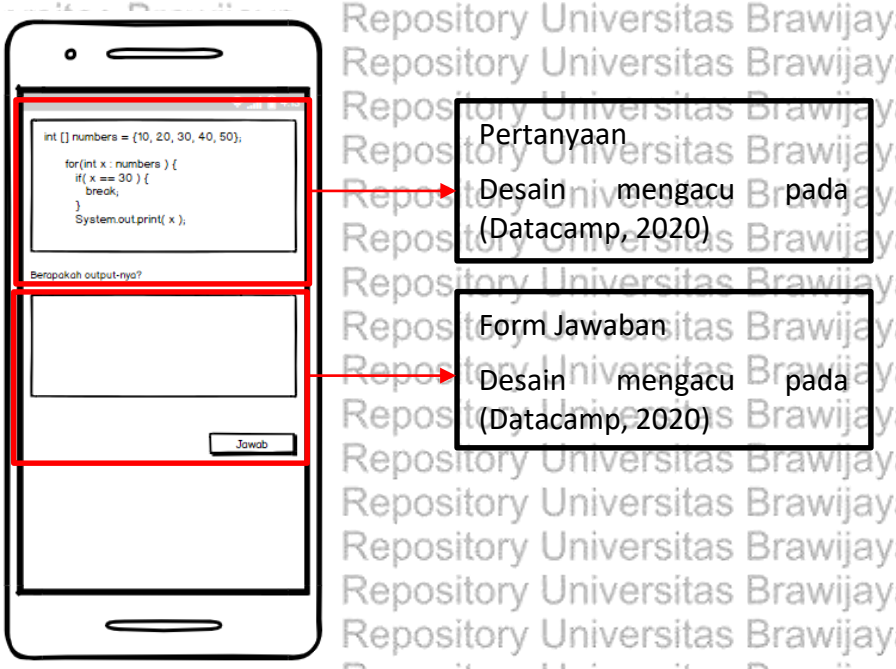
Gambar 5.23 Tampilan Perancangan Antarmuka Halaman Contoh *Source Code*

c. Halaman Latihan

Halaman latihan digunakan untuk mengerjakan isi latihan yang telah dibuat oleh aktor dosen. Perancangan antarmuka latihan terdiri dari empat halaman yang berisi jenis latihan, yakni tebak output bentuk pilihan ganda dan tebak output bentuk essay. Perancangan antarmuka latihan terdapat pada gambar 5.24 sampai 5.25.



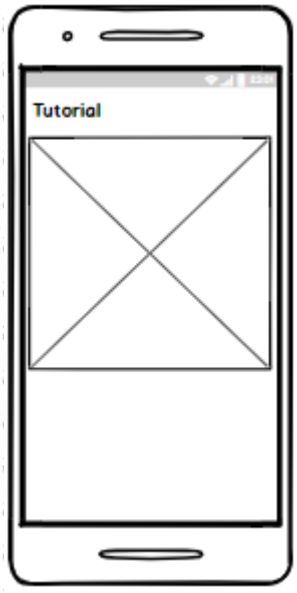
Gambar 5.24 Tampilan Perancangan Antarmuka Halaman Latihan Tebak Output Bentuk Pilihan Ganda



Gambar 5.25 Tampilan Perancangan Antarmuka Halaman Latihan Tebak Output Bentuk Essay

d. Halaman *Tutorial*

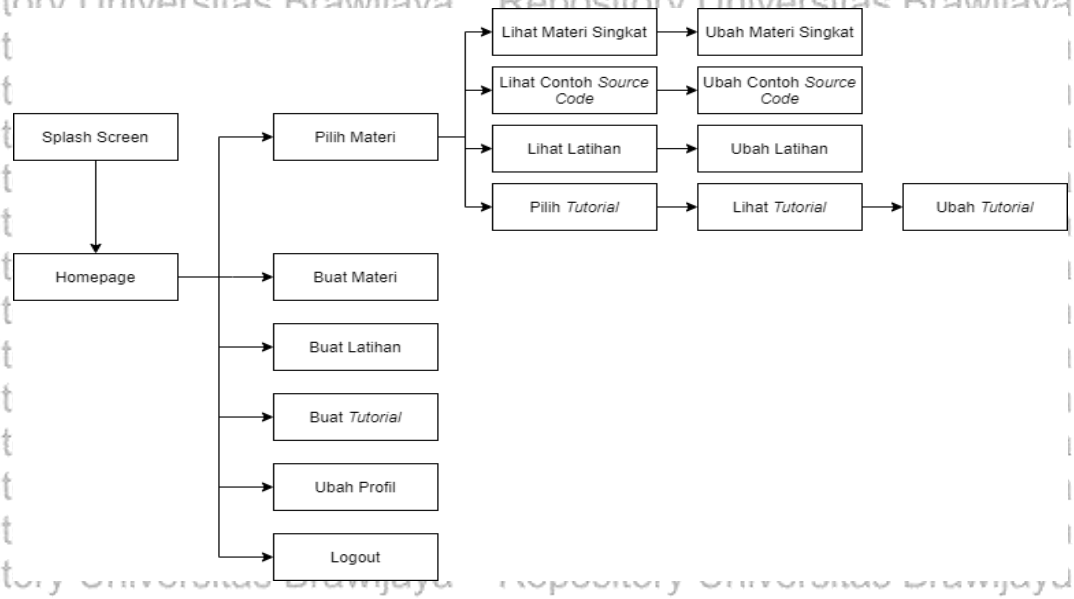
Halaman *tutorial* digunakan untuk melihat konten *tutorial*. Perancangan antarmuka halaman *tutorial* terdapat pada Gambar 5.26.



Gambar 5.26 Tampilan Perancangan Antarmuka Halaman *Tutorial*

5.1.6.3 *Perancangan Antarmuka pada Aktor Dosen*

Antarmuka pada aktor dosen merupakan halaman yang berfungsi untuk masuk ke dalam sistem. Alur proses jalannya aplikasi pada aktor dosen digambarkan pada Gambar 5.27 berikut :



Gambar 5.27 *Screen Flow* Aplikasi pada Aktor Dosen

Antarmuka pada aktor dosen sendiri hamper sama dengan antarmuka pada aktor mahasiswa. Di bagian ini akan dijelaskan perancangan antarmuka yang berkaitan tentang *Create*, *Read*, *Update* (CRUD).

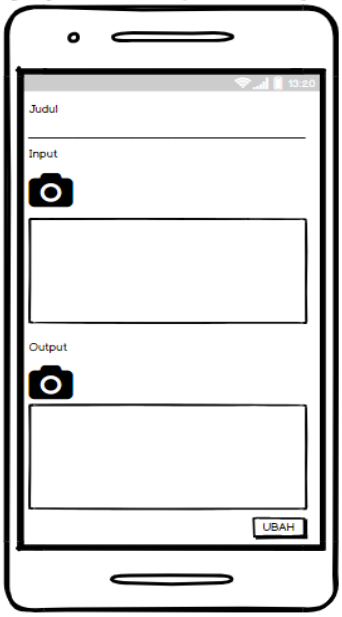
a. Halaman CRUD Materi

Halaman CRUD materi ialah salah satu antarmuka untuk menambah atau mengubah konten materi singkat. Perancangan antarmuka CRUD materi terdapat pada Gambar 5.28.

Gambar 5.28 Tampilan Perancangan Antarmuka CRUD Materi Singkat

b. Halaman CRUD Contoh *Source Code*

Halaman CRUD contoh *source code* ialah salah satu antarmuka untuk menambah atau mengubah konten contoh *source code*. Perancangan antarmuka CRUD contoh *source code* terdapat pada Gambar 5.29.

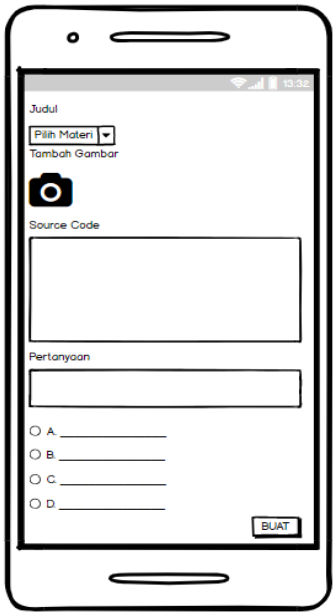


Gambar 5.29 Tampilan Perancangan Antarmuka CRUD Contoh *Source Code*

c. Halaman CRUD Latihan Tebak Output Bentuk Pilihan Ganda

Halaman CRUD latihan tebak output bentuk pilihan ganda ialah salah satu antarmuka untuk menambah atau mengubah konten latihan yang berkategori tebak output bentuk pilihan ganda. Perancangan

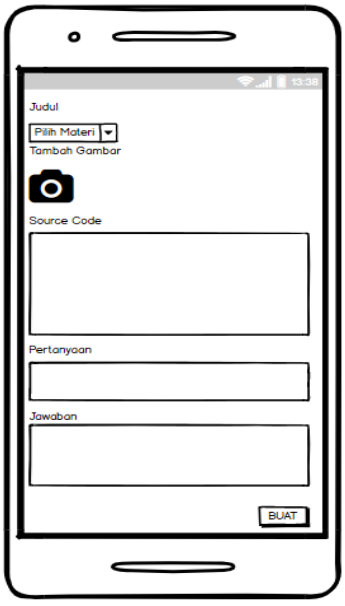
antarmuka CRUD latihan tebak output bentuk pilihan ganda terdapat pada Gambar 5.30.



Gambar 5.30 Tampilan Perancangan Antarmuka CRUD Latihan Tebak Output Bentuk Pilihan Ganda

d. Halaman CRUD Latihan Tebak Output Bentuk Essay

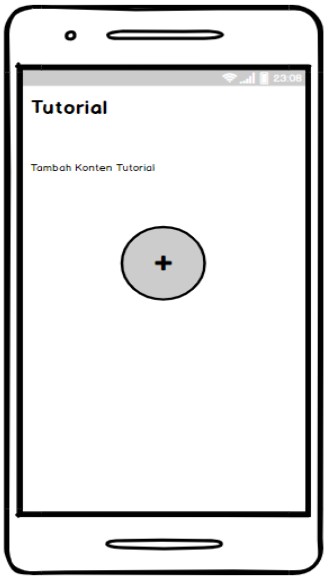
Halaman CRUD latihan tebak output bentuk essay ialah salah satu antarmuka untuk menambah atau mengubah konten latihan yang berkategori tebak output bentuk essay. Perancangan antarmuka CRUD latihan tebak output bentuk essay terdapat pada Gambar 5.31.



Gambar 5.31 Tampilan Perancangan Antarmuka CRUD Latihan Tebak Output Bentuk Essay

e. Halaman CRUD *Tutorial*

Halaman CRUD *Tutorial* ialah salah satu antarmuka untuk menambah atau mengubah konten *tutorial*. Perancangan antarmuka CRUD *tutorial* terdapat pada Gambar 5.32.



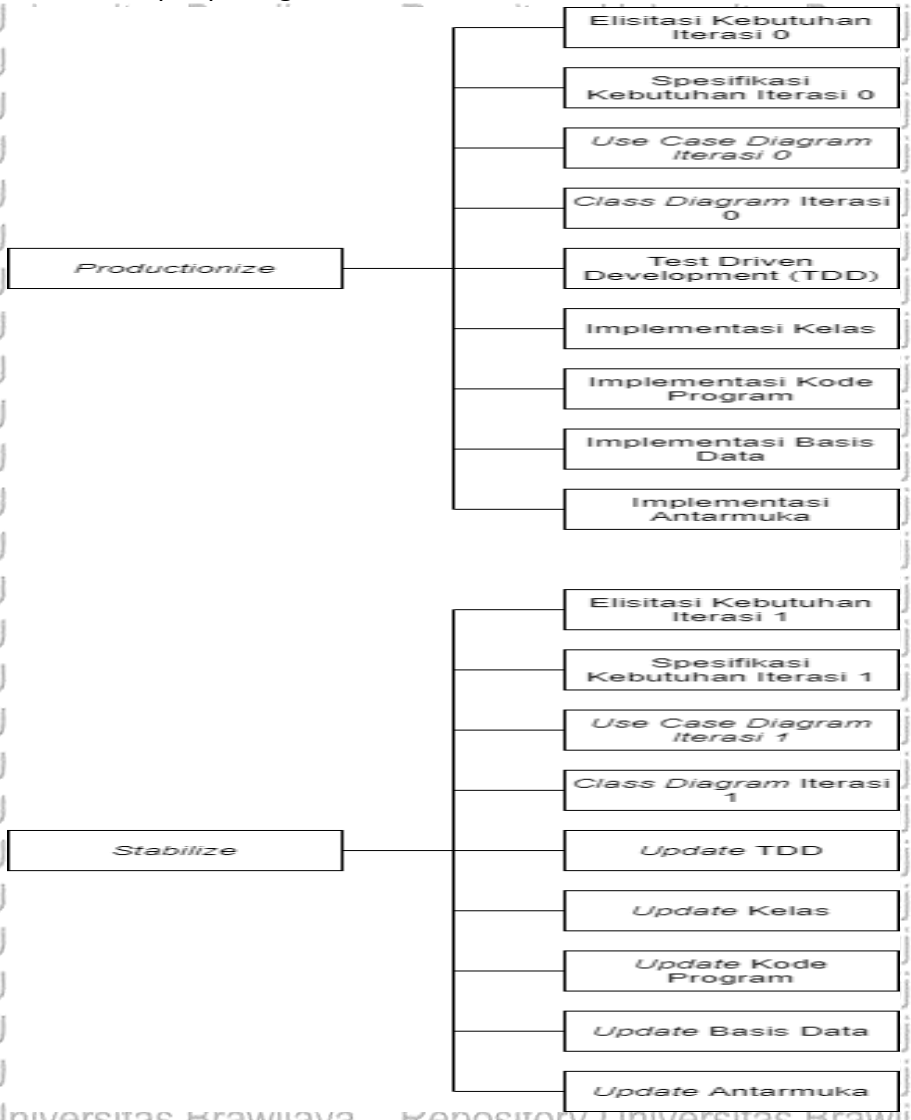
Gambar 5.32 Tampilan Perancangan Antarmuka Halaman CRUD *Tutorial*

5.2 Implementasi (*Productionize* dan *Stabilize*)

Pada Subbab ini akan dijelaskan tahapan dari fase *Productionize* dan *Stabilize*. Pada fase *productionize* terjadi iterasi 0 dimana hasil dari perancangan tahap *initialize* akan diberikan kepada pengguna untuk mengetahui *feedback* awal dari pengguna. Setelah *feedback* diterima, maka akan dilakukan implementasi awal pada aplikasi, kemudian hasil dari implementasi pada fase *productionize* akan diberikan kembali ke pengguna untuk dimintai *feedback* kembali.

Pada fase *stabilize*, hasil *feedback* pengguna terhadap implementasi awal kemudian dijadikan acuan untuk menyempurnakan produk. Setelah produk dirasa telah sesuai dengan kebutuhan pengguna, baru produk akan diujikan di fase *system test & fix*.

Perubahan dan hasil implementasi yang terjadi pada fase *productionize* dan fase *stabilize* terdapat pada gambar 5.33



Gambar 5.33 Diagram Pohon Implementasi

5.2.1 Elisitasi Kebutuhan Iterasi 0

Setelah mendapatkan *feedback* setelah fase *initialize* dilakukan, dilakukan analisis kebutuhan kembali untuk menentukan prioritas kebutuhan. Terdapat perubahan pada prioritas kebutuhan mahasiswa saat dilakukan iterasi 0 dimana kebutuhan sistem untuk menyediakan *tutorial* dihapus. Kebutuhan ini didapatkan dari total 5 responden yang terbagi menjadi 3 responden dari mahasiswa dan 2 responden dari dosen.

Tabel 5.6 adalah hasil dari fase *initial requirement analysis* tahap *productionize*. Pada tahap ini, penulis menentukan prioritas terhadap fungsionalitas yang akan diimplementasikan. Kebutuhan dengan nilai kurang dari 4 dihapus.

Tabel 5.6 Elisitasi Kebutuhan Mahasiswa Iterasi 0

No	Nama Kebutuhan	Prioritas Kebutuhan					Nilai Akhir
		1	2	3	4	5	
1	Sistem disediakan materi singkat						
1.1	Sistem dapat menyediakan materi singkat perulangan *for*						
1.2	Sistem dapat menyediakan materi singkat perulangan *while*	0	0	0	2	3	5
1.3	Sistem dapat menyediakan materi singkat perulangan *do-while*						
2	Sistem disediakan contoh *source code*						
2.1	Sistem dapat menyediakan contoh *source code* perulangan *for*						
2.2	Sistem dapat menyediakan contoh *source code* perulangan *while*	0	0	0	3	2	4
2.3	Sistem dapat menyediakan contoh *source code* perulangan *do-while*						
3	Sistem disediakan latihan						
3.1	Sistem dapat menyediakan latihan perulangan *for*						

3.2	Sistem dapat menyediakan latihan perulangan *while*	0	0	0	4	1	4
3.3	Sistem dapat menyediakan latihan perulangan *do-while*						
4	Sistem terdapat *tutorial*						
4.1	Sistem dapat menyediakan *tutorial* perulangan *for*						
4.2	Sistem dapat menyediakan *tutorial* perulangan *while*	0	2	2	0	1	3
4.3	Sistem dapat menyediakan *tutorial* perulangan *do-while*						

Pada Tabel 5.6 terdapat spesifikasi kebutuhan yang mendapatkan nilai dibawah 4, yakni kebutuhan sistem menyediakan *tutorial*. Oleh karena itu, kebutuhan sistem menyediakan *tutorial* dihapus dalam daftar kebutuhan sistem.

Selain itu, terdapat penambahan kebutuhan berdasarkan *feedback* yang telah didapatkan, yakni disediakannya fitur ubah profil. Kebutuhan ini berguna untuk keperluan identifikasi aktor. Penambahan kebutuhan mahasiswa dan dosen terdapat pada Tabel 5.7.

Tabel 5.7 Elisitasi Kebutuhan Mahasiswa dan Dosen Iterasi 0

No	Nama Kebutuhan	Prioritas Kebutuhan					Nilai Akhir
		1	2	3	4	5	
1	Sistem disediakan fitur ubah profil	0	0	0	2	3	5

5.2.2 Spesifikasi Kebutuhan Iterasi 0

Pada Tabel 5.8 dan Tabel 5.9, terdapat pengurangan fungsi pada iterasi 0. Yaitu fungsi dengan kode BJ_M_004 yaitu fungsi *tutorial* pada aktor mahasiswa, serta fungsi dengan kode BJ_D_009, BJ_D_010, dan BJ_D_011 yaitu fungsi CRUD *tutorial*. Fungsi ini ditambahkan karena *feedback* dari pengguna yang tidak memerlukan fungsi ini. Fungsi *tutorial* dianggap tidak perlu karena pengguna lebih memilih melakukan *tutorial coding* pada perangkat *Personal Computer* (PC) mereka.

Selain pengurangan fungsi, terdapat juga penambahan fungsi yaitu fungsi mengubah profil. Fungsi ini ditambahkan setelah mendapatkan *feedback* dari responden yang tujuan ditambahkannya fungsi ini ialah untuk identifikasi aktor.

Tabel 5.8 Spesifikasi Kebutuhan Fungsional Mahasiswa Iterasi 0

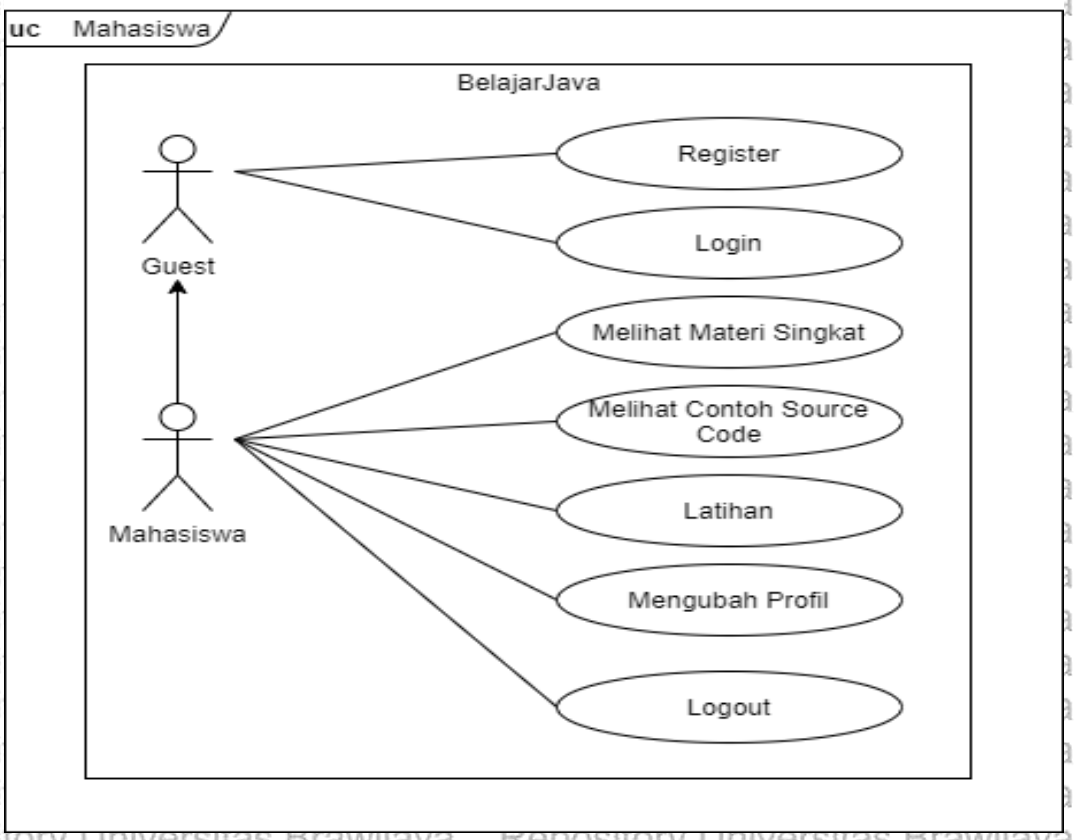
No	Kode Fungsi	Nama Fungsi	Deskripsi	Keterangan
1	BJ_M_004	*Tutorial*	Sistem menyediakan fitur *tutorial* kepada pengguna	Fungsi dihapus
2	BJ_M_005	Mengubah profil	Sistem menyediakan fitur mengubah profil untuk keperluan identifikasi pengguna	Fungsi ditambahkan

Tabel 5.9 Spesifikasi Kebutuhan Fungsional Dosen Iterasi 0

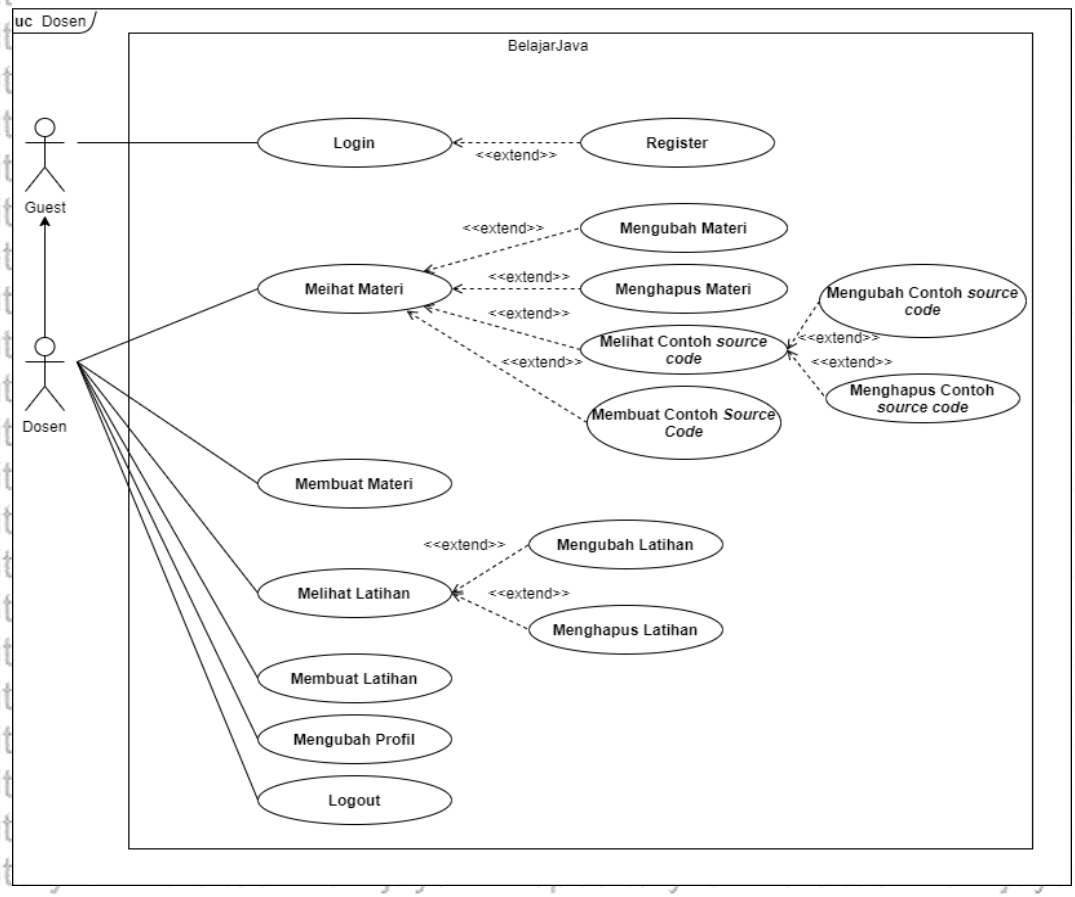
No	Kode Fungsi	Nama Fungsi	Deskripsi	Keterangan
1	BJ_D_009	Buat *tutorial*	Sistem menyediakan fitur membuat konten *tutorial*	Fungsi dihapus
2	BJ_D_010	Ubah *tutorial*	Sistem menyediakan fitur untuk mengubah konten tutorial	Fungsi dihapus
3	BJ_D_011	*Hapus tutorial*	Sistem menyediakan fitur untuk menghapus konten tutorial	Fungsi dihapus
4	BJ_M_005	Mengubah profil	Sistem menyediakan fitur mengubah profil untuk keperluan identifikasi pengguna	Fungsi ditambahkan

5.2.3 *Use Case Diagram* Iterasi 0

Setelah melakukan *update* pada dokumen spesifikasi kebutuhan, langkah selanjutnya ialah melakukan *update* pada *use case diagram*. *Use case diagram* untuk ini adalah hasil dari iterasi 0 yang digambarkan pada Gambar 5.34 dan Gambar 5.35 dimana terjadi penghapusan *use case tutorial* pada *use case* mahasiswa, dan *use case* membuat *tutorial*, *use case* mengubah *tutorial*, dan *use case* menghapus *tutorial* pada *use case* dosen.



Gambar 5.34 *Use Case Diagram* Iterasi 0 dengan Aktor Mahasiswa



Gambar 5.35 *Use Case Diagram* Iterasi 0 dengan Aktor Dosen

5.2.4 *Use Case Scenario* Iterasi 0

Terdapat penambahan *use case* mengubah profil dan penghapusan *use case* melihat dan mengubah *tutorial*.

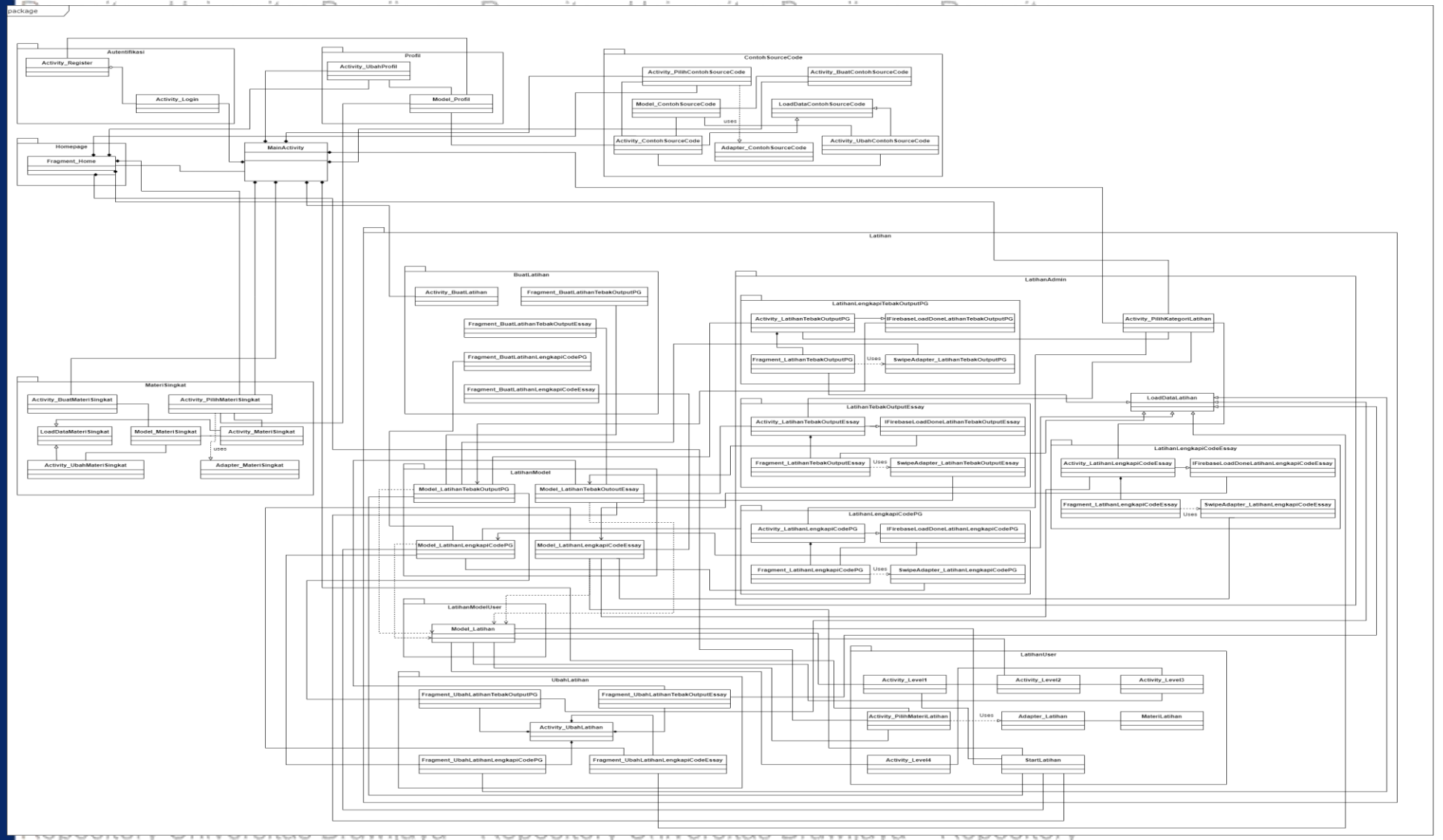
Use case scenario pada Tabel 5.10 menjelaskan alur yang terjadi pada *use case* mengubah profil. Dalam *use case scenario* mengubah profil, aktor yang dilibatkan adalah mahasiswa dan dosen. Hasil yang akan didapatkan setelah menjalani alur adalah notifikasi ditampilkan oleh sistem bahwa data pengguna telah diubah, lalu alur alternatif yang terjadi adalah pesan *error* ditampilkan oleh sistem.

Tabel 5.10 *Use Case Scenario* Iterasi 0 Mengubah Profil

Nama Use Case	Mengubah profil
Aktor	Mahasiswa dan Dosen
Deskripsi	Fungsi bagi pengguna untuk mengubah data pribadi
Pra-Kondisi	Pengguna telah *login* sebagai mahasiswa atau dosen
Alur	1. Aktor memilih menu ubah profil pada tab menu 2. Aktor mengisi *form* data pengguna yang akan diubah 3. Sistem mengubah *database* pengguna

Post-Kondisi	Sistem menampilkan notifikasi data telah diubah
Alternatif	3a. Sistem gagal mengubah *database* pengguna Sistem menampilkan pesan *error* karena terdapat kesalahan seperti kolom *password* lama dan *password* baru tidak sama

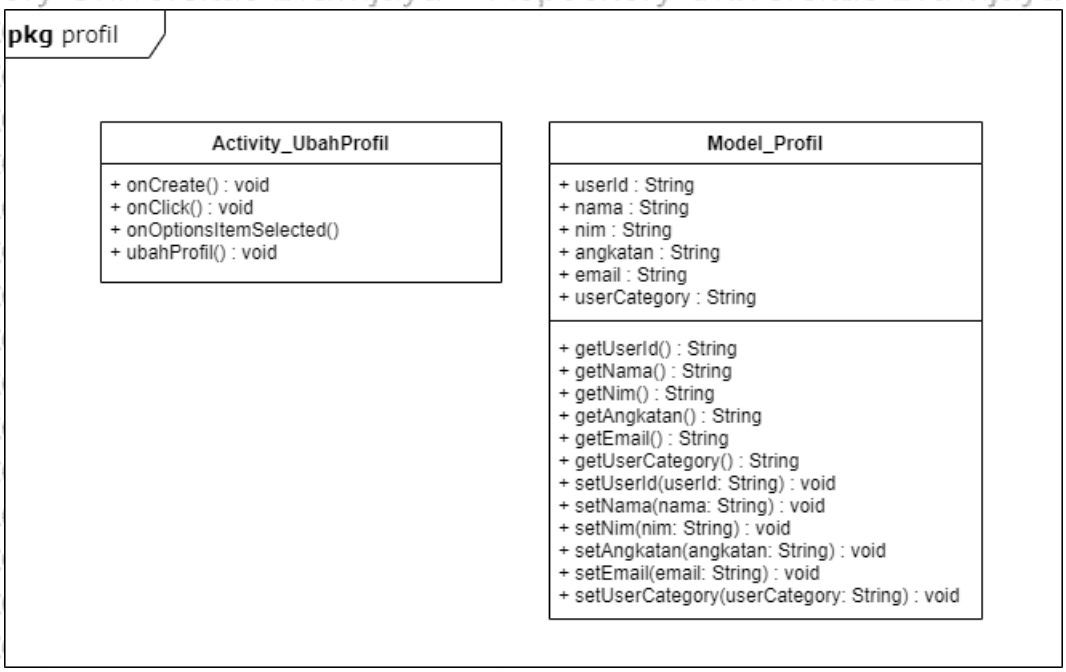
5.2.5 Class Diagram *Iterasi 0*



Gambar 5.36 *Class Diagram* Aplikasi *BelajarJava* Iterasi 0

Dilakukan juga *update* pada *class diagram* setelah terdapat *update* pada daftar dan spesifikasi kebutuhan dimana pada *class diagram* ini paket `Tutorial` dihapus dan paket `Profil` ditambahkan.

Paket `Profil` merupakan *package* yang menjelaskan kelas yang berhubungan dengan *use case* mengubah profil. Paket `Profil` beserta *class*, atribut dan *method*-nya terdapat pada Gambar 5.37.



Gambar 5.37 *Class Diagram* Profil

5.2.6 Spesifikasi Sistem

Setelah perancangan sudah siap, maka langkah selanjutnya yang dilakukan ialah meniapkan sistem yang dibutuhkan untuk mengembangkan perangkat lunak. Pada tahapan ini dijelaskan lingkungan pengembangan aplikasi *BelajarJava*. Aplikasi *BelajarJava* dikembangkan dengan menggunakan perangkat keras dan perangkat lunak.

5.2.6.1 Spesifikasi Perangkat Keras

Aplikasi *BelajarJava* dikembangkan dengan menggunakan perangkat keras dan perangkat lunak. Spesifikasi perangkat keras terdapat pada Tabel 5.11 dan Tabel 5.12 menjelaskan spesifikasi perangkat lunak yang dibutuhkan untuk mengembangkan sistem.

Tabel 5.11 Spesifikasi Perangkat Keras

Perangkat	Model
Laptop	MSI GL62QF
Processor	Intel® Core™ i7-6700HQ

Hard Disk	WDC WD10JPVX-22JC3T0 7200 SATA 1 TB
Graphic Card	GeForce GTX 960M 4GB VRAM
RAM	8192 MB DDR4 SODIMM
Monitor	LG IPS Full HD 22”

5.2.6.2 Spesifikasi Perangkat Lunak

Aplikasi *BelajarJava* dikembangkan dengan menggunakan perangkat *Personal Computer*(PC) dengan menggunakan Bahasa Java. Perangkat lunak yang dipakai untuk mengembangkan aplikasi ialah Android Studio versi 3.0.1 dengan memakai SDK 27.1.1.

Tabel 5.12 Spesifikasi Perangkat Lunak

Perangkat Lunak	Model
OS	Windows 10 Enterprise 2016 LTSB 64-bit
Bahasa Pemrograman	Java
IDE	Android Studio 3.0.1
SDK	27.1.1
Gradle	4.1
Database	Firebase Database 11.0.4

5.2.7 Test Driven Development (TDD)

Terdapat 2 *level* pada teknik TDD, yakni *Acceptance Test Driven Development* (ATDD) dan *Developer Test Driven Development* (DTDD). ATDD dilakukan dengan memeriksa fungsional dan melihat hasil eksekusi dari perangkat lunak, umumnya dikenal dengan sebutan *black box testing*. Sedangkan DTDD dilakukan dengan membagi pengujian menjadi beberapa unit pengujian, umumnya dikenal dengan sebutan *white box testing* (Ambler, 2013).

TDD dilakukan untuk menguji alur logika perancangan algoritma sebelum melakukan implementasi kode tersebut. Hal ini dilakukan agar alur logika yang telah dirancang sebelumnya telah berjalan sesuai rancangan. Pada TDD menghasilkan beberapa hal, yaitu skenario uji, implementasi kode uji dan hasil uji dari setiap skenario uji. Tidak dilakukan pengujian algoritma yang berkaitan dengan *use case* ubah materi singkat, buat contoh *source code*, ubah contoh *source code*, buat latihan, dan ubah latihan karena *method* `buatMateriSingkat` memiliki alur yang sama sehingga hanya dilakukan pengujian satu kali.

5.2.7.1 Pengujian Algoritma buatMateriSingkat()

Pengujian algoritma `buatMateriSingkat` dilakukan untuk menguji alur logika *method* `buatMateriSingkat`. Pengujian ini menggunakan empat skenario, yakni kondisi tidak terdapat gambar dan gagal, kondisi terdapat gambar dan gagal, kondisi tidak terdapat gambar dan sukses, dan kondisi terdapat gambar dan sukses.

a. Skenario pengujian

Tabel 5.13 Skenario Pengujian buatMateriSingkat() Tidak Terdapat Gambar dan Gagal

Kasus Uji	Buat materi singkat
Tujuan Pengujian	Membuktikan form terdapat gambar dan gagal membuat konten
Prosedur Pengujian	1. Inisialisasi nilai variabel 2. Masuk kondisi if dengan kondisi variabel filePath bernilai null atau tidak 3. Lakukan pembuktian apakah sukses atau tidak
Hasil yang Diharapkan	Pesan gagal ditampilkan dan tidak terdapat gambar

Tabel 5.14 Skenario Pengujian buatMateriSingkat() Terdapat Gambar dan Gagal

Kasus Uji	Buat materi singkat
Tujuan Pengujian	Membuktikan form terdapat gambar dan gagal membuat konten
Prosedur Pengujian	1. Inisialisasi nilai variabel 2. Masuk kondisi if dengan kondisi variabel filePath bernilai null atau tidak 3. Lakukan pembuktian apakah sukses atau tidak
Hasil yang Diharapkan	Pesan gagal ditampilkan dan terdapat gambar

Tabel 5.15 Skenario Pengujian buatMateriSingkat() Tidak Terdapat Gambar dan Sukses

Kasus Uji	Buat materi singkat
Tujuan Pengujian	Membuktikan form tidak terdapat gambar dan sukses membuat konten
Prosedur Pengujian	1. Inisialisasi nilai variabel 2. Masuk kondisi if dengan kondisi variabel filePath bernilai null atau tidak 3. Lakukan pembuktian apakah sukses atau tidak
Hasil yang Diharapkan	Pesan sukses ditampilkan dan tidak terdapat gambar

Tabel 5.16 Skenario Pengujian buatMateriSingkat() Terdapat Gambar dan Sukses

Kasus Uji	Buat materi singkat
Tujuan Pengujian	Membuktikan form terdapat gambar dan gagal membuat konten
Prosedur Pengujian	1. Inisialisasi nilai variabel 2. Masuk kondisi if dengan kondisi variabel filePath bernilai null atau tidak 3. Lakukan pembuktian apakah sukses atau tidak
Hasil yang Diharapkan	Pesan gagal ditampilkan dan tidak terdapat gambar

b. Implementasi skenario pengujian

Tabel 5.17 Skenario Pengujian buatMateriSingkat() Tidak Terdapat Gambar dan Gagal

```
1  @Test
2  public void buatMateriSingkat(){
3      filePath = null;
4      boolean isSuccess = false;
5      if(filePath != null){
6          assertTrue("Buat dengan gambar", isSuccess);
7      } else {
8          assertTrue("Buat tanpa gambar", isSuccess);
9      }
10 }
```

Tabel 5.18 Skenario Pengujian buatMateriSingkat() Terdapat Gambar dan Gagal

```
1  @Test
2  public void buatMateriSingkat(){
3      filePath = "ada gambar";
4      boolean isSuccess = false;
5      if(filePath != null){
6          assertTrue("Buat dengan gambar", isSuccess);
7      } else {
8          assertTrue("Buat tanpa gambar", isSuccess);
9      }
10 }
```

Tabel 5.19 Skenario Pengujian buatMateriSingkat() Tidak Terdapat Gambar dan Sukses

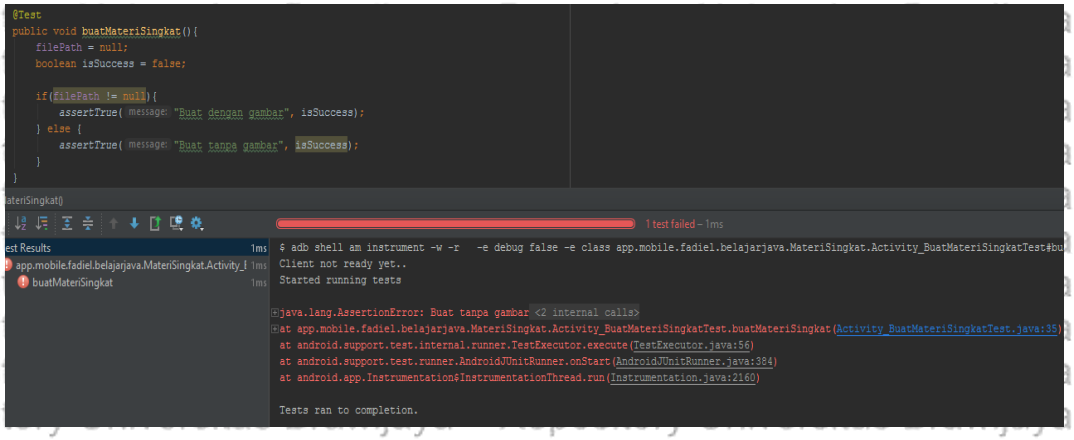
```
1  @Test
2  public void buatMateriSingkat(){
3      filePath = null;
4      boolean isSuccess = true;
5      if(filePath != null){
6          assertTrue("Buat dengan gambar", isSuccess);
7      } else {
8          assertTrue("Buat tanpa gambar", isSuccess);
9      }
10 }
```

Tabel 5.20 Skenario Pengujian buatMateriSingkat() Terdapat Gambar dan Sukses

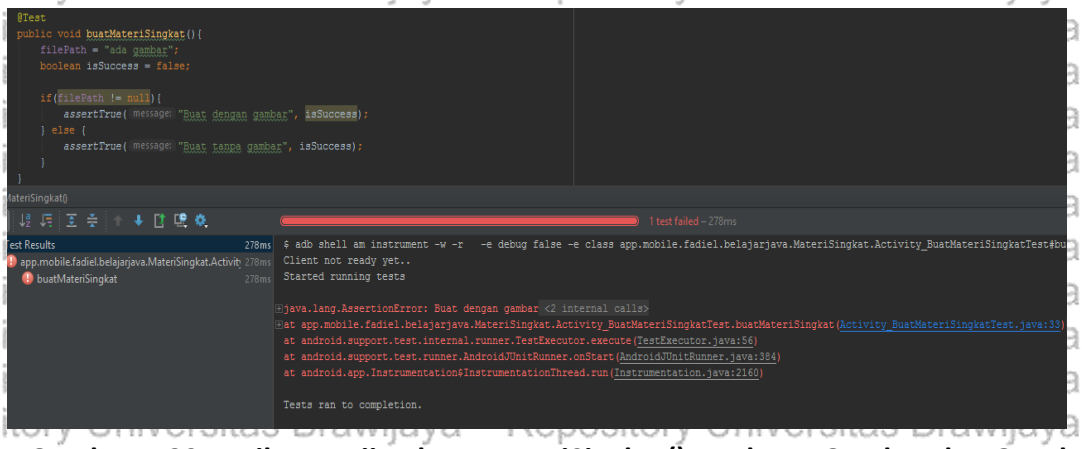
```
1  @Test
2  public void buatMateriSingkat(){
3      filePath = "ada gambar";
4      boolean isSuccess = true;
5      if(filePath != null){
6          assertTrue("Buat dengan gambar", isSuccess);
7      } else {
8          assertTrue("Buat tanpa gambar", isSuccess);
```

```
9   }
10  }
```

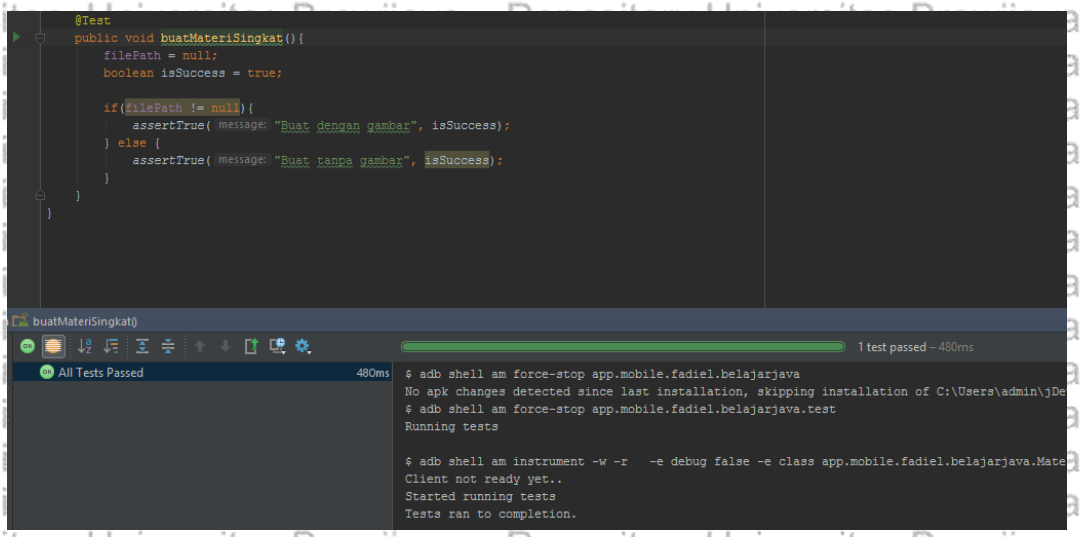
c. Hasil Pengujian



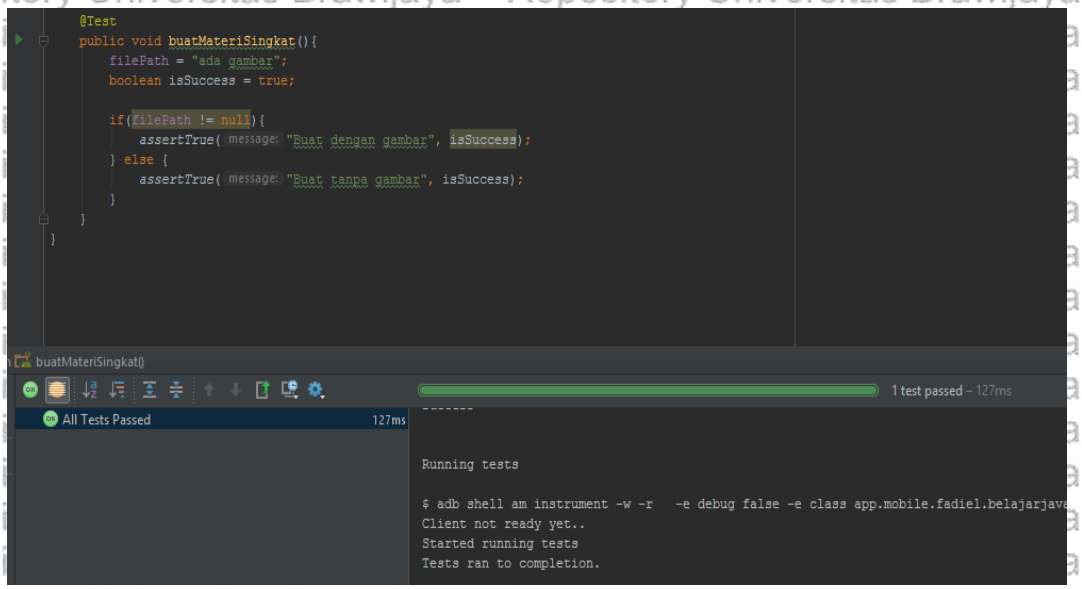
Gambar 5.38 Hasil Pengujian buatMateriSingkat() Tidak Terdapat Gambar dan Gagal



Gambar 5.39 Hasil Pengujian buatMateriSingkat() Terdapat Gambar dan Gagal



Gambar 5.40 Hasil Pengujian buatMateriSingkat() Tidak Terdapat Gambar dan Sukses



Gambar 5.41 Hasil Pengujian buatMateriSingkat() Terdapat Gambar dan Sukses

5.2.7.2 Pengujian Algoritma jawabLatihan()

Pengujian algoritma `jawabLatihan` dilakukan untuk menguji alur logika *method* `jawabLatihan`. Pengujian ini menggunakan dua skenario, yakni kondisi ketika jawaban salah dan kondisi ketika jawaban benar.

a. Skenario pengujian

Tabel 5.21 Skenario Pengujian jawabLatihan() Kondisi Jawaban Benar

Kasus Uji	Cek jawaban benar
Tujuan Pengujian	Membuktikan jawaban benar atau salah

Prosedur Pengujian	1. Inisialisasi variabel jawaban 2. Cek jawaban benar atau tidak
Hasil yang Diharapkan	Pesan berhasil ditampilkan

Tabel 5.22 Skenario Pengujian jawabLatihan() Kondisi Jawaban Salah

Kasus Uji	Cek form kondisi tidak kosong
Tujuan Pengujian	Membuktikan jawaban benar atau salah
Prosedur Pengujian	1. Inisialisasi variabel jawaban 2. Cek jawaban benar atau tidak
Hasil yang Diharapkan	Pesan gagal ditampilkan

b. Implementasi skenario pengujian

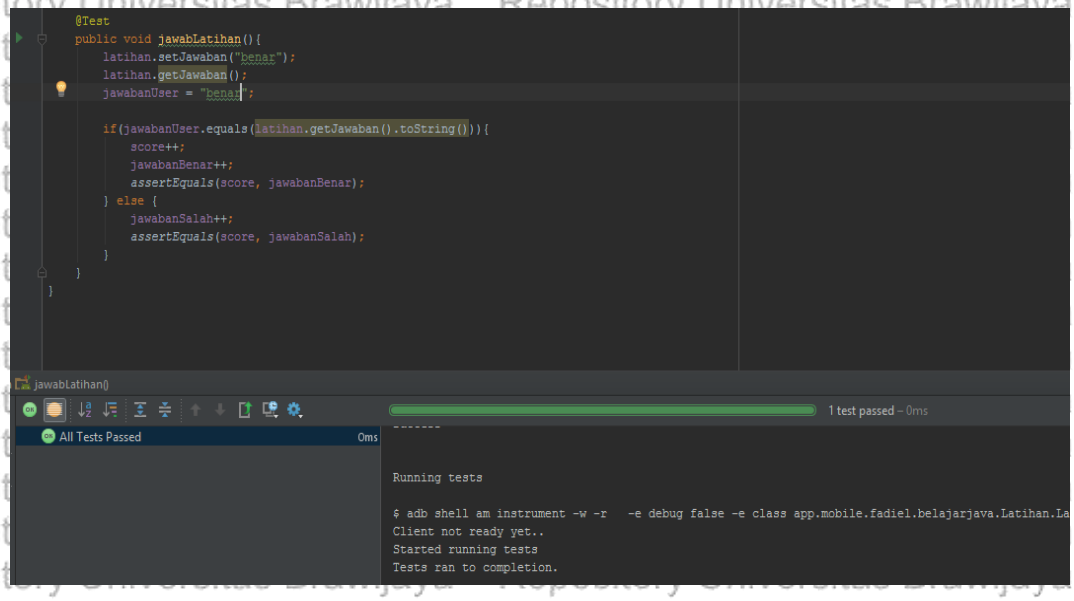
Tabel 5.23 Skenario Pengujian jawabLatihan() Kondisi Jawaban Benar

```
1   @Test
2   public void jawabLatihan(){
3       latihan.setJawaban("benar");
4       latihan.getJawaban();
5       jawabanUser = "benar";
6       if(jawabanUser.equals(latihan.getJawaban().toString())){
7           score ++;
8           jawabanBenar++;
9           assertEquals(score, jawabanBenar);
10      } else {
11      jawabanSalah++;
12      assertEquals(score, jawabanSalah);
13      }
14  }
```

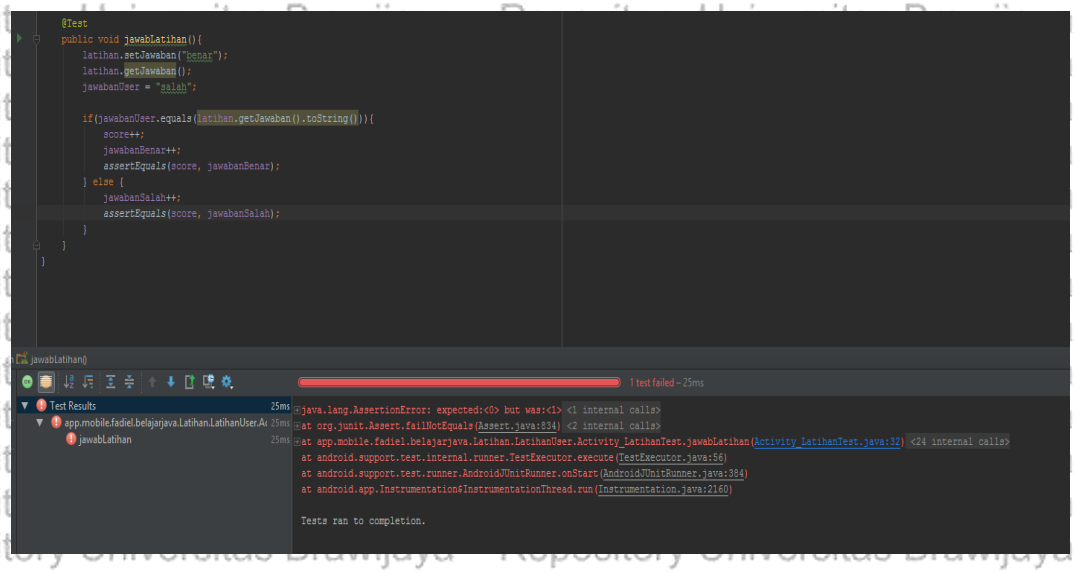
Tabel 5.24 Skenario Pengujian jawabLatihan() Kondisi Jawaban Salah

```
1   @Test
2   public void jawabLatihan(){
3       latihan.setJawaban("benar");
4       latihan.getJawaban();
5       jawabanUser = "salah";
6       if(jawabanUser.equals(latihan.getJawaban().toString())){
7           score++;
8           jawabanBenar++;
9           assertEquals(score, jawabanBenar);
10      } else {
11          jawabanSalah++;
12          assertEquals(score, jawabanSalah);
13      }
14  }
```

c. Hasil Pengujian



Gambar 5.42 Hasil Pengujian jawabLatihan() Kondisi Jawaban Benar



Gambar 5.43 Hasil Pengujian jawabLatihan() Kondisi Jawaban Salah

5.2.8 Implementasi Kelas

Perancangan Kelas yang telah dirancang sebelumnya, kemudian diimplementasikan pada program dengan ekstensi .java. Implementasi kelas pada Aplikasi *BelajarJava* terdapat pada Tabel 5.25 sampai Tabel 5.30.

Tabel 5.25 Implementasi Kelas Autentifikasi pada Aplikasi *BelajarJava*

No.	*Package*	Kelas atau *Interface*
1	Autentifikasi	LoginActivity.java
2	Autentifikasi	RegisterActivity.java

Tabel 5.26 Implementasi Kelas Contoh *Source Code* pada Aplikasi *BelajarJava*

No.	*Package*	Kelas atau *Interface*
1	ContohSourceCode	Activity_BuatContohSourceCode.java
2	ContohSourceCode	Activity_ContohSourceCode.java
3	ContohSourceCode	Activity_EditContohSourceCode.java
4	ContohSourceCode	Activity_PilihContohSourceCode.java
5	ContohSourceCode	Adapter_ContohSourceCode.java
6	ContohSourceCode	LoadDataContohSourceCode.java
7	ContohSourceCode	Model_ContohSourceCode.java

Tabel 5.27 Implementasi Kelas Materi Singkat pada Aplikasi *BelajarJava*

No.	*Package*	Kelas atau *Interface*
1	MateriSingkat	Activity_BuatMateriSingkat.java
2	MateriSingkat	Activity_ MateriSingkat.java
3	MateriSingkat	Activity_Edit MateriSingkat.java
4	MateriSingkat	Activity_Pilih MateriSingkat.java
5	MateriSingkat	Adapter_ MateriSingkat.java
6	MateriSingkat	LoadData MateriSingkat.java
7	MateriSingkat	Model_MateriSingkat.java

Tabel 5.28 Implementasi Kelas Latihan pada Aplikasi *BelajarJava*

No.	*Package*	Kelas atau *Interface*
1	BuatLatihan	Activity_Latihan.java
2	BuatLatihan	Fragment_BuatLatihanPG.java
3	BuatLatihan	Fragment_BuatLatihanEssay.java
4	UbahLatihan	Activity_UbahLatihan.java
5	UbahLatihan	Fragment_UbahLatihanPG.java
6	UbahLatihan	Fragment_UbahLatihanEssay.java
7	LatihanModelUser	Model_Latihan.java
8	LatihanModel	Model_LatihanPG.java
9	LatihanModel	Model_LatihanEssay.java

10	LatihanUser	Activity_LatihanPG.java
11	LatihanUser	Activity_LatihanEssay.java
12	LatihanUser	Activity_PilihMateriLatihan.java

Tabel 5.29 Implementasi Kelas Homepage pada Aplikasi *BelajarJava*

No.	*Package*	Kelas atau *Interface*
1	Homepage	Fragment_Home.java
2	Homepage	MainActivity.java

Tabel 5.30 Implementasi Kelas Profil pada Aplikasi *BelajarJava*

No.	*Package*	Kelas atau *Interface*
1	Profil	EditProfileActivity.java
2	Profil	UserInfo.java

5.2.9 Implementasi Kode Program

Subbab ini menjelaskan hasil implementasi kode aplikasi *BelajarJava* yang telah dikembangkan. Pada bagian ini hanya dijelaskan *method* yang memiliki peran penting pada aplikasi. *Method* yang dijelaskan juga dilakukan *reuse* oleh pengembang untuk diimplementasikan pada kelas yang lain.

5.2.9.1 Implementasi *Method* buatMateriSingkat()

Method `buatMateriSingkat` merupakan *method* untuk menambahkan isi konten pada halaman materi singkat. Pada *Method* ini variabel diakses menggunakan *Model class* yang akan disimpan di *database Firebase*. *Model class* yang diakses pada *method* ini adalah *class* `Model_MateriSingkat`. *Method* ini juga berfungsi untuk memanggil salah satu *method* dari dua method yang bisa dipanggil, yakni *method* `buatDenganGambar atau buatTanpaGambar`.

Tabel 5.31 Implementasi *Method* buatMateriSingkat()

```
1  private void buatMateriSingkat() {
2   databaseReference =
3  FirebaseDatabase.getInstance().getReference();
4           DatabaseReference postsRef =
5  databaseReference.child("materiSingkat").child(materiSingkat.getId
6  ());
7           DatabaseReference newPostRef =
8  postsRef.child("materi_singkat").push();
9           materiSingkat.setId(newPostRef.getKey());
10          storageReference =
11 FirebaseStorage.getInstance().getReference();
12 StorageReference storageRef =
```

```
storageReference.child("judulMateri/"+
UUID.randomUUID().toString());

        final ProgressDialog progressDialog = new
ProgressDialog(Activity_BuatMateriSingkat.this);
        progressDialog.setTitle("Menambah Konten");
        progressDialog.show();

        if(filePath != null){
            buatDenganGambar(storageRef, progressDialog,
newPostRef);
        } else {
            buatTanpaGambar(progressDialog, newPostRef);
        }
    }
```

Tabel 5.32 Penjelasan *Method* buatMateriSingkat()

Nomor Kode Program	Penjelasan
1	Inisialisasi *method* buatMateriSingkat
2-8	Inisiliasi *database Firebase*
9-14	Inisialisasi *Firebase Storage* untuk menyimpan *file* berupa gambar
15-18	Memanggil *class* ProgressDialog untuk menambahkan dialog saat melakukan *upload* gambar
19-21	Kondisi if jika variabel filePath tidak bernilai null akan memanggil *method* buatDenganGambar
22-24	Kondisi else jika variabel filePath bernilai null akan memanggil *method* buatTanpaGambar
25	Kurung penutup dari *method* buatMateriSingkat

5.2.9.2 Implementasi *Method* buatSourceCode()

Method `buatSourceCode` merupakan *method* untuk menambahkan isi konten pada halaman contoh *source code*. Pada *Method* ini variabel diakses menggunakan *model class* yang akan disimpan di *database Firebase*. *Model class* yang diakses pada *method* ini adalah *class* `Model_SourceCode`. *Method* ini juga berfungsi untuk memanggil salah satu *method* dari dua method yang bisa dipanggil, yakni *method* `buatDenganGambar` atau `buatTanpaGambar`.

Tabel 5.33 Implementasi *Method* buatSourceCode()

```
1  private void buatSourceCode(){
2          databaseReference =
3  FirebaseDatabase.getInstance().getReference();
4          final DatabaseReference newPostsRef =
5  databaseReference.child("source_code").push();
6          newPostsRef.keepSynced(true);
```

```
7           sourceCode.setId(newPostsRef.getKey());
8           sourceCode.setGambarInput("");
9           sourceCode.setGambarOutput("");

10          storageReference =
11  FirebaseStorage.getInstance().getReference();
12          StorageReference storageRef1 =
13  storageReference.child("source_code/"+
14  UUID.randomUUID().toString());
15          StorageReference storageRef2 =
16  storageReference.child("source_code/"+
17  UUID.randomUUID().toString());
18          final ProgressDialog progressDialog = new
19  ProgressDialog(Activity_BuatSourceCode.this);
20          progressDialog.setTitle("Menambah Konten");
21          progressDialog.show();

22          if(filePathInput != null && filePathOutput != null){
23              buatDenganGambarInputdanOutput(storageRef1,
24  storageRef2, postsRef, newPostsRef, progressDialog);
25          } else if (filePathInput != null || filePathOutput !=
26  null) {
27              if(filePathInput != null) {
28                  buatDenganGambarInput(storageRef1, newPostsRef,
29  progressDialog);
30              } else if (filePathOutput != null) {
31                  buatDenganGambarOutput(storageRef1, newPostsRef,
32  progressDialog);
33              }
34          } else {
35              buatTanpaGambar(newPostsRef, progressDialog);
36          }
37      }
```

Tabel 5.34 Penjelasan *Method* buatContohSourceCode()

Nomor Kode Program	Penjelasan
1	Inisialisasi *method* buatSourceCode
2-6	Inisialisasi *database Firebase*
7	Set nilai variabel id dari *class* Model_SourceCode
8	Set nilai variabel gambarInput dari *class* Model_SourceCode
9	Set nilai variabel gambarOutput dari *class* Model_SourceCode
10-17	Inisialisasi *Firebase Storage* untuk menyimpan *file* berupa gambar
17-21	Memanggil *class* ProgressDialog untuk menambahkan dialog saat melakukan *upload* gambar
22-25	Kondisi if jika variabel filePathInput tidak bernilai null dan filePathOutput tidak bernilai null akan memanggil *method* buatDenganGambarInputdanOutput

26-30	Kondisi if jika variabel filePathInput tidak bernilai null akan memanggil *method* buatDenganGambarInput
31-33	Kondisi if jika variabel filePathOutput tidak bernilai null akan memanggil *method* buatDenganGambarOutput
34-36	Kondisi else jika variabel filePathInput bernilai null dan filePathOutput bernilai null akan memanggil *method* buatTanpaGambar
37	Kurung penutup dari *method* buatSourceCode

5.2.9.3 Implementasi *Method* ubahMateriSingkat()

Method `ubahMateriSingkat` merupakan *method* untuk mengubah isi konten pada halaman materi singkat. Hampir sama pada *method* `buatMateriSingkat`, perbedaan pada *method* ini yakni tidak mendeklarasikan variabel `FirebaseStorage`, melainkan variabel `DatabaseReference` yang mewakili lokasi dari *database* materi singkat.

Tabel 5.35 Implementasi *Method* ubahMateriSingkat()

```
1   private void buatMateriSingkat() {
2           databaseReference =
3   FirebaseDatabase.getInstance().getReference();
4           DatabaseReference postsRef =
5   databaseReference.child("materi_singkat").child(materi.getId());
6           materiSingkat.setId(newPostRef.getKey());
7           storageReference =
8   FirebaseStorage.getInstance().getReference();
9           StorageReference storageRef =
10  storageReference.child("subMateri/"+
11  UUID.randomUUID().toString());
12          final ProgressDialog progressDialog = new
13  ProgressDialog(Activity_BuatSubMateri.this);
14          progressDialog.setTitle("Menambah Konten");
15          progressDialog.show();
16          if(filePath != null){
17              buatDenganGambar(storageRef, progressDialog,
18  newPostRef);
19          } else {
20              buatTanpaGambar(progressDialog, newPostRef);
21          }
22      }
```

Tabel 5.36 Penjelasan *Method* ubahMateriSingkat()

Nomor Kode Program	Penjelasan
1	Inisialisasi *method* ubahMateri

2-5	Inisiliasi *database Firebase*
6	Set nilai variabel id dari *class* Model_SubMateri
7-11	Inisialisasi data *storage Firebase*
12-15	Memanggil *class* ProgressDialog untuk menambahkan dialog
16-18	Kondisi if jika variabel filePath tidak bernilai null akan memanggil *method* ubahDenganGambar
19-21	Kondisi else jika variabel filePath bernilai null akan memanggil *method* ubahTanpaGambar
22	Penutup *method* buatMateri

5.2.9.4 Implementasi *Method* ubahSourceCode()

Method `ubahSourceCode` merupakan *method* untuk mengubah isi konten pada halaman materi singkat. Hampir sama pada *method* `buatMateri`, perbedaan pada *method* ini yakni tidak mendeklarasikan variabel `FirebaseStorage`, melainkan variabel `DatabaseReference` yang mewakili lokasi dari *database* materi singkat.

Tabel 5.37 Implementasi *Method* ubahContohSourceCode()

```
1   private void buatSourceCode(){
2           databaseReference =
3   FirebaseDatabase.getInstance().getReference();
4           final DatabaseReference postsRef =
5   databaseReference.child("contoh_source_code").child(contohSourceCo
6   de.getId());
7           newPostsRef.keepSynced(true);
8           sourceCode.setId(newPostsRef.getKey());
9           sourceCode.setGambarInput("");
10          sourceCode.setGambarOutput("");
11          storageReference =
12  FirebaseStorage.getInstance().getReference();
13          StorageReference storageRef1 =
14  storageReference.child("source_code/"+
15  UUID.randomUUID().toString());
16          StorageReference storageRef2 =
17  storageReference.child("source_code/"+
18  UUID.randomUUID().toString());
19          final ProgressDialog progressDialog = new
20  ProgressDialog(Activity_BuatSourceCode.this);
21          progressDialog.setTitle("Menambah Konten");
22          progressDialog.show();
23          if(filePathInput != null && filePathOutput != null){
24              buatDenganGambarInputdanOutput(storageRef1,
25  storageRef2, postsRef, newPostsRef, progressDialog);
26          } else if (filePathInput != null || filePathOutput !=
27  null) {
28              if(filePathInput != null) {
```

```
29                  buatDenganGambarInput(storageRef1, newPostsRef,
30  progressDialog);
31              } else if (filePathOutput != null) {
32                  buatDenganGambarOutput(storageRef1, newPostsRef,
33  progressDialog);
34              }
35          } else {
36              buatTanpaGambar(newPostsRef, progressDialog);
37          }
38      }
```

Tabel 5.38 Penjelasan *Method* ubahContohSourceCode()

Nomor Kode Program	Penjelasan
1	Inisialisasi *method* ubahContohSourceCode
2-7	Inisialisasi *database Firebase*
8	Set nilai variabel id dari *class* Model_ContohSourceCode
9	Set nilai variabel gambarInput dari *class* Model_ContohSourceCode
10	Set nilai variabel gambarOutput dari *class* Model_ContohSourceCode
11-18	Inisialisasi data *storage Firebase*
19-22	Memanggil *class* ProgressDialog untuk menambahkan dialog
23-25	Kondisi if jika variabel filePathInput tidak bernilai null dan filePathOutput tidak bernilai null akan memanggil *method* ubahDenganGambarInputdanOutput
26-30	Kondisi if jika variabel filePathInput tidak bernilai null akan memanggil *method* ubahDenganGambarInput
31-34	Kondisi if jika variabel filePathOutput tidak bernilai null akan memanggil *method* ubahDenganGambarOutput
35-37	Kondisi else jika variabel filePathInput bernilai null dan filePathOutput bernilai null akan memanggil *method* ubahTanpaGambar
38	Penutup *method* ubahSourceCode

5.2.9.5 Implementasi *Method* jawabLatihan()

Method `jawabLatihan` merupakan *method* untuk menjawab soal latihan. *Method* ini juga berguna untuk memuat pertanyaan dan melakukan pengecekan apakah jawaban yang diberikan benar atau salah.

Tabel 5.39 Implementasi *Method* jawabLatihan()

```
1   private void jawabLatihan() {
2       if(Model_Latihan.materiLatihan.equals("Perulangan For")){
3           if(index < totalPertanyaan-1) {
4   if(checkJawaban.getJawabanUser().equals(Model_Latihan.latihanTeba
5   kOutputPGListPF.get(index).getKunciJawaban())){
6                   score+=10;
7                   jawabanBenar++;
8                   loadLatihan(++index);
9               }
10              else {
11                  score+=0;
12                  jawabanSalah++;
13                  loadLatihan(++index);
14              }
15          }
16          else {
17              score +=
18  (checkJawaban.getJawabanUser().equals(Model_Latihan.latihanTebakO
19  utputPGListPF.get(index).getKunciJawaban())) ? +10 : +0;
20              jawabanBenar +=
21  (checkJawaban.getJawabanUser().equals(Model_Latihan.latihanTebakO
22  utputPGListPF.get(index).getKunciJawaban())) ? +1 : +0;
23              jawabanSalah +=
24  (checkJawaban.getJawabanUser().equals(Model_Latihan.latihanTebakO
25  utputPGListPF.get(index).getKunciJawaban())) ? +0 : +1;
26
27              Intent intentLv2 = new Intent(Activity_Level1.this,
28  Activity_Level2.class);
29              intentLv2.putExtra("scoreLv1", score);
30              startActivity(intentLv2);
31              finish();
32          }
33      } if (Model_Latihan.materiLatihan.equals("Perulangan
34  While")){
35          if(index < totalPertanyaan-1) {
36  if(checkJawaban.getJawabanUser().equals(Model_Latihan.latihanTeba
37  kOutputPGListPW.get(index).getKunciJawaban())){
38                  score+=10;
39                  jawabanBenar++;
40                  loadLatihan(++index);
41              }
42              else {
43                  score+=0;
44                  jawabanSalah++;
45                  loadLatihan(++index);
46              }
47          } else {
48              score +=
49   (checkJawaban.getJawabanUser().equals(Model_Latihan.latihanTebakO
50  utputPGListPW.get(index).getKunciJawaban())) ? +10 : +0;
51              jawabanBenar +=
52   (checkJawaban.getJawabanUser().equals(Model_Latihan.latihanTebakO
53  utputPGListPW.get(index).getKunciJawaban())) ? +1 : +0;
54              jawabanSalah +=
55   (checkJawaban.getJawabanUser().equals(Model_Latihan.latihanTebakO
56  utputPGListPW.get(index).getKunciJawaban())) ? +0 : +1;
57
58              Intent intentLv2 = new Intent(Activity_Level1.this,
59  Activity_Level2.class);
```

```
60              intentLv2.putExtra("scoreLv1", score);
61              startActivity(intentLv2);
62              finish();
63          }
64      } if (Model_Latihan.materiLatihan.equals("Perulangan Do-
65  While")){
66          if(index < totalPertanyaan-1) {
67
68  if(checkJawaban.getJawabanUser().equals(Model_Latihan.latihanTeba
69  kOutputPGListPDW.get(index).getKunciJawaban())){
70              score+=10;
71              jawabanBenar++;
72              loadLatihan(++index);
73          }
74          else {
75              score+=0;
76              jawabanSalah++;
77              loadLatihan(++index);
78          }
79      } else {
80          score +=
81  (checkJawaban.getJawabanUser().equals(Model_Latihan.latihanTebakO
82  utputPGListPDW.get(index).getKunciJawaban())) ? +10 : +0;
83          jawabanBenar +=
84  (checkJawaban.getJawabanUser().equals(Model_Latihan.latihanTebakO
85  utputPGListPDW.get(index).getKunciJawaban())) ? +1 : +0;
86          jawabanSalah +=
87  (checkJawaban.getJawabanUser().equals(Model_Latihan.latihanTebakO
88  utputPGListPDW.get(index).getKunciJawaban())) ? +0 : +1;
89
90          Intent intentLv2 = new Intent(Activity_Level1.this,
91  Activity_Level2.class);
92          intentLv2.putExtra("scoreLv1", score);
93          startActivity(intentLv2);
94
95          finish();
96      }
97  }
98  }
```

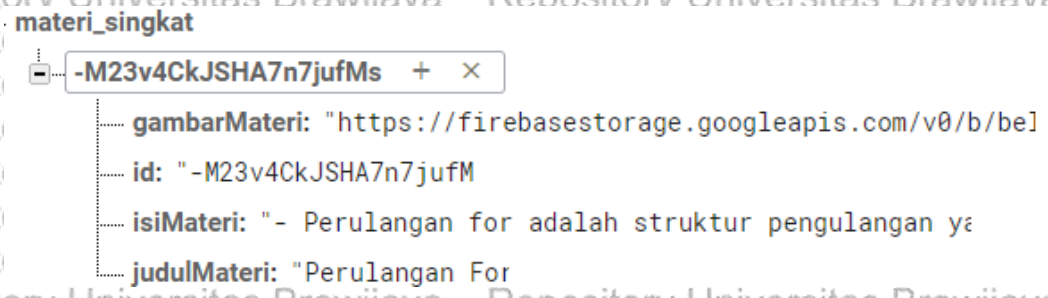
Tabel 5.40 Penjelasan *Method* jawabLatihan()

Nomor Kode Program	Penjelasan
1	Inisialisasi *method* jawabLatihan
2	Kondisi if jika variabel materiLatihan = “Perulangan For”
3-15	Kondisi if jika index < totalPertanyaan-1 maka jawaban dari latihan dengan materi perulangan for akan dicek dan dihitung skor-nya
16-32	Kondisi else jika index > totalPertanyaan-1 maka jawaban dari pertanyaan terakhir akan dicek dan skor dari pertanyaan dengan materi perulangan for akan dikirim ke *class* Activity_Level2
33-34	Kondisi if jika variabel materiLatihan = “Perulangan While”

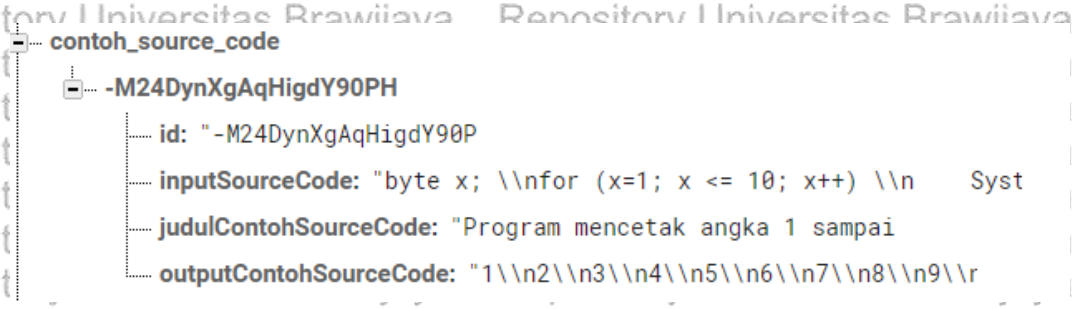
33-46	Kondisi if jika index < totalPertanyaan-1 maka jawaban dari latihan dengan materi perulangan while akan dicek dan dihitung skor-nya
47-63	Kondisi else jika index > totalPertanyaan-1 maka jawaban dari pertanyaan terakhir akan dicek dan skor dari pertanyaan dengan materi perulangan while akan dikirim ke *class* Activity_Level2
64-78	Kondisi if jika variabel materiLatihan = "Perulangan Do-While"
79-88	Kondisi if jika index < totalPertanyaan-1 maka jawaban dari latihan dengan materi perulangan do-while akan dicek dan dihitung skor-nya
89-97	Kondisi else jika index > totalPertanyaan-1 maka jawaban dari pertanyaan terakhir akan dicek dan skor dari pertanyaan dengan materi perulangan do-while akan dikirim ke *class* Activity_Level2
98	Kurung penutup dari *method* jawabLatihan

5.2.10 Implementasi Basis Data

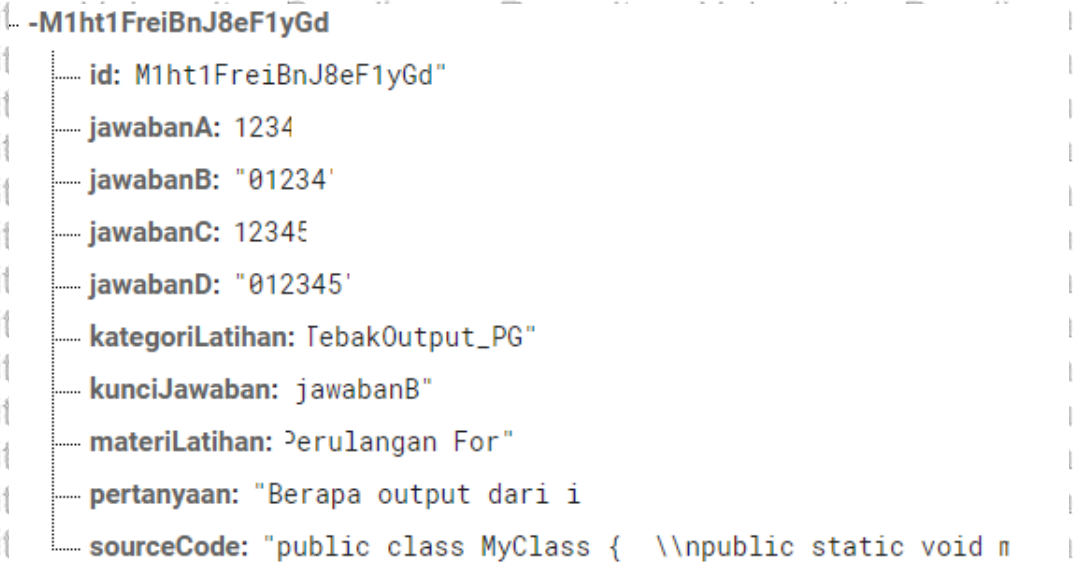
Subbab ini menjelaskan hasil implementasi basis data aplikasi *BelajarJava* pada rilis awal. Hasil implementasi basis data rilis awal aplikasi *BelajarJava* terdapat pada gambar 5.44 sampai 5.47.



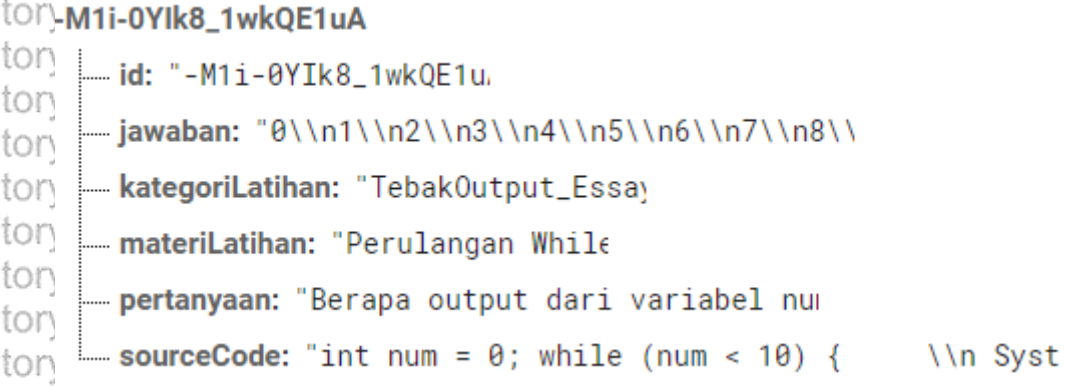
Gambar 5.44 Implementasi Basis Data Materi Singkat



Gambar 5.45 Implementasi Basis Data Contoh *Source Code*



Gambar 5.46 Implementasi Basis Data Latihan dengan Bentuk Pilihan Ganda

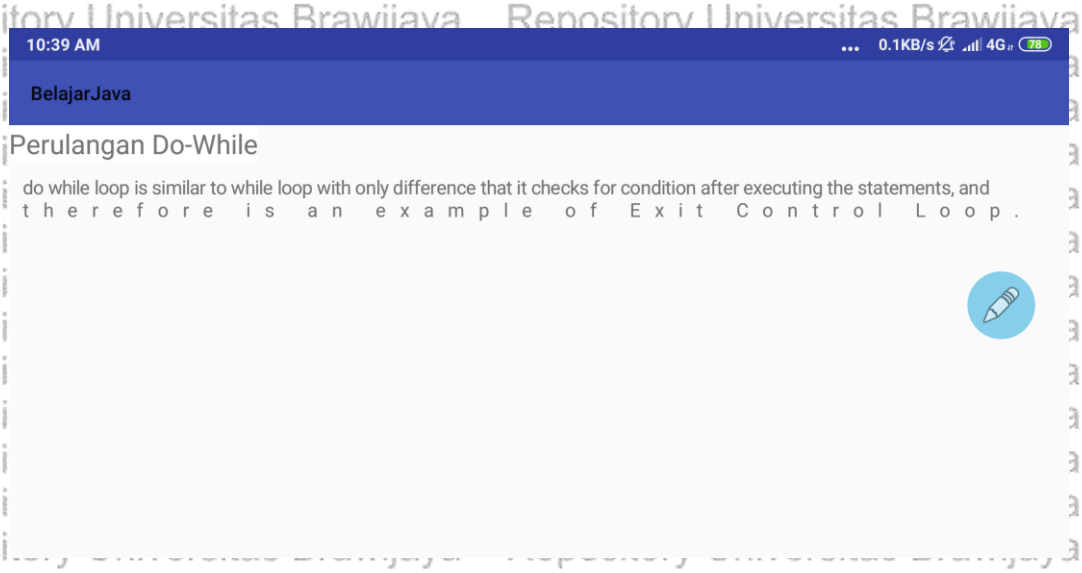


Gambar 5.47 Implementasi Basis Data Latihan dengan Bentuk *Essay*

5.2.11 Implementasi Antarmuka

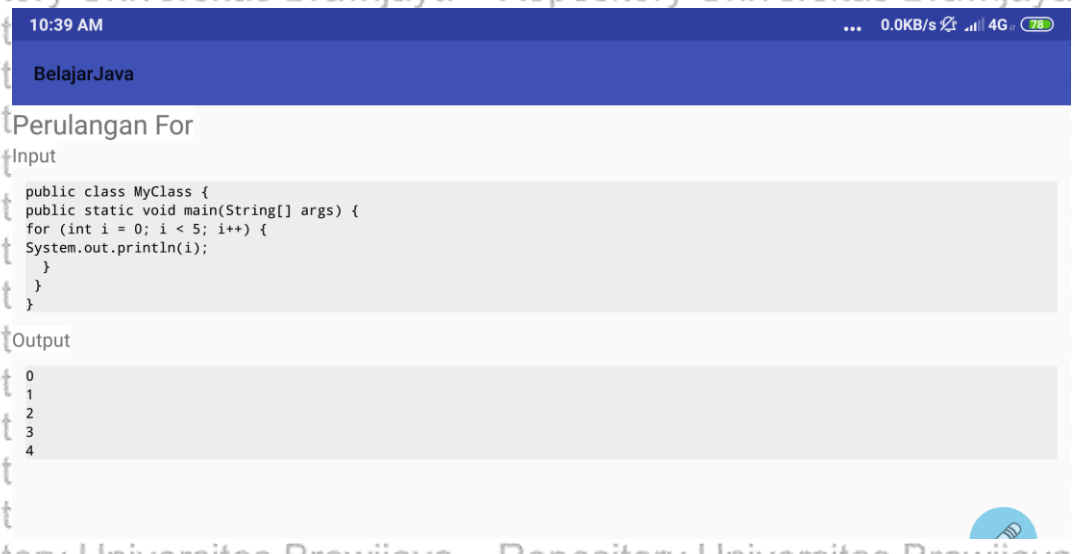
Subbab ini menjelaskan hasil implementasi antarmuka aplikasi *BelajarJava* pada rilis awal. Pada rilis awal, *layout* yang digunakan adalah *layout horizontal.*

Gambar 5.48 merupakan halaman materi singkat dimana fungsi dari halaman ini yakni untuk menampilkan konten materi singkat.



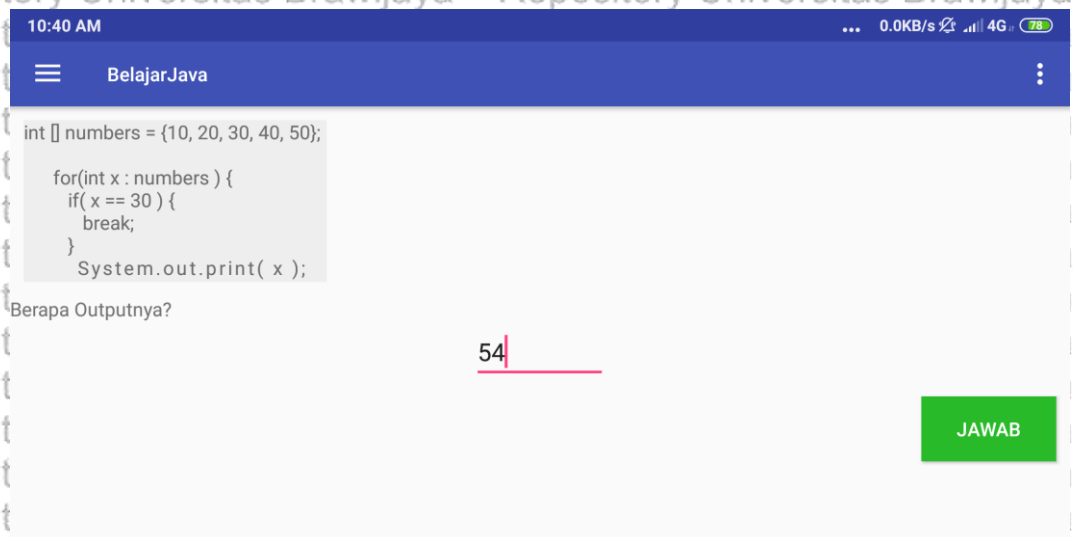
Gambar 5.48 Implementasi Antarmuka Materi Singkat pada Rilis Awal

Gambar 5.49 merupakan halaman contoh *source code* dimana pada halaman ini menampilkan konten contoh *source code*.



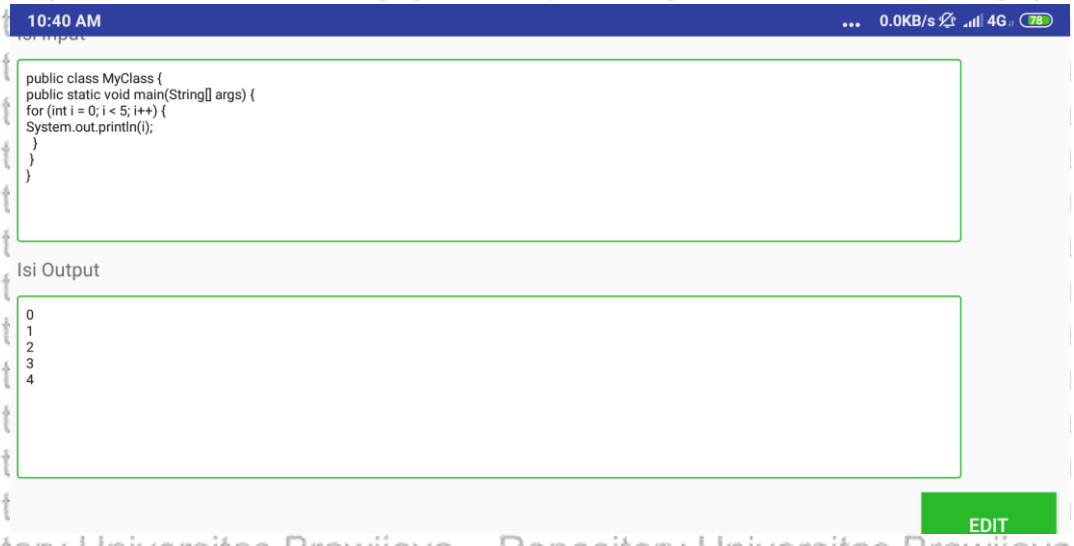
Gambar 5.49 Implementasi Antarmuka Contoh *Source Code* pada Rilis Awal

Gambar 5.50 merupakan halaman latihan dengan soal *essay* dimana halaman ini digunakan untuk menjalankan fitur latihan dengan soal *essay*.



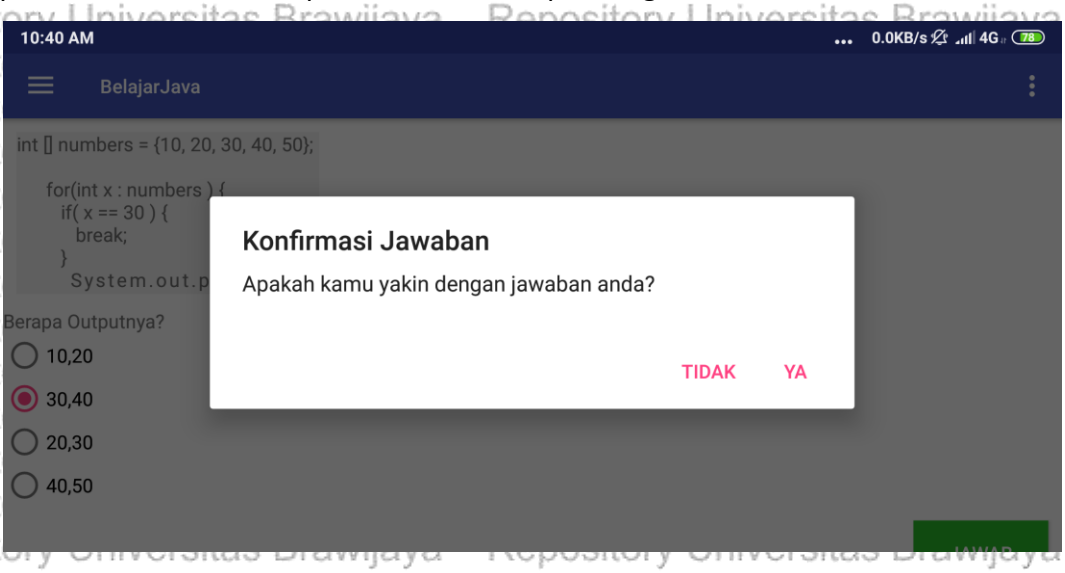
Gambar 5.50 Implementasi Antarmuka Latihan bentuk *Essay* pada Rilis Awal

Gambar 5.51 merupakan halaman CRUD latihan dimana pada halaman ini aktor dosen dapat mengubah soal latihan.



Gambar 5.51 Implementasi Antarmuka CRUD Latihan pada Rilis Awal

Gambar 5.52 merupakan halaman latihan dengan soal pilihan ganda dimana pada latihan ini menampilkan soal latihan pilihan ganda.



Gambar 5.52 Implementasi Antarmuka Latihan bentuk Pilihan Ganda pada Rilis Awal

5.2.12 Elisitasi Kebutuhan Iterasi 1

Setelah rilis implementasi awal aplikasi, aplikasi diberikan kembali ke pengguna untuk dimintai *feedback* kembali untuk dijadikan acuan untuk penyempurnaan produk.

Tabel 5.41 adalah elisitasi kebutuhan mahasiswa hasil dari fase *requirement analysis* tahap *stabilize*. Pada tahap ini, penulis menambahkan beberapa kebutuhan berdasarkan *feedback* dari pengguna yang didapatkan dari fase *productionize*. Kemudian kebutuhan tersebut ditambahkan ke dalam *use case*.

Tabel 5.41 Elisitasi Kebutuhan Mahasiswa Iterasi 1

No	Nama Kebutuhan	Prioritas Kebutuhan					Nilai Akhir
		1	2	3	4	5	
1	Sistem disediakan fitur *reset password*						
1.1	Sistem dapat menyediakan fitur untuk pengguna yang lupa *password*	0	0	0	1	3	5
2	Sistem disediakan fitur lihat skor						
2.1	Sistem dapat menyediakan fitur untuk melihat skor latihan mahasiswa	0	0	0	0	4	5

Tabel 5.42 adalah hasil dari fase *requirement analysis* tahap *stabilize*. Pada tahap ini, penulis menambahkan beberapa kebutuhan berdasarkan *feedback* dari pengguna yang didapatkan dari fase *productionize*. Kemudian kebutuhan tersebut ditambahkan ke dalam *use case*. Pada sisi dosen, dilakukan iterasi kembali. Namun, karena tidak ada perubahan pada iterasi 0, maka hanya ditulis penambahan kebutuhan saja.

Tabel 5.42 Elisitasi Kebutuhan Dosen Iterasi 1

No	Nama Kebutuhan	Prioritas Kebutuhan					Nilai Akhir
		1	2	3	4	5	
1	Sistem disediakan fitur lihat skor mahasiswa						
1.1	Sistem harus menyediakan fitur untuk melihat skor latihan mahasiswa	0	0	0	0	1	5

5.2.13 Spesifikasi Kebutuhan Iterasi 1

Spesifikasi kebutuhan fungsional *guest* mendapatkan penambahan fungsi pada iterasi 1. Yaitu fungsi dengan kode BJ_G_003 yaitu fungsi *reset password* yang terdapat pada Tabel 5.43. Fungsi ini ditambahkan karena *feedback* dari pengguna yang memerlukan fitur ini jika pengguna lupa terhadap *password* akunnya.

Tabel 5.43 Spesifikasi Kebutuhan Fungsional Guest Iterasi 1

No	Kode Fungsi	Nama Fungsi	Deskripsi	Keterangan
1	BJ_G_003	*Reset Password*	Sistem menyediakan fungsi *reset password*	Penambahan fungsi

Spesifikasi kebutuhan fungsional mahasiswa pada Tabel 5.44 mendapatkan penambahan fungsi dan penambahan deskripsi fungsi pada iterasi 1. Penambahan fungsi dengan kode BJ_M_007 yaitu fungsi lihat skor dilakukan karena terdapat perubahan fungsi pada kode BJ_M_003 yaitu fungsi latihan dimana pengguna akan mendapatkan skor setelah melakukan latihan.

Tabel 5.44 Spesifikasi Kebutuhan Fungsional Mahasiswa Iterasi 1

No	Kode Fungsi	Nama Fungsi	Deskripsi	Keterangan
1	BJ_M_007	Lihat skor	Sistem menyediakan fungsi untuk melihat skor latihan	Fungsi ditambahkan

2	BJ_M_003	Latihan	Sistem menyediakan fitur evaluasi kepada pengguna dan melakukan penghitungan skor terhadap pertanyaan yang dijawab	Penambahan fungsi

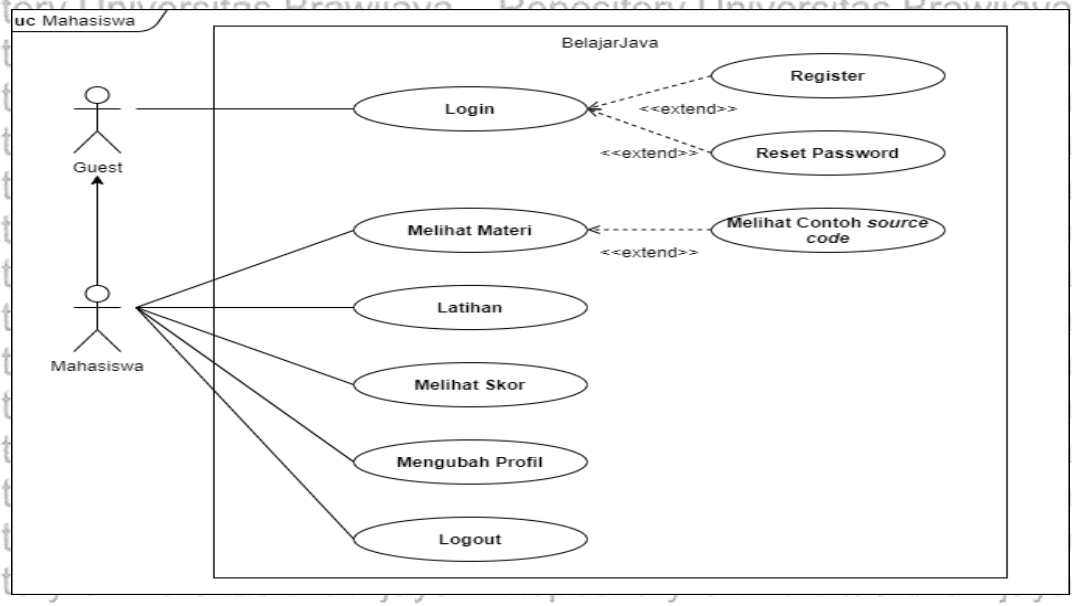
Tabel 5.45 mendapatkan penambahan fungsi pada iterasi 1. Penambahan fungsi dengan kode BJ_D_008 yaitu fungsi lihat skor mahasiswa dilakukan karena terdapat penambahan fungsi lihat skor pada aplikasi mahasiswa. Fungsi lihat skor mahasiswa bertujuan agar dosen dapat melihat skor latihan yang telah dilakukan mahasiswa. Ditambahkan juga fungsi lihat data mahasiswa dengan kode BJ_D_012 yang berintegrasi dengan kode BJ_D_013 dimana fungsi tersebut merupakan fungsi lihat skor mahasiswa agar dosen dapat mencari data mahasiswa beserta skor yang didapatkan.

Tabel 5.45 Spesifikasi Kebutuhan Fungsional Dosen Iterasi 1

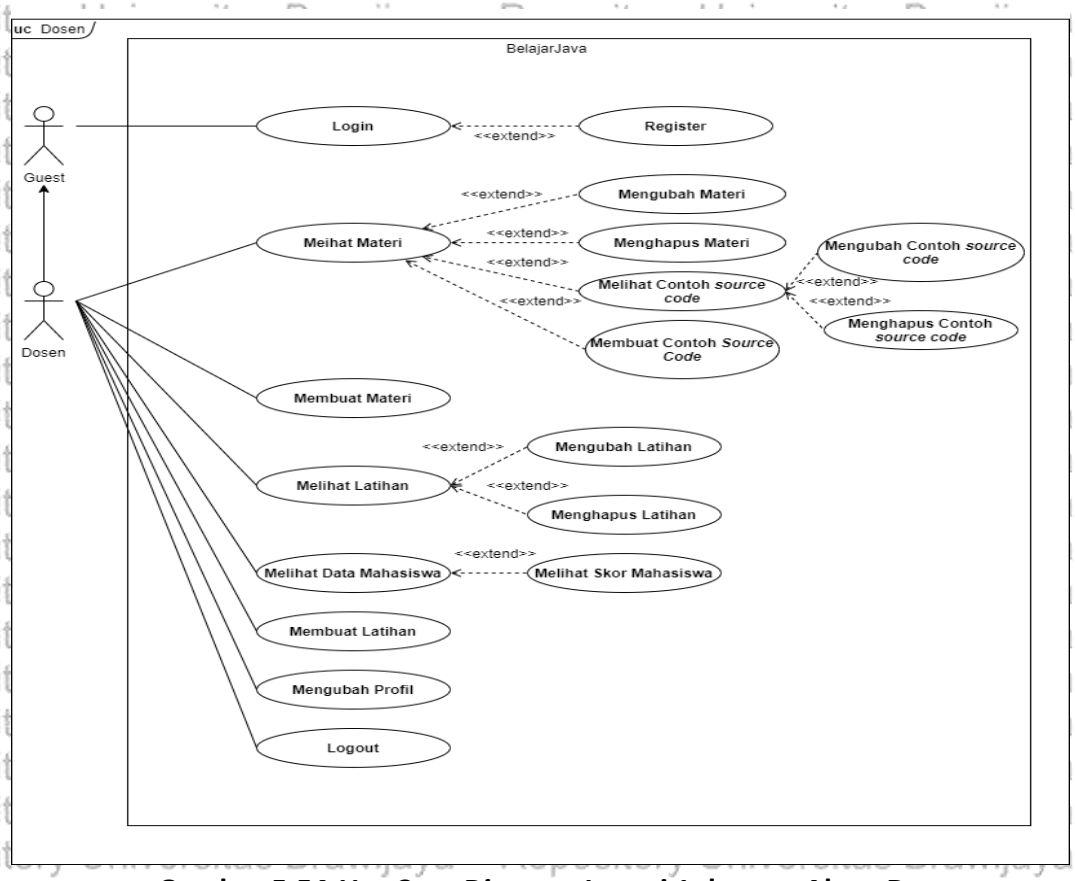
No	Kode Fungsi	Nama Fungsi	Deskripsi	Keterangan
1	BJ_D_012	Melihat data mahasiswa	Sistem menyediakan fungsi untuk melihat data identitas mahasiswa	Fungsi ditambahkan
2	BJ_D_013	Melihat skor mahasiswa	Sistem menyediakan fungsi untuk melihat skor latihan mahasiswa	Fungsi ditambahkan

5.2.14 *Use Case Diagram* Iterasi 1

Use case diagram ini merupakan perubahan dari *use case diagram* iterasi 0 pada fase *stabilize*. *Use case diagram* untuk ini adalah hasil dari iterasi 1 yang digambarkan pada Gambar 5.53 dan Gambar 5.54.



Gambar 5.53 *Use Case Diagram* Iterasi 1 dengan Aktor Mahasiswa



Gambar 5.54 *Use Case Diagram* Iterasi 1 dengan Aktor Dosen

Use case diagram pada Gambar 5.53 dan Gambar 5.54 menunjukkan hasil iterasi 1, dimana terdapat penambahan *use case reset password, use case* melihat data mahasiswa, *use case* melihat skor dan *use case* melihat skor mahasiswa. Penambahakan *use case reset password* dilakukan karena diperlukan fungsi jika *user* lupa *password*. Penambahan *use case* lihat skor ditambahkan karena terdapat penambahan fungsi pada *use case* latihan. Sedangkan penambahan *use case me*lihat data mahasiswa dilakukan untuk melihat data mahasiswa yang telah terdaftar dalam sistem. Penambahan *use case me*lihat skor mahasiswa dilakukan untuk melihat skor mahasiswa yang telah mengerjakan latihan yang tersedia di aplikasi *BelajarJava.*

5.2.15 *Use Case Scenario* Iterasi 1

Terdapat perubahan *use case* pada *use case* dengan aktor *guest*, mahasiswa, maupun dosen. Pada *use case guest*, terdapat penambahan *use case reset password*. Pada *use case* mahasiswa, terdapat perubahan pada fitur *use case* yang berhubungan dengan materi singkat dan contoh *source code*, dimana *use case* materi singkat berubah nama menjadi materi, sedangkan terjadi perubahan *extend* pada kedua fitur dimana untuk mengakses contoh *source code* bisa dilakukan ketika melakukan *use case* melihat materi, serta penambahan *use case* lihat skor. Pada *use case* dosen, terjadi penambahan *use case* melihat data mahasiswa dan melihat skor mahasiswa.

5.2.15.1 *Use Case Scenario* Guest Iterasi 1

Pada *use case scenario guest*, hanya dijelaskan *use case reset password* saja karena *use case* yang lain tidak terdapat perubahan.

Use case scenario pada Tabel 5.46 menjelaskan alur yang terjadi pada *use case reset password*. Dalam *use case scenario reset password*, aktor yang dilibatkan adalah *guest*. Hasil yang akan didapatkan setelah menjalani alur adalah aktor menerima sebuah *e-mail* untuk mendapatkan *password* yang baru, lalu terdapat alur alternatif jika terdapat proses yang gagal dalam melakukan alur *reset password*.

Tabel 5.46 *Use Case Scenario Reset Password*

Nama *Use Case*	*Reset Password*
Aktor	*Guest*
Deskripsi	Pengguna yang lupa *password* melakukan fungsi *reset password* untuk mendapatkan *password* yang baru
Pra-Kondisi	Aktor memilih menu lupa *password*
Alur	1. Sistem menampilkan form berupa *e-mail* 2. Aktor mengisi form *e-mail* 3. Sistem mengirimkan *e-mail* kepada pengguna
Post-Kondisi	*E-mail* untuk mendapatkan *password* yang baru dikirimkan ke pengguna

Alternatif	2a. *E-mail* belum terdaftar Sistem menampilkan pesan error

5.2.15.2 *Use Case Scenario* Mahasiswa Iterasi 1

Pada *use case scenario guest*, dijelaskan *use case* terkait fitur materi, contoh *source code* dan fitur latihan yang terdapat perubahan serta akan dijelaskan *use case* baru yaitu melihat skor. Walaupun tidak terdapat perubahan kebutuhan pada *use case* dengan fitur materi maupun *use case* dengan fitur contoh *source code*, tetapi terdapat perubahan alur sehingga dilakukan *update* pada dokumen *use case scenario*.

Use case scenario pada Tabel 5.47 menjelaskan alur yang terjadi pada *use case* lihat materi. Dalam *use case scenario* lihat materi, aktor yang dilibatkan adalah mahasiswa. Hasil yang akan didapatkan setelah menjalani alur adalah sistem yang menampilkan materi tentang pemrograman Java, lalu tidak terdapat alur alternatif dari alur yang terjadi.

Tabel 5.47 *Use Case Scenario* Melihat Materi

Nama Use Case	Melihat materi
Aktor	Mahasiswa
Deskripsi	Aktor mempelajari materi seputar pemrograman Java
Pra-Kondisi	Aktor login sebagai mahasiswa
Alur	1. Sistem menampilkan halaman beranda atau halaman terakhir yang dibuka oleh pengguna 2. Aktor memilih menu materi pada tab menu 3. Aktor memilih materi apa yang diinginkan
Post-Kondisi	Sistem menampilkan konten materi seputar pemrograman Java
Alternatif	-

Use case scenario pada Tabel 5.48 menjelaskan alur yang terjadi pada *use case* lihat contoh *source code*. Dalam *use case scenario* lihat contoh *source code*, aktor yang dilibatkan adalah mahasiswa. Hasil yang akan didapatkan setelah menjalani alur adalah sistem yang menampilkan contoh *source code* tentang pemrograman Java, lalu tidak terdapat alur alternatif dari alur yang terjadi.

Tabel 5.48 *Use Case Scenario* Melihat Contoh *Source Code*

Nama Use Case	Melihat contoh *source code*
Aktor	Mahasiswa
Deskripsi	Aktor mempelajari *source code* seputar pemrograman Java

Pra-Kondisi	Aktor telah login sebagai mahasiswa
Alur	1. Sistem menampilkan halaman beranda atau halaman terakhir yang dibuka oleh pengguna 2. Aktor memilih menu materi pada tab menu 3. Aktor memilih materi apa yang diinginkan 4. Aktor melakukan *swipe* ke bagian *source code*
Post-Kondisi	Sistem menampilkan contoh *source code* Bahasa Java
Alternatif	-

Use case scenario pada Tabel 5.49 menjelaskan alur yang terjadi pada *use case* latihan. Dalam *use case scenario* latihan, aktor yang dilibatkan adalah mahasiswa. Hasil yang akan didapatkan setelah menjalani alur adalah sistem yang menampilkan nilai berdasarkan latihan yang telah dikerjakan, lalu tidak terdapat alur alternatif dari alur yang terjadi.

Tabel 5.49 *Use Case Scenario* Latihan

Nama Use Case	Latihan
Aktor	Mahasiswa
Deskripsi	Aktor menjawab pertanyaan seputar pemrograman Java
Pra-Kondisi	Aktor telah *login* sebagai mahasiswa
Alur	1. Sistem menampilkan halaman beranda atau halaman terakhir yang dibuka oleh pengguna 2. Aktor memilih menu latihan pada tab menu 3. Aktor memilih materi latihan apa yang diinginkan 4. Aktor mengerjakan latihan dari beberapa soal yang tersedia 5. Sistem menampilkan skor latihan
Post-Kondisi	Sistem melakukan *update* terhadap skor latihan
Alternatif	-

Use case scenario pada Tabel 5.50 menjelaskan alur yang terjadi pada *use case* ubah profil. Dalam *use case scenario* lihat skor, aktor yang dilibatkan adalah mahasiswa dan dosen. Hasil yang akan didapatkan setelah menjalani alur adalah sistem menampilkan data skor latihan, lalu tidak terdapat alur alternatif yang terjadi.

Tabel 5.50 *Use Case Scenario* Melihat Skor

Nama Use Case	Melhat skor
Aktor	Mahasiswa

Deskripsi	Fungsi untuk melihat skor latihan
Pra-Kondisi	Pengguna telah *login* sebagai mahasiswa
Alur	1. Aktor memilih menu lihat skor pada tab menu
Post-Kondisi	Sistem menampilkan skor latihan
Alternatif	-

5.2.15.3 *Use Case Scenario* Dosen Iterasi 1

Perubahan pada *use case* dosen terdapat pada fitur contoh *source code* dimana untuk membuka konten *source code* dilakukan dengan cara membuka konten materi terlebih dahulu. Terdapat perubahan alur pada *use case* dengan fitur contoh *source code* sehingga dilakukan *update* pada dokumen *use case diagram* dan use *case scenario*.

Use case scenario pada Tabel 5.51 menjelaskan alur yang terjadi pada *use case* membuat contoh *source code*. Dalam *use case scenario* membuat contoh *source code*, aktor yang dilibatkan adalah dosen. Hasil yang akan didapatkan setelah menjalani alur adalah sistem menampilkan notifikasi bahwa konten contoh *source code* telah dibuat. Terdapat alur alternatif dimana jika terjadi *error* maka sistem akan menampilkan notifikasi.

Tabel 5.51 *Use Case Scenario* Membuat Contoh *Source Code*

Nama Use Case	Membuat contoh *source code*
Aktor	Dosen
Deskripsi	Sistem menyediakan fitur untuk membuat konten contoh *source code*
Pra-Kondisi	Pengguna membuka konten materi
Alur	1. Aktor membuka halaman konten *source code* 2. Aktor memilih menu membuat konten *source code* 3. Isi form 4. Sistem melakukan *update* ke *database*
Post-Kondisi	Sistem menampilkan notifikasi bahwa konten telah dibuat
Alternatif	4a. Jika terjadi *error*, maka sistem akan menampilkan pesan *error*

Use case scenario pada Tabel 5.52 menjelaskan alur yang terjadi pada *use case* ubah contoh *source code*. Dalam *use case scenario* ubah contoh *source code*, aktor yang dilibatkan adalah dosen. Hasil yang akan didapatkan setelah menjalani alur adalah sistem mengubah konten contoh *source code*, dan tidak terdapat alur alternatif.

Tabel 5.52 *Use Case Scenario* Mengubah Contoh *Source Code*

Nama Use Case	Ubah contoh *source code*
Aktor	Dosen
Deskripsi	Sistem menyediakan fitur untuk mengubah konten contoh *source code*
Pra-Kondisi	Pengguna melihat salah satu konten contoh *source code* di konten materi
Alur	1. Aktor memilih menu ubah 2. Dosen mengisi *form* 3. Dosen menekan tombol ubah 4. *Database* contoh *source code* diubah
Post-Kondisi	Konten contoh *source code* diubah
Alternatif	-

Use case scenario pada Tabel 5.53 menjelaskan alur yang terjadi pada *use case* ubah latihan. Dalam *use case scenario* ubah latihan, aktor yang dilibatkan adalah dosen. Hasil yang akan didapatkan setelah menjalani alur adalah sistem mengubah isi latihan yang terdapat pada aplikasi.

Tabel 5.53 *Use Case Scenario* Mengubah Latihan

Nama Use Case	Mengubah Latihan
Aktor	Dosen
Deskripsi	Sistem menyediakan fitur untuk mengubah latihan
Pra-Kondisi	Pengguna melihat salah satu konten latihan
Alur	1. Aktor memilih menu ubah 2. Dosen mengisi *form* 3. Dosen menekan tombol ubah 4. *Database* latihan diubah
Post-Kondisi	Konten latihan diubah
Alternatif	-

Use case scenario pada Tabel 5.54 menjelaskan alur yang terjadi pada *use case* melihat data mahasiswa. Dalam *use case scenario* melihat data mahasiswa, aktor yang dilibatkan adalah mahasiswa. Hasil yang akan didapatkan setelah menjalani alur adalah sistem menampilkan identitas mahasiswa.

Tabel 5.54 *Use Case Scenario* Melihat Data Mahasiswa

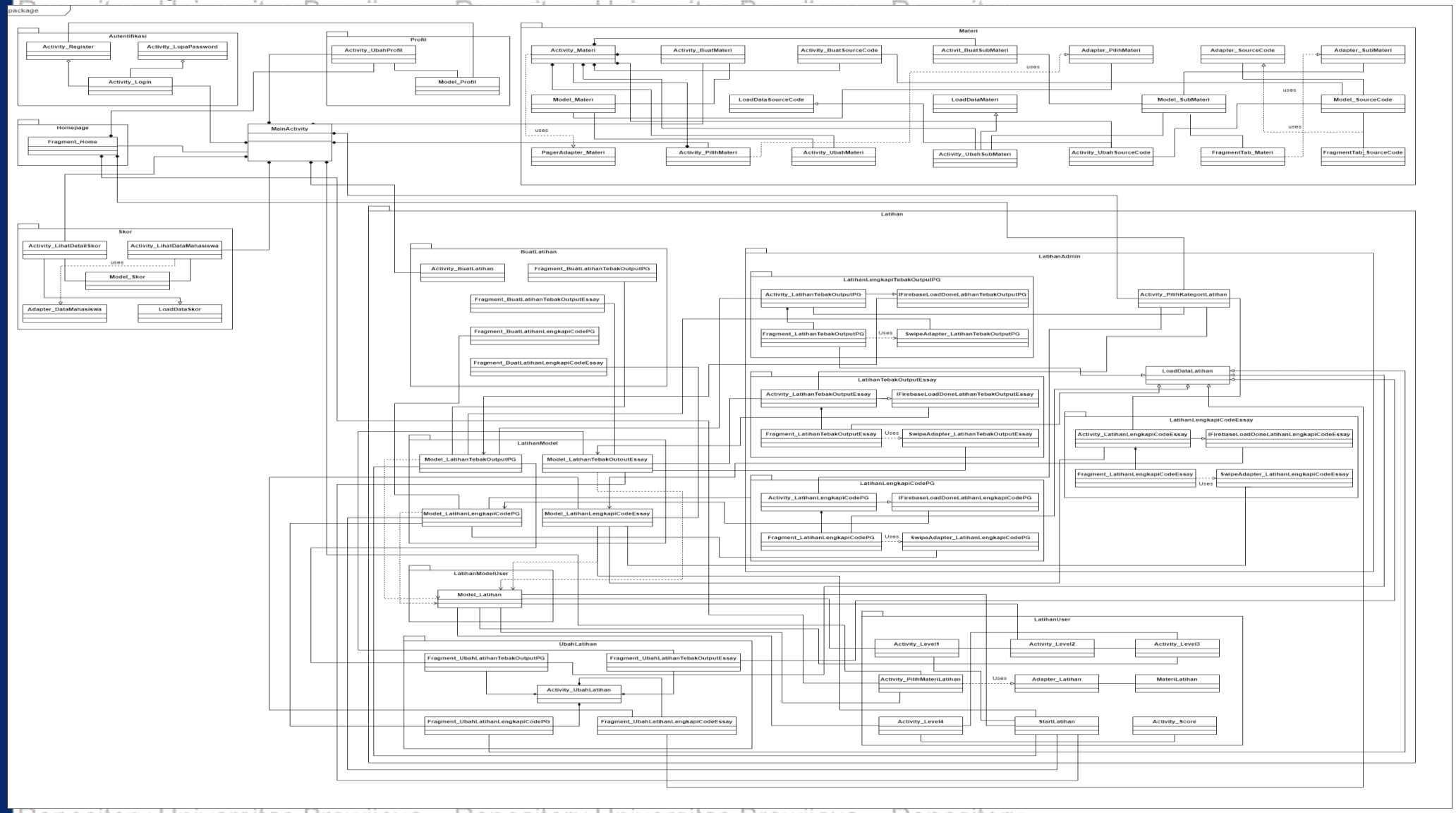
Nama Use Case	Melhat data mahasiswa
Aktor	Dosen
Deskripsi	Fungsi untuk melihat data mahasiswa
Pra-Kondisi	Pengguna telah *login* sebagai dosen
Alur	1. Aktor memilih menu lihat skor pada tab menu
Post-Kondisi	Sistem menampilkan identitas mahasiswa
Alternatif	-

Use case scenario pada Tabel 4.23 menjelaskan alur yang terjadi pada *use case* ubah profil. Dalam *use case scenario* lihat skor, aktor yang dilibatkan adalah mahasiswa dan dosen. Hasil yang akan didapatkan setelah menjalani alur adalah sistem menampilkan data skor latihan, lalu tidak terdapat alur alternatif yang terjadi.

Tabel 5.55 *Use Case Scenario* Melihat Skor Mahasiswa

Nama Use Case	Melhat skor
Aktor	Mahasiswa
Deskripsi	Fungsi untuk melihat skor latihan
Pra-Kondisi	Pengguna meng-klik data mahasiswa yang ingin dilihat data skornya.
Alur	1. Aktor memilih menu lihat skor pada tab menu 2. Aktor memilih mahasiswa yang ingin dilihat skornya.
Post-Kondisi	Sistem menampilkan skor latihan mahasiswa tersebut
Alternatif	-

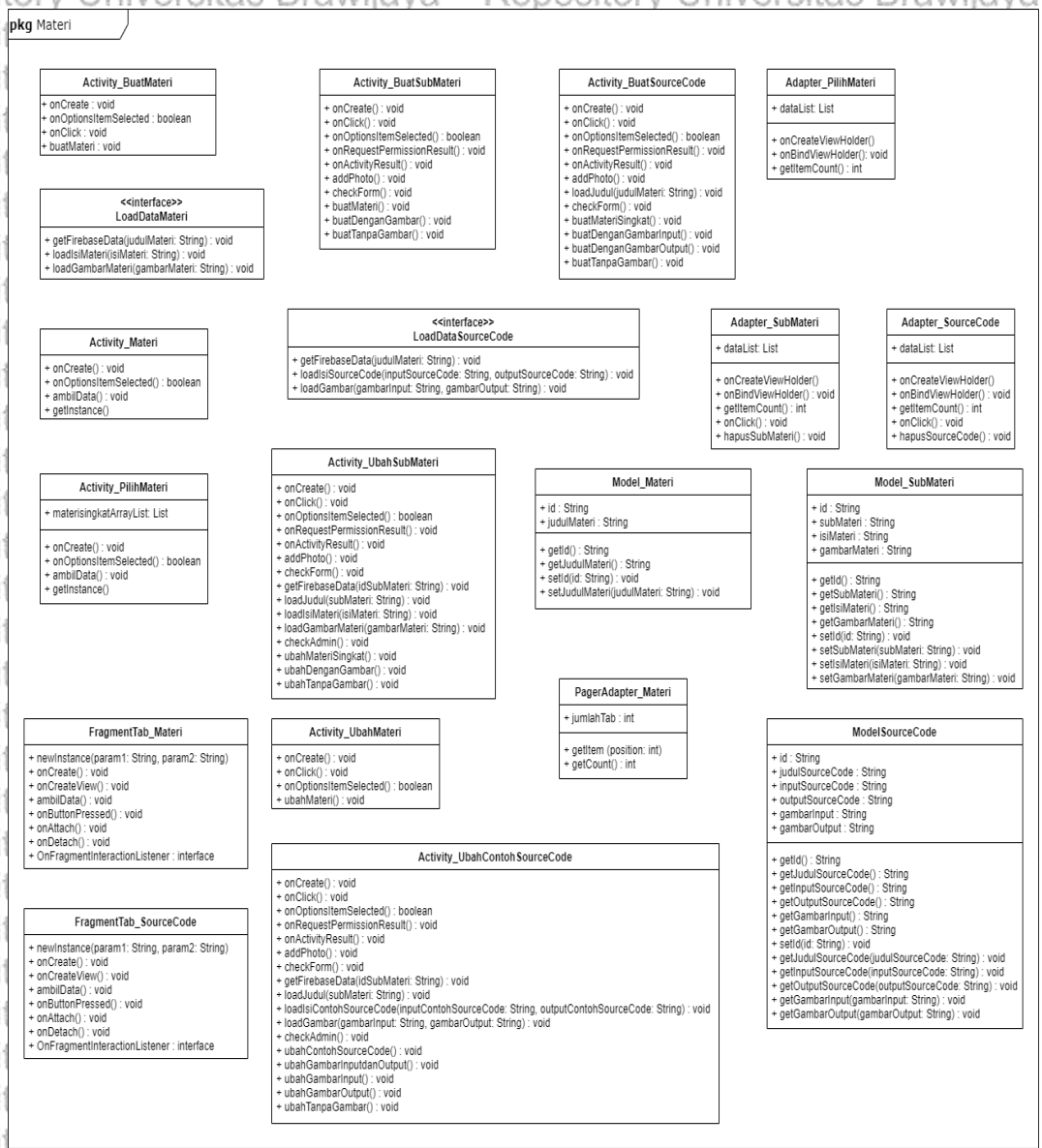
5.2.16 *Class Diagram* Iterasi 1



Gambar 5.55 *Class Diagram* Latihan Iterasi 1

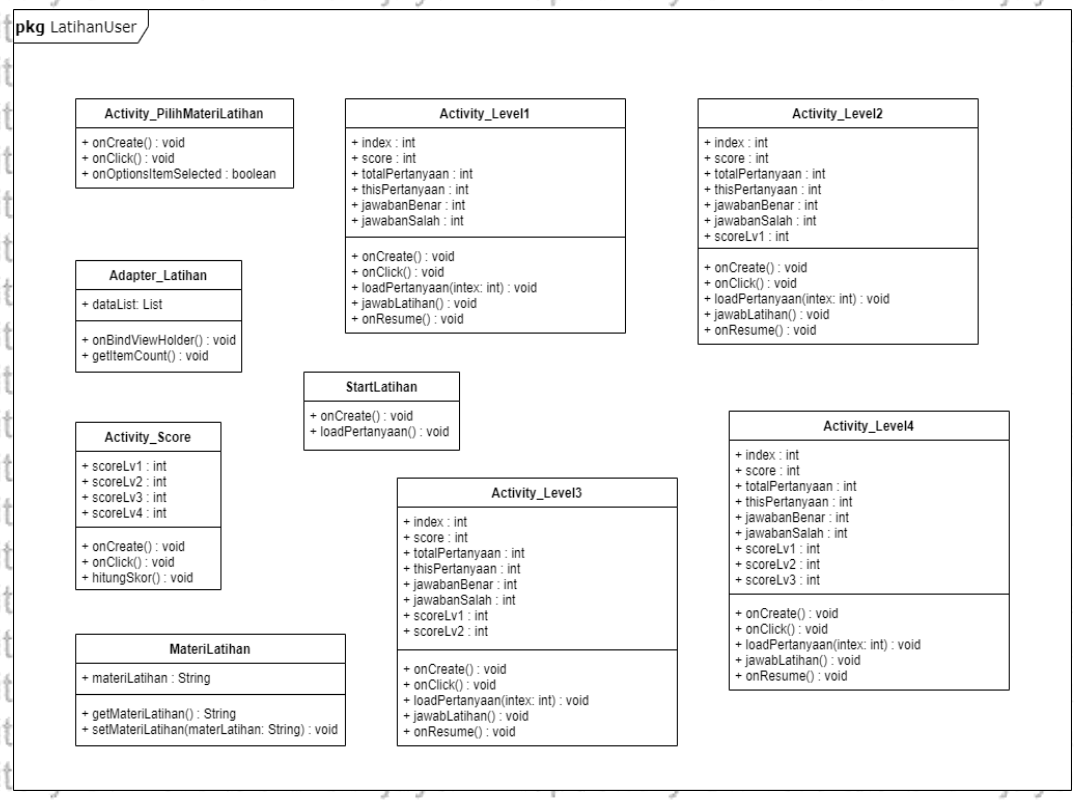
Pada Gambar 5.55 ditunjukkan *class diagram* aplikasi *BelajarJava* hasil dari iterasi 1 secara seluruhan. Terdapat penambahan paket `Skor`, penambahan *class* `Activity_Score` pada *class* `LatihanUser`, dan penggabungan paket `MateriSingkat` dan paket `ContohSourceCode` menjadi `Materi`. Dari Gambar 5.10, sama seperti penjelasan *class diagram* hasil iterasi 0, diambil tiga *package* utama untuk dijelaskan atribut dan *method* pada setiap kelas. Paket utama yang akan dijelaskan adalah paket `Materi`, paket, dan paket `LatihanUser` yang mengalami perubahan atribut, *method*, dan *class*.

Pada paket `Materi` mengalami perubahan signifikan karena ada penggabungan paket `MateriSingkat` dan paket `ContohSourceCode` di iterasi 0. Hasil perubahan paket `Materi` beserta *class*, atribut dan *method*-nya terdapat pada Gambar 5.56.



Gambar 5.56 *Class Diagram* Materi Iterasi 1

Pada paket `LatihanUser` mengalami penambahan *class* `Activity_Score`, dan terjadi penambahan atribut dan *method* pada *class* `Activity_Level1`, *class* `Activity_Level2`, *class* `Activity_Level3`, dan *class* `Activity_Level4`. Perubahan tersebut dikarenakan penambahan *use case* lihat skor, dan lihat skor mahasiswa serta penambahan fungsi untuk melakukan penilaian terhadap hasil latihan yang dikerjakan oleh aktor mahasiswa. Hasil perubahan paket `LatihanUser` beserta *class*, atribut dan *method*-nya terdapat pada Gambar 5.57.



Gambar 5.57 *Class Diagram* Latihan Iterasi 1

5.2.17 *Update* Perancangan Algoritma

Pada subbab ini, terdapat *update* terhadap perancangan algoritma pada *method* yang akan ditambahkan dan akan menjadi *core* pada sistem. Algoritma tersebut yakni cek form dimana *method* ini akan di *reuse* di setiap *class*.

Algoritma cek form merupakan tahapan proses untuk melakukan fungsi cek form dalam kondisi terisi atau tidak. Implementasi algoritma lihat materi singkat dijelaskan pada Gambar 5.58.

```
Algoritma cekForm
Deklarasi :
- Hashmap -> judulMateri, isiMateri

Deskripsi :
  - Masukkan : judulMateri, isiMateri
  - Proses :
      1. Deklarasi variabel judulMateri, isiMateri, gambarMateri
```

```
2. Masukkan kondisi if jika form terdapat yang tidak terisi atau
   tidak :

   a. Jika kondisi if memenuhi, maka akan memanggil method
      buatKonten
   b. Jika tidak memenuhi kondisi if, maka akan menampilkan pesan
      toast bahwa pengguna harus mengisi form terlebih dahulu
```

Gambar 5.58 Algoritma Method Cek Form

5.2.18 *Update* Test Driven Development (TDD)

Pada fase *stabilize*, dilakukan TDD kembali untuk menguji perancangan algoritma `checkForm`, dimana algoritma ini penting karena algoritma ini dipakai di semua fitur dari mulai autentifikasi, membuat konten, sampai menjawab latihan.

Pengujian algoritma `checkForm` dilakukan untuk menguji alur logika *method* `checkForm`. Pengujian ini menggunakan dua skenario dengan kondisi salah dan kondisi benar, dimana kondisi salah saat form kosong, dan kondisi benar saat form tidak kosong.

a. Skenario pengujian

Tabel 5.56 Skenario Pengujian checkForm() Kondisi Salah

Kasus Uji	Cek form kondisi kosong
Tujuan Pengujian	Membuktikan form kosong atau tidak
Prosedur Pengujian	1. Inisialisasi variabel form 2. Cek form apakah terdapat form yang kosong
Hasil yang Diharapkan	Pesan gagal ditampilkan sistem

Tabel 5.57 Skenario Pengujian checkForm() Kondisi Benar

Kasus Uji	Cek form kondisi tidak kosong
Tujuan Pengujian	Membuktikan form kosong atau tidak
Prosedur Pengujian	1. Inisialisasi variabel form 2. Cek form apakah terdapat form yang kosong
Hasil yang Diharapkan	Pesan sukses ditampilkan sistem

b. Implementasi skenario pengujian

Tabel 5.58 Skenario Pengujian checkForm() Kondisi Salah

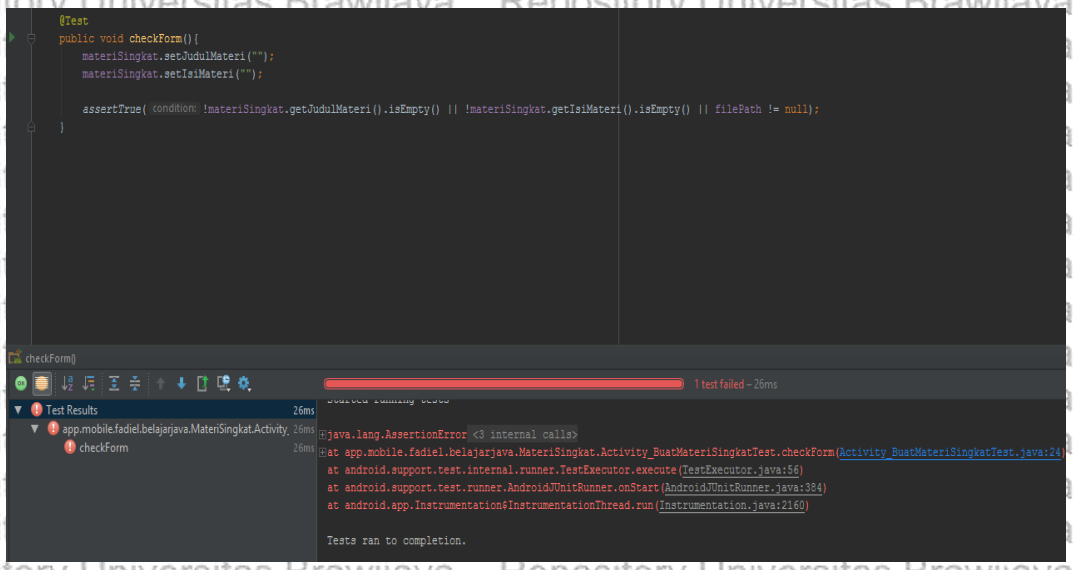
```
1  @Test
2  public void checkForm(){
3      materiSingkat.setJudulMateri("");
4      materiSingkat.setIsiMateri("");
5      assertTrue(!materiSingkat.getJudulMateri().isEmpty() ||
```

```
6   !materiSingkat.getIsiMateri().isEmpty() || filePath != null);
7   }
```

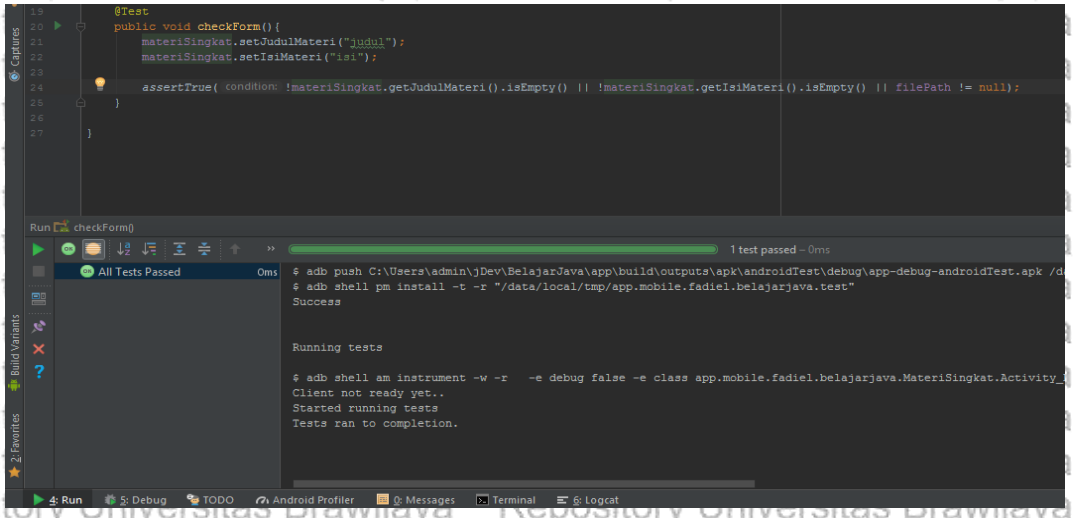
Tabel 5.59 Skenario Pengujian checkForm() Kondisi Benar

```
1   @Test
2   public void checkForm(){
3       materiSingkat.setJudulMateri("judul");
4       materiSingkat.setIsiMateri("isi");
5       assertTrue(!materiSingkat.getJudulMateri().isEmpty() ||
6   !materiSingkat.getIsiMateri().isEmpty() || filePath != null);
7   }
```

c. Hasil pengujian



Gambar 5.59 Hasil Pengujian checkForm Kondisi Salah



Gambar 5.60 Hasil Pengujian checkForm Kondisi Benar

5.2.19 *Update* Kelas

Perancangan Kelas yang telah dirancang sebelumnya, kemudian diimplementasikan pada program dengan ekstensi .java. Implementasi kelas autentifikasi ditunjukkan oleh Tabel 5.60 sampai Tabel 5.65.

Tabel 5.60 Implementasi Kelas Autentifikasi pada Aplikasi *BelajarJava*

No.	*Package*	Kelas atau *Interface*
1	Autentifikasi	Activity_Login
2	Autentifikasi	Activity_Register
3	Autentifikasi	Activity_LupaPassword

Tabel 5.61 Implementasi Kelas Materi Singkat pada Aplikasi *BelajarJava*

No.	*Package*	Kelas atau *Interface*
1	Materi	Activity_BuatMateri
2	Materi	Activity_BuatSourceCode
3	Materi	Activity_BuatSubMateri
4	Materi	Activity_Materi
5	Materi	Activity_PilihMateri
6	Materi	Activity_UbahMateri
7	Materi	Activity_UbahSourceCode
8	Materi	Activity_UbahSubMateri
9	Materi	Adapter_PIlihMateri
10	Materi	Adapter_SourceCode
11	Materi	Adapter_SubMateri
12	Materi	FragmentTab_Materi
13	Materi	FragmentTab_SourceCode
14	Materi	LoadDataMateri
15	Materi	LoadDataSourceCode
16	Materi	Model_Materi
17	Materi	Model_SourceCode
18	Materi	Model_SubMateri
19	Materi	PageAdapter_Materi

Tabel 5.62 Implementasi Kelas Latihan pada Aplikasi *BelajarJava*

No.	*Package*	Kelas atau *Interface*
1	BuatLatihan	Activity_BuatLatihan
2	BuatLatihan	Fragment_BuatLatihanLengkapiCodePG
3	BuatLatihan	Fragment_BuatLatihanLengkapiCodeEssay
4	BuatLatihan	Fragment_BuatLatihanTebakOutputPG
5	BuatLatihan	Fragment_BuatLatihanTebakOutputEssay
6	UbahLatihan	Activity_UbahLatihan
7	UbahLatihan	Fragment_UbahLatihanLengkapiCodePG
8	UbahLatihan	Fragment_UbahLatihanLengkapiCodeEssay
9	UbahLatihan	Fragment_UbahLatihanTebakOutputPG
10	UbahLatihan	Fragment_UbahLatihanTebakOutputEssay
11	LatihanModelUser	Model_Latihan
12	LatihanUser	Activity_Level1
13	LatihanUser	Activity_Level2
14	LatihanUser	Activity_Level3
15	LatihanUser	Activity_Level4
16	LatihanUser	Activity_PilihMateriLatihan
17	LatihanUser	Activity_Score
18	LatihanUser	Adapter_Latihan
19	LatihanUser	MateriLatihan
20	LatihanUser	StartLatihan
21	LatihanAdmin	LoadDataLatihan
22	LatihanAdmin	Activity_PilihKategoriLatihan
23	LatihanModel	Model_LatihanTebakOutputPG
24	LatihanModel	Model_LatihanTebakOutputEssay
25	LatihanModel	Model_LatihanLengkapiCodePG
26	LatihanModel	Model_ LatihanLengkapiCodeEssay
27	LatihanModelUser	Model_Latihan
28	LatihanLengkapiCodeEssay	Activity_LatihanLengkapiCodeEssay
29	LatihanLengkapiCodeEssay	Fragment_LatihanLengkapiCodeEssay

30	LatihanLengkapiCodeEssay	IFirebaseLoadDoneLatihanLengkapiCodeEssay
31	LatihanLengkapiCodeEssay	SwipeAdapter_LatihanLengkapiCodeEssay
32	LatihanLengkapiCodePG	Activity_LatihanLengkapiCodePG
33	LatihanLengkapiCodePG	Fragment_LatihanLengkapiCodePG
34	LatihanLengkapiCodePG	IFirebaseLoadDoneLatihanLengkapiCodePG
35	LatihanLengkapiCodePG	SwipeAdapter_LatihanLengkapiCodePG
36	LatihanTebakOutputPG	Activity_LatihanTebakOutputPG
37	LatihanTebakOutputPG	Fragment_LatihanTebakOutputPG
38	LatihanTebakOutputPG	IFirebaseLoadDoneLatihanTebakOutputPG
39	LatihanTebakOutputPG	SwipeAdapter_LatihanTebakOutputPG
40	LatihanTebakOutputEssay	Activity_LatihanTebakOutputEssay
41	LatihanTebakOutputEssay	Fragment_LatihanTebakOutputEssay
42	LatihanTebakOutputEssay	IFirebaseLoadDoneLatihanTebakOutputEssay
43	LatihanTebakOutputEssay	SwipeAdapter_LatihanTebakOutputEssay

Tabel 5.63 Implementasi Kelas Homepage pada Aplikasi *BelajarJava*

No.	*Package*	Kelas atau *Interface*
1	Homepage	Fragment_Home
2	Homepage	MainActivity

Tabel 5.64 Implementasi Kelas Profil pada Aplikasi *BelajarJava*

No.	*Package*	Kelas atau *Interface*
1	Profil	EditProfileActivity
2	Profil	UserInfo

Tabel 5.65 Implementasi Kelas Skor pada Aplikasi *BelajarJava*

No.	*Package*	Kelas atau *Interface*
1	Skor	Activity_LihatDataMahasiswa
2	Skor	Activity_LihatDetailSkor
3	Skor	Adapter_DataMahasiswa

4	Skor	LoadDataSkor
5	Skor	Model_Skor

5.2.20 *Update* Kode Program

Sub bab ini menjelaskan hasil implementasi kode aplikasi *BelajarJava* yang telah dikembangkan. Pada bagian ini hanya dijelaskan *method* yang memiliki peran penting pada aplikasi. *Method* yang dijelaskan juga dilakukan *reuse* oleh pengembang untuk diimplementasikan pada kelas yang lain.

5.2.20.1 *Update Method* checkForm()

Method `buatMateri` merupakan *method* untuk menambahkan isi konten pada halaman materi singkat. Pada *Method* ini variabel diakses menggunakan *Model class* yang akan disimpan di *database Firebase*. *Model class* yang diakses pada *method* ini adalah *class* `Model_Materi`. *Method* ini juga berfungsi untuk memanggil salah satu *method* dari dua method yang bisa dipanggil, yakni *method* `buatDenganGambar` atau `buatTanpaGambar`.

Tabel 5.66 *Update Method* checkForm()

```
1   subMateri.setSubMateri(etSubMateri.getText().toString());
2   subMateri.setIsiMateri(etIsiMateriSingkat.getText().toString());
3           if(subMateri.getSubMateri().trim().length() > 0){
4               if(filePath!=null ||
5   subMateri.getIsiMateri().trim().length() > 0 ||
6   subMateri.getGambarMateri().trim().length() > 0){
7                   ubahMateriSingkat();
8               } else {
9                   Toast.makeText(getBaseContext(), "Isi atau gambar
10  kosong..", Toast.LENGTH_SHORT).show();
11              }
12          } else {
13              Toast.makeText(getBaseContext(), "Judul konten
14  kosong.", Toast.LENGTH_SHORT).show();
15          }
```

Tabel 5.67 Penjelasan *Update Method* checkForm()

Nomor Kode Program	Penjelasan
1	Set *value* variabel subMateri
2	Set *value* variabel isiMateri
3	Kondisi if jika variabel subMateri tidak kosong
4-6	Kondisi if jika *attachment file* kosong, variabel isiMateri kosong atau variabel gambarMateriKosong
7	Memanggil *method* ubahMateriSingkat()

8-11	Kondisi else jika *attachment file* kosong, variabel isiMateri kosong atau variabel gambarMateriKosong terpenuhi
12-14	Kondisi else jika variabel subMateri tidak kosong
15	Kurung penutup dari *method* checkForm

5.2.20.2 *Update Method* buatMateri()

Method `buatMateri` di *update* dikarenakan terdapat perubahan pada alur. Nama method ini awalnya adalah *method* `buatMateriSingkat` dimana *method* tersebut kemudian di *reuse* dan dilakukan perubahan. *Method* `buatMateri` merupakan *method* untuk menambahkan isi konten pada halaman materi singkat. Pada *Method* ini variabel diakses menggunakan *Model class* yang akan disimpan di *database Firebase*. *Model class* yang diakses pada *method* ini adalah *class* `Model_Materi`. *Method* ini juga berfungsi untuk memanggil salah satu *method* dari dua method yang bisa dipanggil, yakni *method* `buatDenganGambar` atau `buatTanpaGambar`.

Tabel 5.68 *Update Method* buatMateri()

```
1   private void buatMateri() {
2           databaseReference =
3   FirebaseDatabase.getInstance().getReference();
4           DatabaseReference postsRef =
5   databaseReference.child("materi").child(materi.getId());
6           DatabaseReference newPostRef =
7   postsRef.child("sub_materi").push();
8           subMateri.setId(newPostRef.getKey());
9           storageReference =
10  FirebaseStorage.getInstance().getReference();
11  StorageReference storageRef = storageReference.child("subMateri/"+
12  UUID.randomUUID().toString());
13          final ProgressDialog progressDialog = new
14  ProgressDialog(Activity_BuatSubMateri.this);
15          progressDialog.setTitle("Menambah Konten");
16          progressDialog.show();
17          if(filePath != null){
18                  buatDenganGambar(storageRef, progressDialog,
19  newPostRef);
20          } else {
21                  buatTanpaGambar(progressDialog, newPostRef);
22          }
23      }
```

Tabel 5.69 Penjelasan *Update Method* buatMateri()

Nomor Kode Program	Penjelasan
1	Inisialisasi *method* buatMateri
2-8	Inisiliasi *database Firebase*
9-12	Inisialisasi *Firebase Storage* untuk menyimpan *file* berupa gambar
13-16	Memanggil *class* ProgressDialog untuk menambahkan dialog saat melakukan *upload* gambar
17-19	Kondisi if jika variabel filePath tidak bernilai null akan memanggil *method* buatDenganGambar
20-22	Kondisi else jika variabel filePath bernilai null akan memanggil *method* buatTanpaGambar
23	Kurung penutup dari *method* buatMateri

5.2.20.3 *Update Method* buatSourceCode()

Method `buatSourceCode` di *update* dikarenakan terdapat perubahan pada alur. Nama method ini awalnya adalah *method* `buatContohSourceCode` dimana *method* tersebut kemudian di *reuse* dan dilakukan perubahan. *Method* `buatSourceCode` merupakan *method* untuk menambahkan isi konten pada halaman *source code*. Pada *Method* ini variabel diakses menggunakan *model class* yang akan disimpan di *database Firebase*. *Model class* yang diakses pada *method* ini adalah *class* `Model_SourceCode`. *Method* ini juga berfungsi untuk memanggil salah satu *method* dari dua method yang bisa dipanggil, yakni *method* `buatDenganGambar atau buatTanpaGambar`.

Tabel 5.70 *Update Method* buatSourceCode()

```
1   private void buatSourceCode(){
2           databaseReference =
3   FirebaseDatabase.getInstance().getReference();
4           final DatabaseReference postsRef =
5   databaseReference.child("materi").child(materi.getId());
6           final DatabaseReference newPostsRef =
7   postsRef.child("source_code").push();
8           newPostsRef.keepSynced(true);
9           sourceCode.setId(newPostsRef.getKey());
10          sourceCode.setGambarInput("");
11          sourceCode.setGambarOutput("");
12          storageReference =
13  FirebaseStorage.getInstance().getReference();
14          StorageReference storageRef1 =
15  storageReference.child("source_code/"+
16  UUID.randomUUID().toString());
```

```
17          StorageReference storageRef2 =
18  storageReference.child("source_code/"+
19  UUID.randomUUID().toString());
20          final ProgressDialog progressDialog = new
21  ProgressDialog(Activity_BuatSourceCode.this);
22          progressDialog.setTitle("Menambah Konten");
23          progressDialog.show();
24          if(filePathInput != null && filePathOutput != null){
25              buatDenganGambarInputdanOutput(storageRef1,
26  storageRef2, postsRef, newPostsRef, progressDialog);
27          } else if (filePathInput != null || filePathOutput !=
28  null) {
29              if(filePathInput != null) {
30                  buatDenganGambarInput(storageRef1, newPostsRef,
31  progressDialog);
32              } else if (filePathOutput != null) {
33                  buatDenganGambarOutput(storageRef1, newPostsRef,
34  progressDialog);
35              }
36          } else {
37              buatTanpaGambar(newPostsRef, progressDialog);
38          }
39      }
```

Tabel 5.71 Penjelasan *Update Method* buatSourceCode()

Nomor Kode Program	Penjelasan
1	Inisialisasi *method* buatSourceCode
2-8	Inisialisasi *database Firebase*
9	Set nilai variabel id dari *class* Model_SourceCode
10	Set nilai variabel gambarInput dari *class* Model_SourceCode
11	Set nilai variabel gambarOutput dari *class* Model_SourceCode
12-19	Inisialisasi *Firebase Storage* untuk menyimpan *file* berupa gambar
20-23	Memanggil *class* ProgressDialog untuk menambahkan dialog saat melakukan *upload* gambar
24-26	Kondisi if jika variabel filePathInput tidak bernilai null dan filePathOutput tidak bernilai null akan memanggil *method* buatDenganGambarInputdanOutput
27-31	Kondisi if jika variabel filePathInput tidak bernilai null akan memanggil *method* buatDenganGambarInput
32-35	Kondisi if jika variabel filePathOutput tidak bernilai null akan memanggil *method* buatDenganGambarOutput

36-38	Kondisi else jika variabel filePathInput bernilai null dan filePathOutput bernilai null akan memanggil *method* buatTanpaGambar
39	Kurung penutup dari *method* buatSourceCode

5.2.20.4 *Update Method* ubahMateri()

Method `ubahMateri` di *update* dikarenakan terdapat perubahan pada alur. Nama method ini awalnya adalah *method* `ubahMateriSingkat` dimana *method* tersebut kemudian di *reuse* dan dilakukan perubahan. *Method* `ubahMateri` merupakan *method* untuk mengubah isi konten pada halaman materi singkat. Hampir sama pada *method* `buatMateri`, perbedaan pada *method* ini yakni tidak mendeklarasikan variabel `FirebaseStorage`, melainkan variabel `DatabaseReference` yang mewakili lokasi dari *database* materi singkat.

Tabel 5.72 *Update Method* ubahMateri()

```
1   private void buatMateri() {
2           databaseReference =
3   FirebaseDatabase.getInstance().getReference();
4           DatabaseReference postsRef =
5   databaseReference.child("materi").child(materi.getId());
6           DatabaseReference newPostRef =
7   postsRef.child("sub_materi").push();
8           subMateri.setId(newPostRef.getKey());
9           storageReference =
10  FirebaseStorage.getInstance().getReference();
11          StorageReference storageRef =
12  storageReference.child("subMateri/"+
13  UUID.randomUUID().toString());
14          final ProgressDialog progressDialog = new
15  ProgressDialog(Activity_BuatSubMateri.this);
16          progressDialog.setTitle("Menambah Konten");
17          progressDialog.show();
18          if(filePath != null){
19                  buatDenganGambar(storageRef, progressDialog,
20  newPostRef);
21          } else {
22                  buatTanpaGambar(progressDialog, newPostRef);
23          }
24      }
```

Tabel 5.73 Penjelasan *Update Method* ubahMateri

Nomor Kode Program	Penjelasan
1	Inisialisasi *method* ubahMateri
2-7	Inisiliasi *database Firebase*
8	Set nilai variabel id dari *class* Model_SubMateri
9-13	Inisialisasi data *storage Firebase*
14-17	Memanggil *class* ProgressDialog untuk menambahkan dialog
18-20	Kondisi if jika variabel filePath tidak bernilai null akan memanggil *method* ubahDenganGambar
21-23	Kondisi else jika variabel filePath bernilai null akan memanggil *method* ubahTanpaGambar
24	Penutup *method* buatMateri

5.2.20.5 *Update Method* ubahSourceCode()

Method `ubahSourceCode` di *update* dikarenakan terdapat perubahan pada alur. Nama method ini awalnya adalah *method* `buatSourceCode` dimana *method* tersebut kemudian di *reuse* dan dilakukan perubahan. *Method* `ubahSourceCode` merupakan *method* untuk mengubah isi konten pada halaman materi singkat. Hampir sama pada *method* `buatMateri`, perbedaan pada *method* ini yakni tidak mendeklarasikan variabel `FirebaseStorage`, melainkan variabel `DatabaseReference` yang mewakili lokasi dari *database* materi.

Tabel 5.74 Implementasi *Update Method* ubahSourceCode()

```
1  private void buatSourceCode(){
2          databaseReference =
3  FirebaseDatabase.getInstance().getReference();
4          final DatabaseReference postsRef =
5  databaseReference.child("materi").child(materi.getId());
6          final DatabaseReference newPostsRef =
7  postsRef.child("source_code").push();
8          newPostsRef.keepSynced(true);
9          sourceCode.setId(newPostsRef.getKey());
10         sourceCode.setGambarInput("");
11         sourceCode.setGambarOutput("");
12         storageReference =
13 FirebaseStorage.getInstance().getReference();
14         StorageReference storageRef1 =
15 storageReference.child("source_code/"+
16 UUID.randomUUID().toString());
17         StorageReference storageRef2 =
18 storageReference.child("source_code/"+
19 UUID.randomUUID().toString());
```

```
20          final ProgressDialog progressDialog = new
21  ProgressDialog(Activity_BuatSourceCode.this);
22          progressDialog.setTitle("Menambah Konten");
23          progressDialog.show();
24          if(filePathInput != null && filePathOutput != null){
25              buatDenganGambarInputdanOutput(storageRef1,
26  storageRef2, postsRef, newPostsRef, progressDialog);
27          } else if (filePathInput != null || filePathOutput !=
28  null) {
29              if(filePathInput != null) {
30                  buatDenganGambarInput(storageRef1, newPostsRef,
31  progressDialog);
32              } else if (filePathOutput != null) {
33                  buatDenganGambarOutput(storageRef1, newPostsRef,
34  progressDialog);
35              }
36          } else {
37              buatTanpaGambar(newPostsRef, progressDialog);
38          }
39      }
```

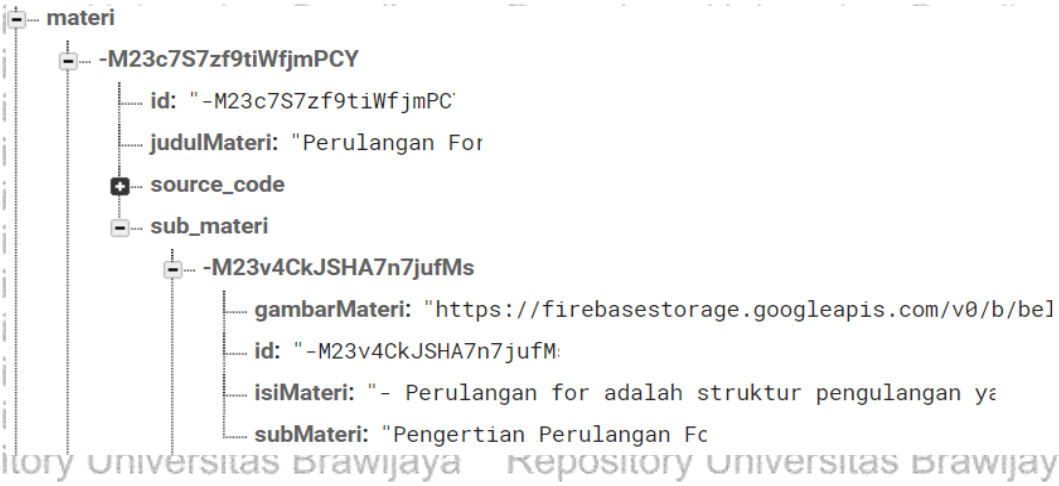
Tabel 5.75 Penjelasan *Update Method* ubahSourceCode()

Nomor Kode Program	Penjelasan
1	Inisialisasi *method* ubahContohSourceCode
2-8	Inisialisasi *database Firebase*
9	Set nilai variabel id dari *class* Model_ContohSourceCode
10	Set nilai variabel gambarInput dari *class* Model_ContohSourceCode
11	Set nilai variabel gambarOutput dari *class* Model_ContohSourceCode
12-19	Inisialisasi data *storage Firebase*
20-23	Memanggil *class* ProgressDialog untuk menambahkan dialog
24-26	Kondisi if jika variabel filePathInput tidak bernilai null dan filePathOutput tidak bernilai null akan memanggil *method* ubahDenganGambarInputdanOutput
27-31	Kondisi if jika variabel filePathInput tidak bernilai null akan memanggil *method* ubahDenganGambarInput
32-35	Kondisi if jika variabel filePathOutput tidak bernilai null akan memanggil *method* ubahDenganGambarOutput
36-38	Kondisi else jika variabel filePathInput bernilai null dan filePathOutput bernilai null akan memanggil *method* ubahTanpaGambar

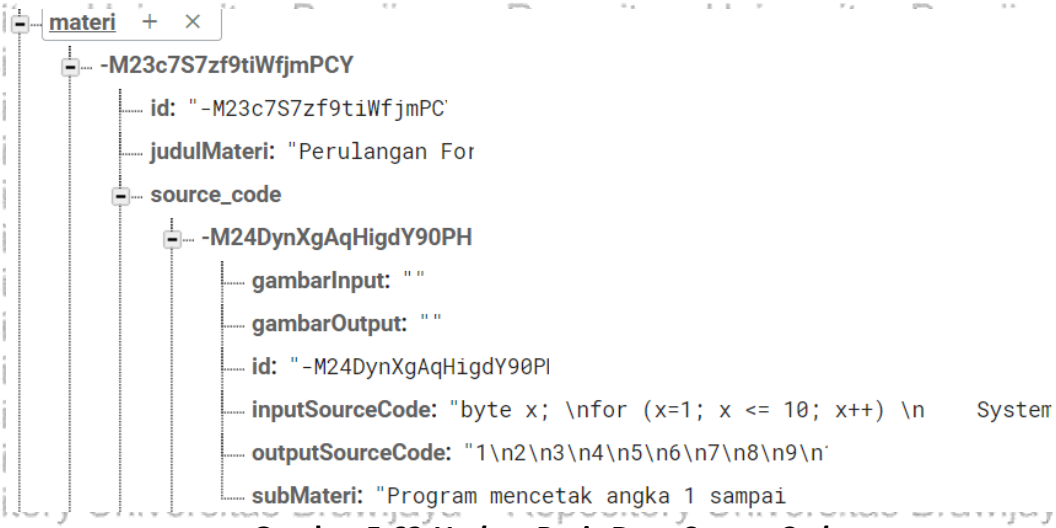
39	Penutup *method* ubahSourceCode

5.2.21 *Update* Basis Data

Implementasi basis data pada aplikasi *BelajarJava* menggunakan *firebase realtime database*. Hasil implementasi basis data pada aplikasi *BelajarJava* menggunakan *script* dalam bentuk *JSON*. Implementasi basis data pada aplikasi *BelajarJava* ditunjukkan dalam Gambar 5.61 sampai Gambar 5.65 :

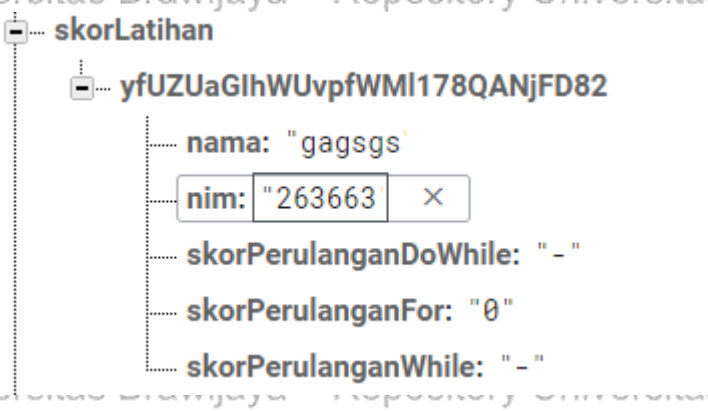


Gambar 5.61 *Update* Basis Data Materi



Gambar 5.62 *Update* Basis Data *Source Code*

Gambar 5.63 *Update* Basis Data Latihan



Gambar 5.64 *Update* Basis Data Skor

Gambar 5.65 *Update* Basis Data Profil

5.2.22 *Update* Antarmuka

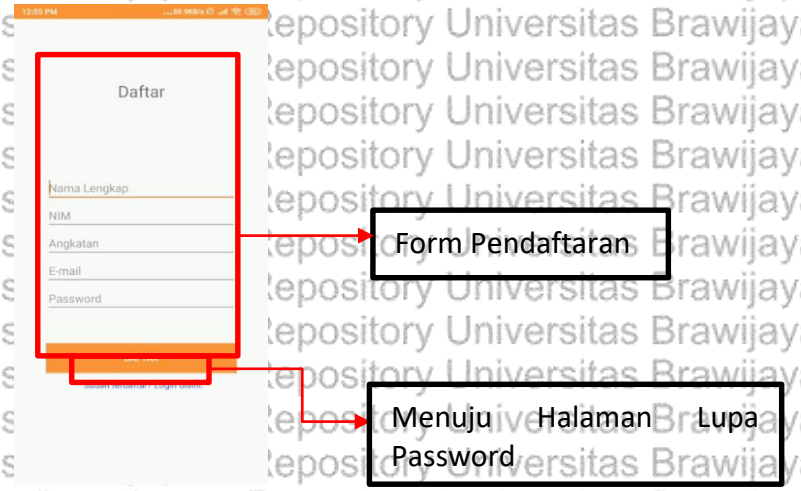
Subbab ini menjelaskan hasil tampilan aplikasi *BelajarJava* yang dikembangkan hasil dari iterasi 1 pengguna. Antarmuka terdapat perubahan khususnya pada tampilan *background* yang bertujuan untuk mempercantik tampilan antarmuka dari aplikasi yang dikembangkan

5.2.22.1 *Update Antarmuka Autentifikasi*

Pada subbab ini menjelaskan hasil tampilan halaman autentifikasi yang terdiri dari *register*, *login*, dan *reset password* pada aplikasi *BelajarJava*.

a. Halaman *Register*

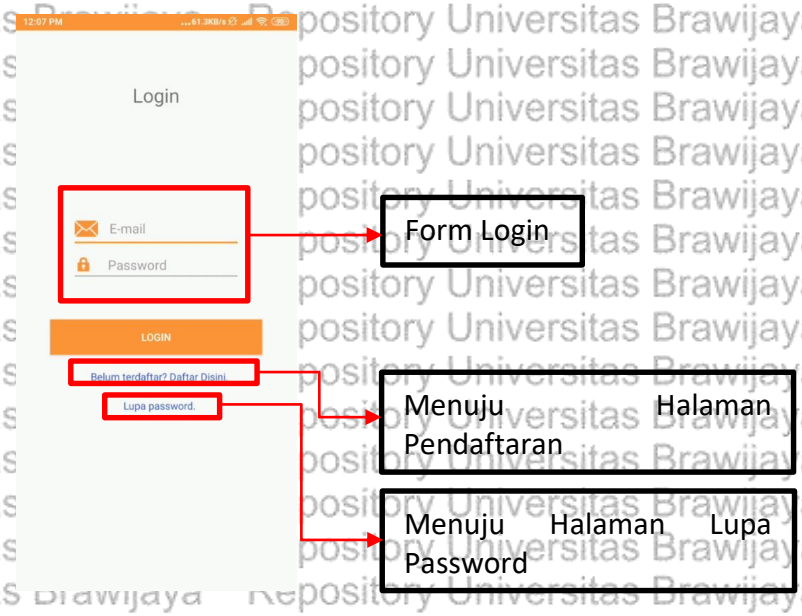
Halaman *register* merupakan antarmuka untuk daftar pengguna baru. Antarmuka halaman *register* ditunjukkan oleh Gambar 5.66 berikut :



Gambar 5.66 *Update* Antarmuka Halaman *Register*

b. Halaman *Login*

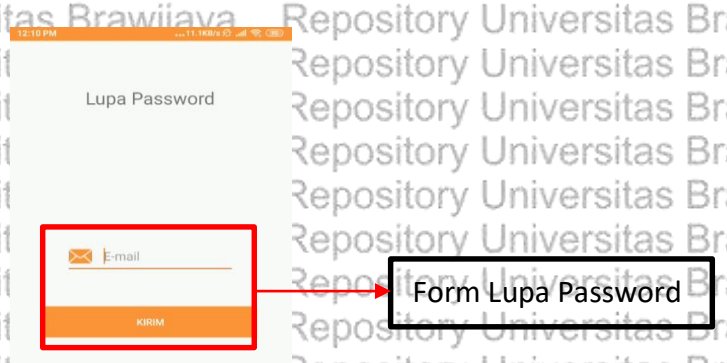
Halaman *login* merupakan halaman untuk melakukan autentifikasi. Antarmuka halaman *login* ditunjukkan oleh Gambar 5.67 berikut :



Gambar 5.67 *Update* Antarmuka Halaman *Login*

c. Halaman *Reset Password*

Halaman *reset password* merupakan antarmuka bagi pengguna yang melupakan *password* untuk mendapatkan kembali *password* yang baru. Antarmuka halaman *reset password* ditunjukkan oleh Gambar 5.68 berikut :



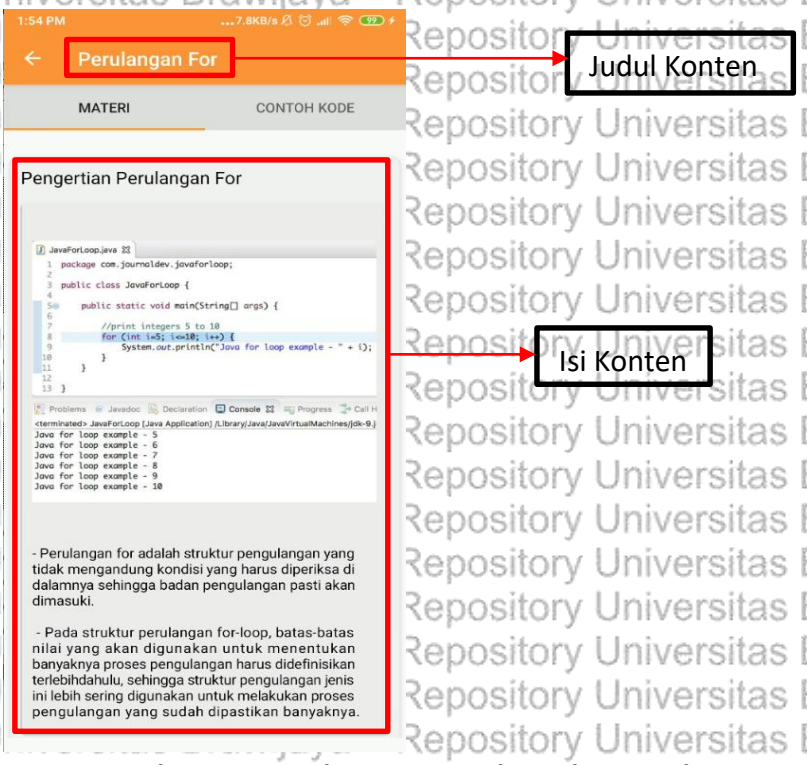
Gambar 5.68 *Update* Antarmuka Halaman *Reset Password*

5.2.22.2 *Update* Antarmuka Materi

Pada subbab ini menjelaskan hasil tampilan fitur materi pada aplikasi *BelajarJava*. Fitur materi pada aplikasi *BelajarJava* terdiri dari 3 bagian, yakni buat materi, ubah materi, dan lihat materi singkat.

a. Halaman Lihat Materi

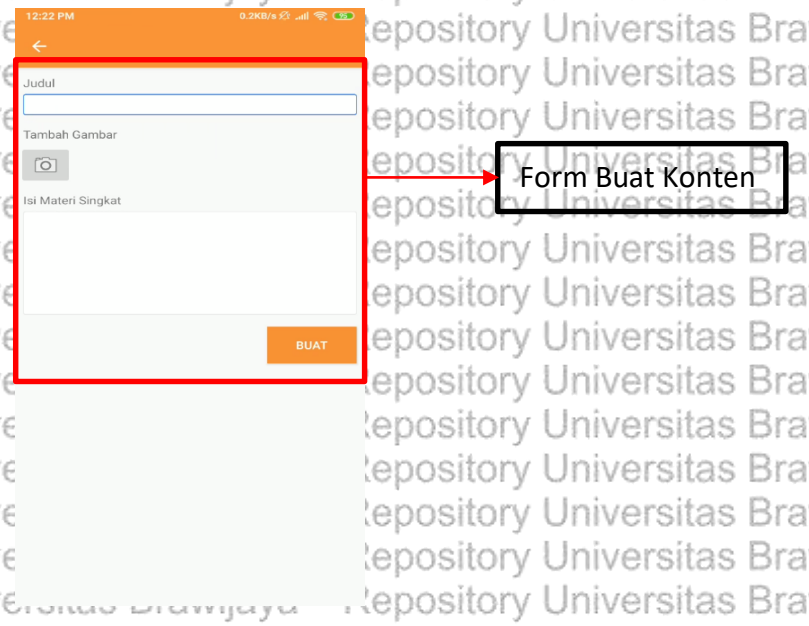
Halaman lihat materi merupakan antarmuka untuk melihat konten materi. Antarmuka halaman lihat materi ditunjukkan oleh Gambar 5.69 berikut :



Gambar 5.69 *Update* Antarmuka Halaman Lihat Materi

b. Halaman Buat Materi

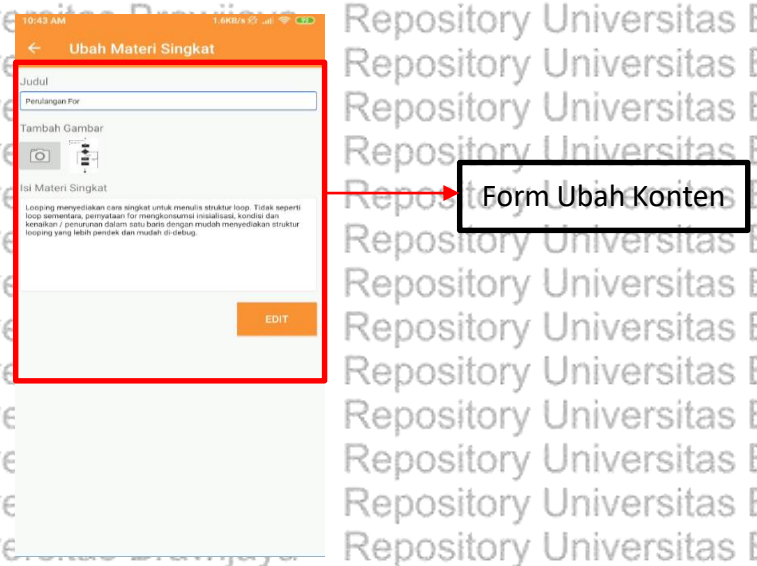
Halaman buat materi merupakan antarmuka yang berfungsi untuk membuat konten materi. Antarmuka halaman buat materi ditunjukkan oleh Gambar 5.70 berikut :



Gambar 5.70 *Update* Antarmuka Halaman Buat Materi

c. Halaman Ubah Materi

Halaman ubah materi merupakan antarmuka untuk mengubah konten materi. Antarmuka halaman ubah materi ditunjukkan oleh Gambar 5.71 berikut :



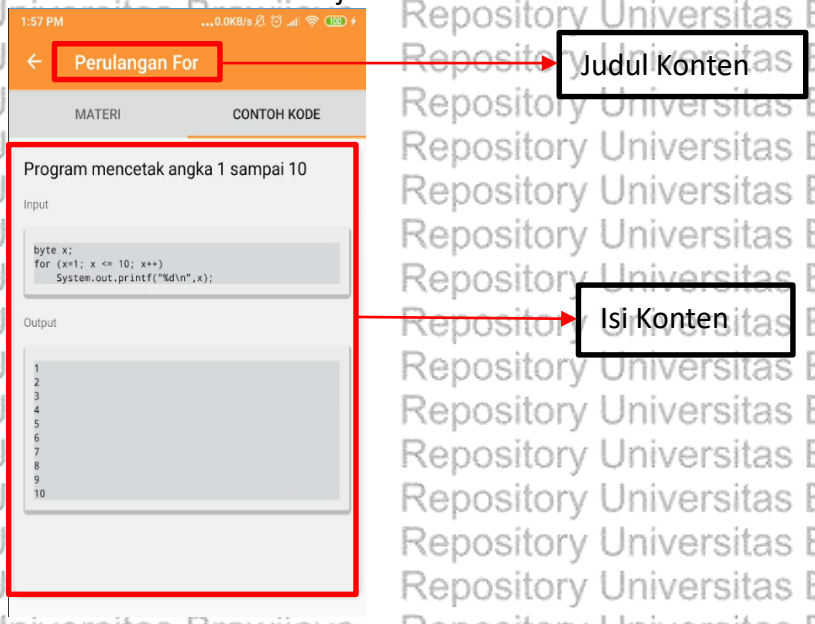
Gambar 5.71 *Update* Halaman Antarmuka Ubah Materi

5.2.22.3 *Update* Antarmuka Contoh *Source Code*

Pada subbab ini menjelaskan hasil tampilan fitur contoh *source code* pada aplikasi *BelajarJava*. Fitur contoh *source* code pada aplikasi *BelajarJava* terdiri dari 3 bagian, yakni buat contoh *source code*, ubah contoh *source code*, dan lihat materi.

a. Halaman Lihat Contoh *Source Code*

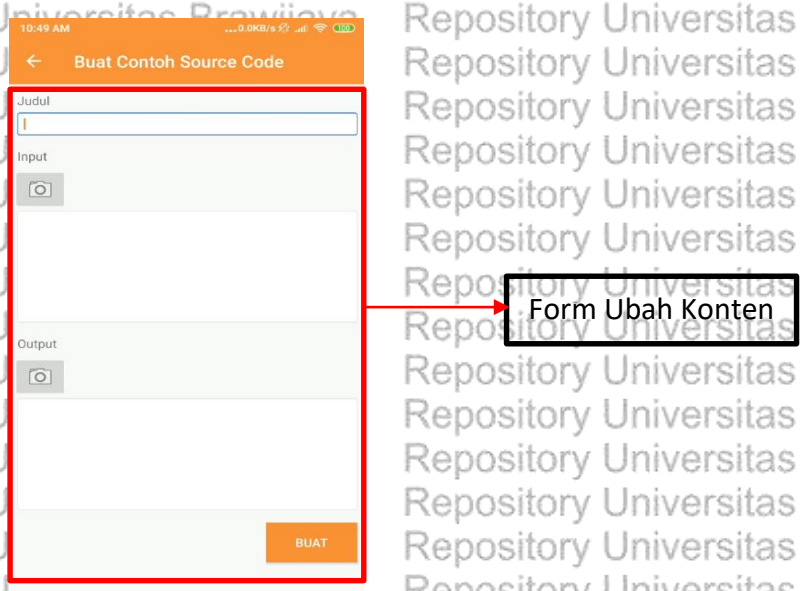
Halaman lihat contoh *source code* merupakan antarmuka yang berfungsi untuk melihat konten contoh source code. Antarmuka halaman lihat contoh *source code* ditunjukkan oleh Gambar 5.72 berikut :



Gambar 5.72 *Update* Antarmuka Halaman Lihat Contoh Source Code

b. Halaman Buat Contoh *Source Code*

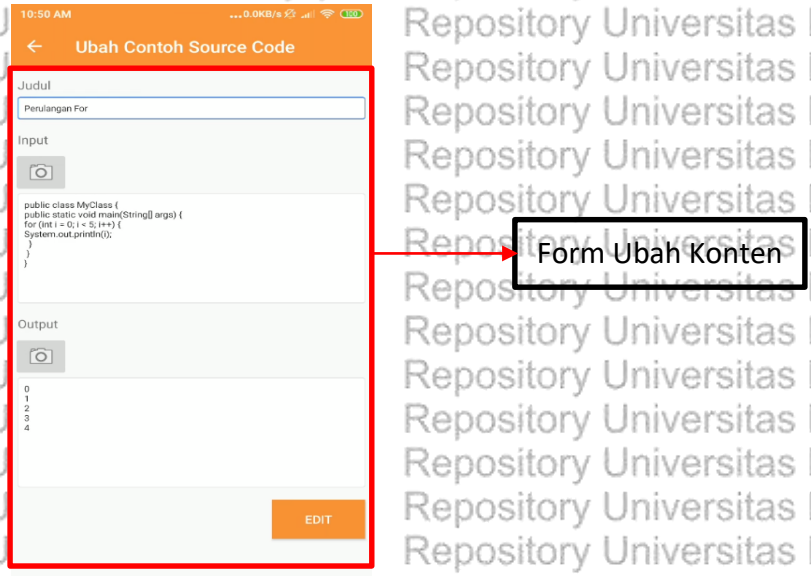
Halaman ubah contoh *source code* merupakan antarmuka yang berfungsi untuk mengubah konten contoh *source code*. Antarmuka halaman buat contoh *source code* ditunjukkan oleh Gambar 5.73 berikut :



Gambar 5.73 *Update* Antarmuka Halaman Buat Contoh *Source Code*

c. Halaman Ubah Contoh *Source Code*

Halaman ubah contoh *source code* merupakan antarmuka yang berfungsi untuk mengubah konten contoh *source code*. Antarmuka halaman ubah contoh *source code* ditunjukkan oleh Gambar 5.74 berikut :



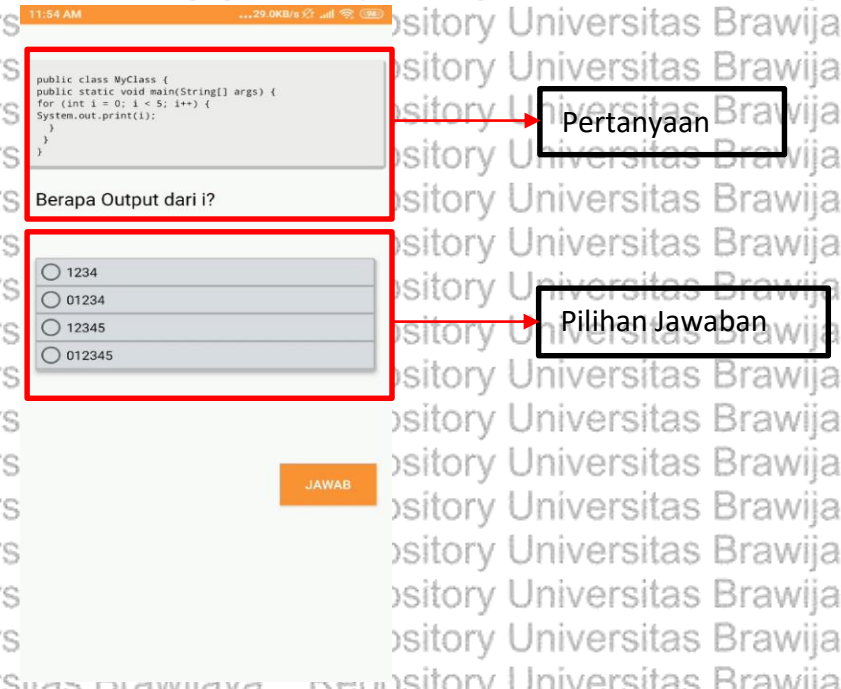
Gambar 5.74 *Update* Antarmuka Halaman Ubah Contoh *Source Code*

5.2.22.4 *Update* Antarmuka Latihan

Pada subbab ini menjelaskan hasil tampilan fitur latihan pada aplikasi *BelajarJava*. Fitur latihan pada aplikasi *BelajarJava* terdiri dari 3 bagian, yakni halaman latihan, halaman buat latihan, dan halaman ubah latihan.

a. Halaman Latihan

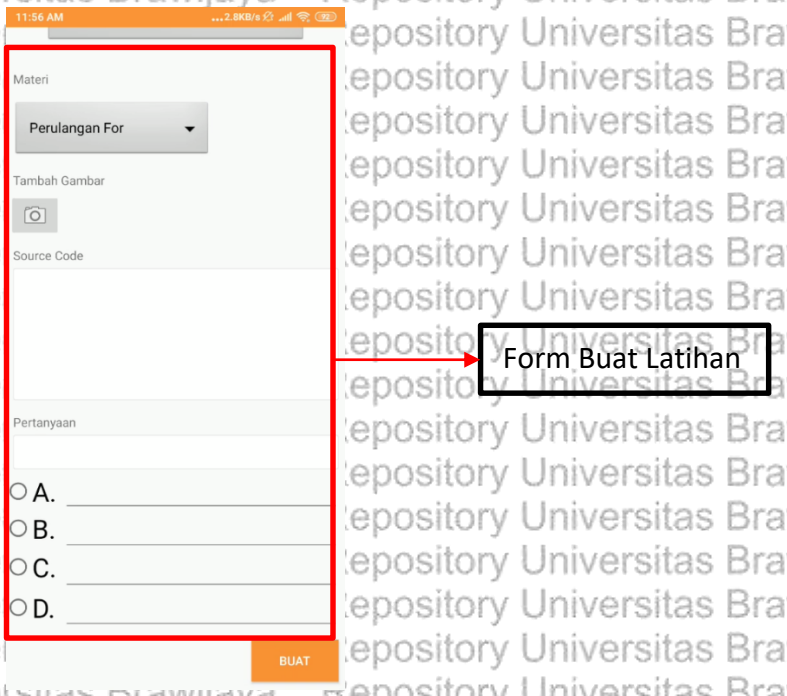
Halaman latihan merupakan antarmuka yang berfungsi untuk menjawab latihan soal. Antarmuka halaman latihan ditunjukkan oleh Gambar 5.75 berikut :



Gambar 5.75 *Update* Antarmuka Halaman Latihan

b. Halaman Buat Latihan

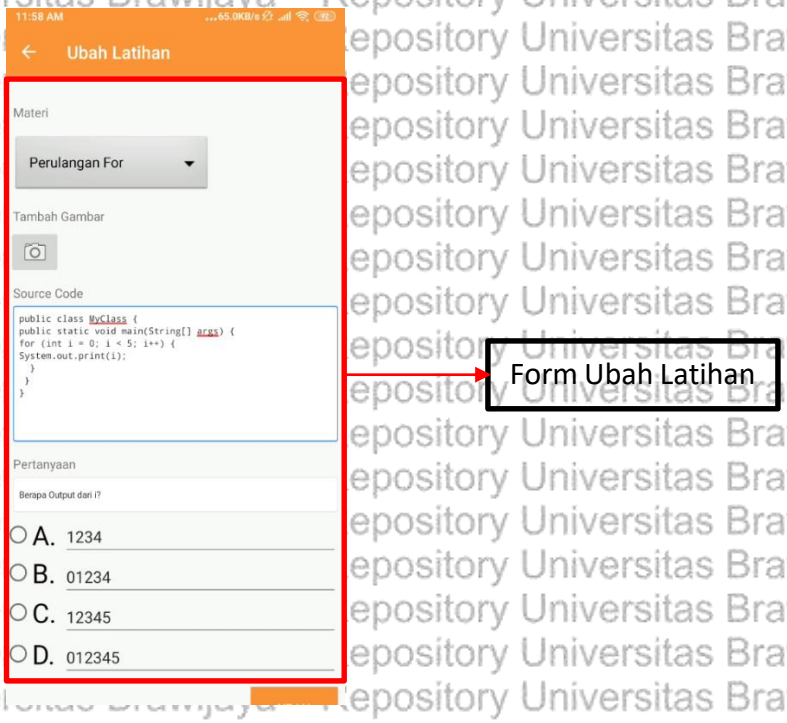
Halaman buat latihan merupakan antarmuka untuk membuat konten latihan soal. Antarmuka halaman buat latihan ditunjukkan oleh Gambar 5.76 berikut :



Gambar 5.76 *Update* Antarmuka Halaman Buat Latihan

c. Halaman Ubah Latihan

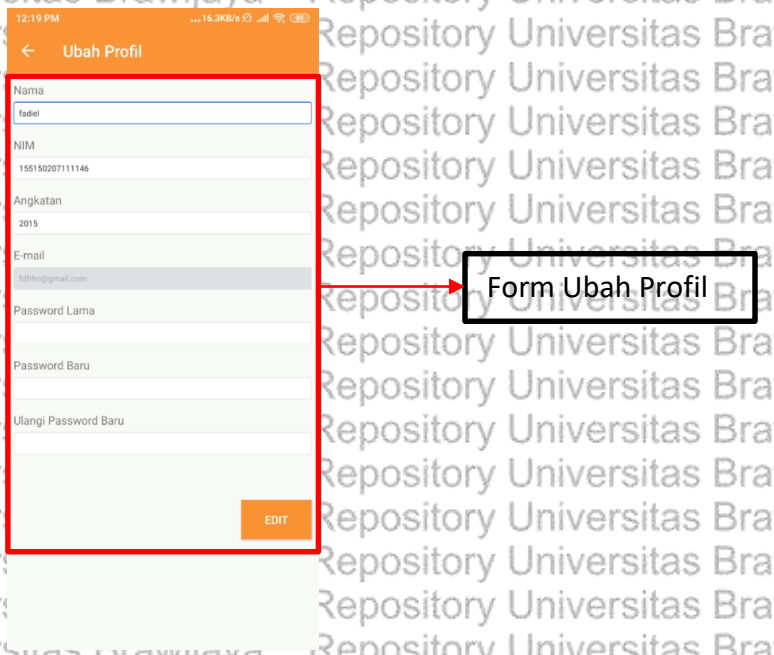
Halaman ubah latihan merupakan antarmuka yang berfungsi untuk membuat konten latihan. Antarmuka halaman latihan ditunjukkan oleh Gambar 5.77 berikut :



Gambar 5.77 *Update* Antarmuka Halaman Ubah Latihan

5.2.22.5 Implementasi Antarmuka Ubah Profil

Pada bagian ini menjelaskan hasil tampilan fitur ubah profil yang terdapat pada aplikasi *BelajarJava*. Antarmuka halaman ubah profil berfungsi untuk mengubah data pengguna. Antarmuka halaman ubah profil ditunjukkan oleh Gambar 5.78.



Gambar 5.78 *Update* Antarmuka Halaman Ubah Profil

5.2.22.6 *Update* Antarmuka Skor

Pada bagian ini menjelaskan hasil tampilan fitur skor pada aplikasi *BelajarJava*. Fitur skor pada aplikasi *BelajarJava* terdiri dari 2 bagian, yakni lihat data mahasiswa, dan lihat data skor.

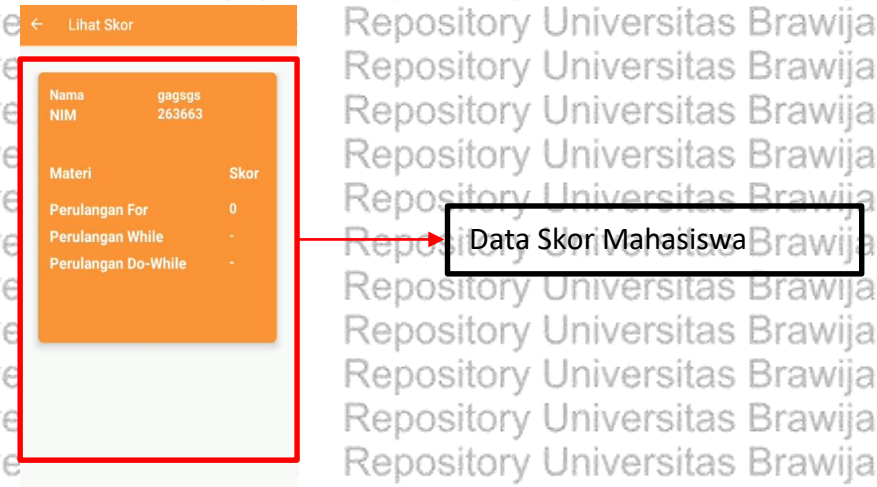
a. Halaman lihat data mahasiswa

Halaman latihan merupakan antarmuka yang berfungsi untuk melihat data mahasiswa. Antarmuka halaman lihat data mahasiswa ditunjukkan oleh Gambar 5.79 berikut :

Gambar 5.79 *Update* Antarmuka Halaman Lihat Data Mahasiswa

b. Halaman lihat data skor

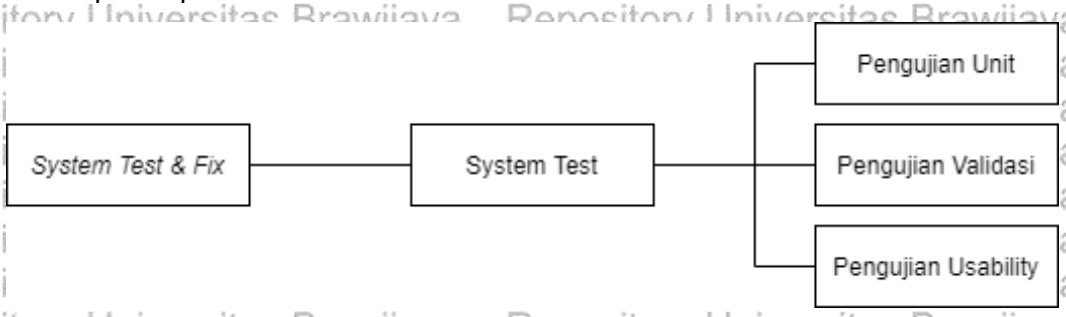
Halaman lihat data skor merupakan antarmuka yang berfungsi untuk melihat data skor latihan mahasiswa. Antarmuka halaman latihan ditunjukkan oleh Gambar 5.80 berikut :



Gambar 5.80 *Update* Antarmuka Halaman Lihat Data Skor

BAB 6 PENGUJIAN

Bab ini fase *System Test &* Fix. Pengujian akan dilakukan dengan melakukan pengujian fungsional dan non-fungsional sistem, serta analisa dari hasil pengujian yang telah dilakukan. Tahapan pengujian sistem yang akan dijelaskan pada bab ini ditampilkan pada Gambar 6.1.



Gambar 6.1 Diagram Pohon Pengujian

6.1 Pengujian Fungsional

Pengujian fungsional dilakukan untuk menguji seluruh kebutuhan fungsional sistem. Penulis melakukan pengujian unit, dan validasi. Pada tahap pengujian ini, penulis hanya melakukan pengujian pada *method* fungsi utama saja.

6.1.1 Pengujian Unit

Sistem yang telah dikembangkan akan dilakukan pengujian unit menggunakan teknik *Basis Path Testing* pada *method* dari suatu kelas. Teknik *Basic Path Testing* dilakukan dengan memodelkan suatu algoritma pada suatu *flow graph*, yang kemudian ditentukan jumlah *cyclomatic complexity* pada algoritma. Pada pengujian ini dilakukan dengan menggunakan kode program yang telah di implementasi setelah TDD dilakukan.

Untuk menghitung *cyclomatic complexity*, digunakan persamaan V(G) = E-N+2. V(G) adalah jumlah *cyclomatic complexity*, lalu E adalah garis yang menghubungkan suatu node dengan node yang lain, dan N merupakan jumlah node.

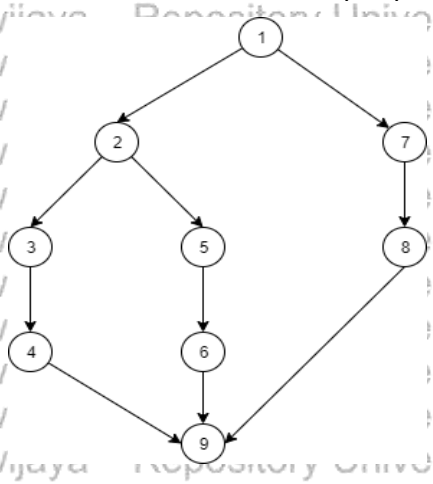
6.1.1.1 Pengujian *Method* checkForm

Method `checkForm` merupakan *method* penting dalam kode program sistem *BelajarJava*, dikarenakan *method* ini merupakan unit *method* dalam *class* yang terkait dengan *use case* buat materi, ubah materi, buat contoh *source code*, ubah contoh *source code*, buat latihan, dan ubah latihan. Walaupun *method* ini di implementasikan pada kelas yang lain, namun alur pada *method* ini sama. *Method* cekForm yang diuji ialah *method* yang berada pada *class* buatMateriSingkat.

```
private void checkForm() {
        materi.setJudulMateri(EtJudul.getText().toString());
materi.setIsiMateri(EtIsiMateri.getText().toString());
        materi.setGambarMateri("");

if(materi.getSubMateri().trim().length() > 0){
        if (filePath != null ||
materiSingkat.getIsiMateri().trim().length() > 0){
            buatMateriSingkat();
        } else {
            Toast.makeText(getBaseContext(), "Isi
gambar/teks materi terlebih dahulu.",
Toast.LENGTH_SHORT).show();
        }
    } else {
        Toast.makeText(getBaseContext(), "Isi judul
terlebih dahulu.", Toast.LENGTH_SHORT).show();
    }
}
```

Gambar 6.2 Kode Program *Method* checkForm

Berdasarkan kode program pada Gambar 6.2, dibuat *flow graph* dari *method* `checkForm`. *Flow* graph *method* `checkForm` terdapat pada Gambar 6.3.



Gambar 6.3 *Flow Graph Method* checkForm

Setelah ditentukan model algoritma dalam bentuk *flow graph*, kemudian dilakukan penghitungan *cyclomatic complexity*. Berdasarkan gambar 6.3, jumlah *cyclomatic complexity* dari *method* `checkForm` adalah sebagai berikut :

V(G) = 10 – 9 + 2

= 1 + 2

= 3

Berdasarkan nilai *cyclomatic complexity* yang telah ditentukan, didapatkan tiga jalur independen, yaitu :

Jalur 1 : 1 – 2 – 3 – 4 – 9

Jalur 2 : 1 – 2 – 5 – 6 – 9

Jalur 3 : 1 – 7 – 8 – 9

Berdasarkan jumlah jalur independen yang telah didapatkan, dilakukan pengujian tiap jalur independen yang terdapat pada Tabel 6.1.

Tabel 6.1 Kasus Uji Pengujian Unit *Method* checkForm

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Nilai variabel judulMateri kosong	Pemanggilan *method* buatMateri	Pemanggilan *method* buatMateri
2	Nilai variabel filePath null atau nilai variabel isiMateri kosong	Menampilkan pesan toast untuk mengisi gambar atau isi materi	Menampilkan pesan toast untuk mengisi gambar atau isi materi
3	Nilai variabel judulMateri kosong	Menampilkan pesan toast untuk mengisi judul	Menampilkan pesan toast untuk mengisi judul

6.1.1.2 Pengujian *Method* buatMateri

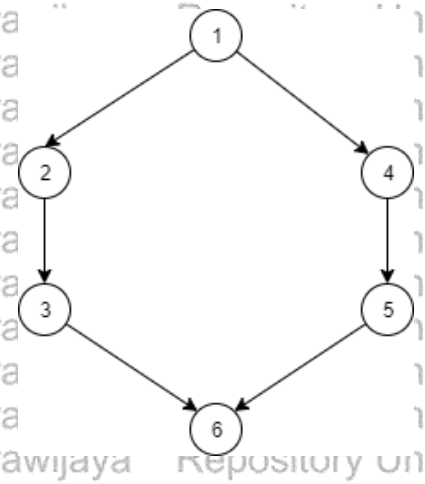
Method `buatMateri` merupakan *method* untuk membuat konten materi. Pada pengujian ini, hanya diuji *method* `buatMateri` yang alurnya sama dengan *method* `ubahMateri`, `buatContohSourceCode`, `ubahContohSourceCode`, `buatLatihan`, dan `ubahLatihan`.

```
private void buatMateri() {
        databaseReference =
FirebaseDatabase.getInstance().getReference();
        DatabaseReference postsRef =
databaseReference.child("materi").child(materi.getId());
        DatabaseReference newPostRef =
postsRef.child("sub_materi").push();
        subMateri.setId(newPostRef.getKey());

        storageReference =
FirebaseStorage.getInstance().getReference();
        StorageReference storageRef =
storageReference.child("subMateri/"+
UUID.randomUUID().toString());

        final ProgressDialog progressDialog = new
ProgressDialog(Activity_BuatSubMateri.this);
        progressDialog.setTitle("Menambah Konten");
        progressDialog.show();

        if(filePath != null){
            buatDenganGambar(storageRef, progressDialog,
newPostRef);
        } else {
            buatTanpaGambar(progressDialog, newPostRef);
        }
    }
```

1 2 3 4 5 6

Gambar 6.4 Kode Program *Method* buatMateri

Berdasarkan kode program pada Gambar 6.4, dibuat *flow graph* dari *method* `buatMateri`. *Flow* graph *method* `buatMateri` terdapat pada Gambar 6.5.



Gambar 6.5 *Flow Graph Method* buatMateri

Setelah ditentukan model algoritma dalam bentuk *flow graph*, kemudian dilakukan penghitungan *cyclomatic complexity*. Berdasarkan Gambar 6.5, jumlah *cyclomatic complexity* dari *method* `buatMateri` adalah sebagai berikut :

V(G) = 6 – 6 + 2

= 0 + 2

= 2

Berdasarkan nilai *cyclomatic complexity* yang telah ditentukan, didapatkan tiga jalur independen, yaitu :

Jalur 1 : 1 – 2 – 3 – 6

Jalur 2 : 1 – 4 – 5 – 6

Berdasarkan jumlah jalur independen yang telah didapatkan, dilakukan pengujian tiap jalur independen tersebut yang terdapat pada Tabel 6.2.

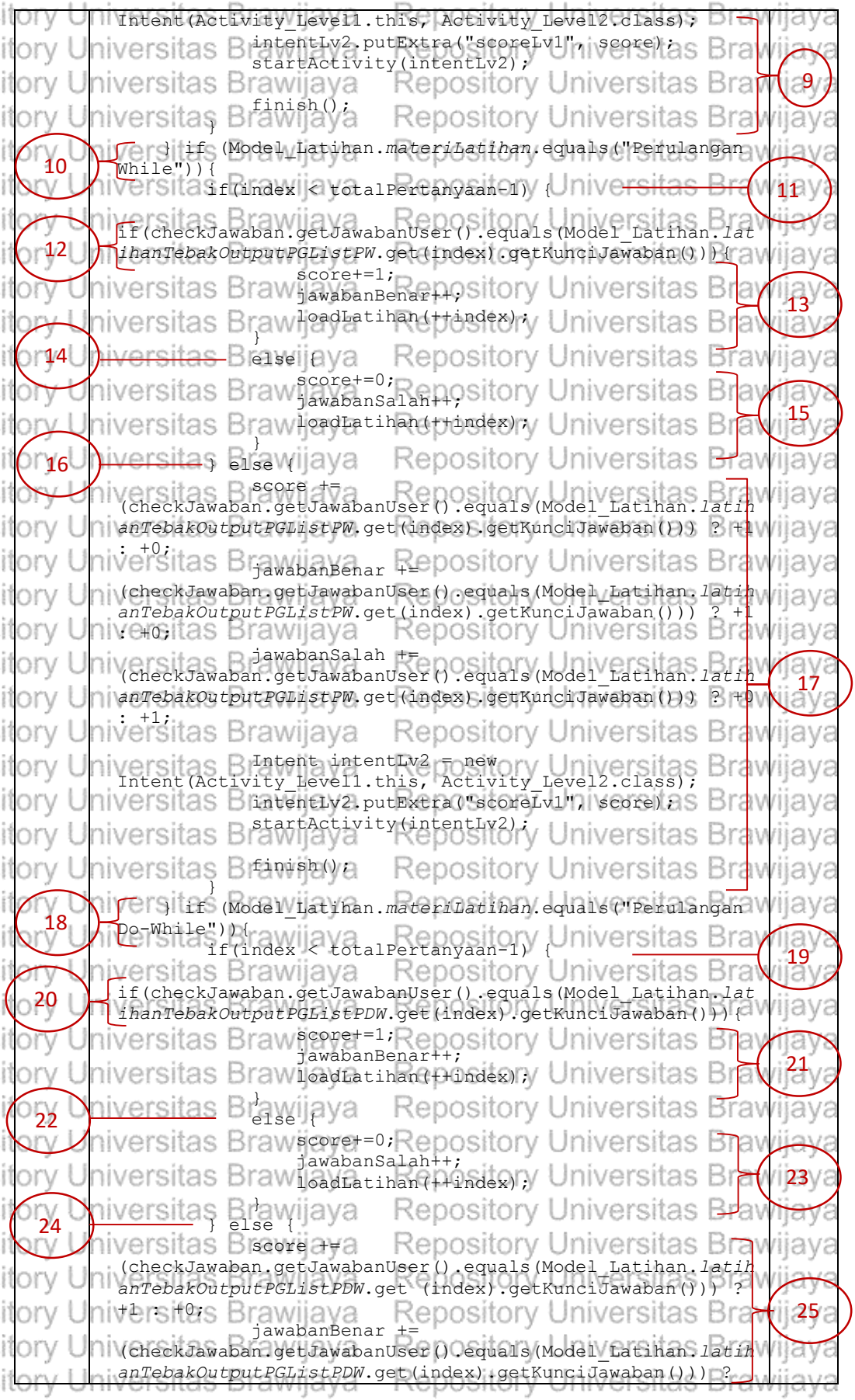
Tabel 6.2 Kasus Uji Pengujian Unit *Method* buatMateriSingkat

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Jika nilai variabel filePath tidak bernilai null	Pemanggilan *method* buatDenganGambar	Pemanggilan *method* buatDenganGambar
2	Jika nilai variabel filePath bernilai null	Pemanggilan *method* buatTanpaGambar	Pemanggilan *method* buatTanpaGambar

6.1.1.3 Pengujian *Method* jawabLatihan

Method jawabLatihan merupakan *method* untuk menjawab soal latihan. Kode program *method* `jawabLatihan` terdapat pada Gambar 6.6.

```
private void jawabLatihan() {
    if(Model_Latihan.materiLatihan.equals("Perulangan
For")){
        if(index < totalPertanyaan-1) {
if(checkJawaban.getJawabanUser().equals(Model_Latihan.lat
ihanTebakOutputPGListPF.get(index).getKunciJawaban())){
                score+=1;
                jawabanBenar++;
                loadLatihan(++index);
            }
            else {
                score+=0;
                jawabanSalah++;
                loadLatihan(++index) ;
            }
        }
        else {
            score +=
(checkJawaban.getJawabanUser().equals(Model_Latihan.latih
anTebakOutputPGListPF.get(index).getKunciJawaban())) ? +1
: +0;
            jawabanBenar +=
(checkJawaban.getJawabanUser().equals(Model_Latihan.latih
anTebakOutputPGListPF.get(index).getKunciJawaban())) ? +1
: +0;
            jawabanSalah +=
(checkJawaban.getJawabanUser().equals(Model_Latihan.latih
anTebakOutputPGListPF.get(index).getKunciJawaban())) ? +0
: +1;
            Intent intentLv2 = new
```


Intent(Activity_Level1.this, Activity_Level2.class);
intentLv2.putExtra("scoreLv1", score);
startActivity(intentLv2);
finish();
} } if (Model_Latihan.materiLatihan.equals("Perulangan While")){
if(index < totalPertanyaan-1) {
if(checkJawaban.getJawabanUser().equals(Model_Latihan.latihanTebakOutputPGListPW.get(index).getKunciJawaban())){
score+=1;
jawabanBenar++;
loadLatihan(++index);
}
else {
score+=0;
jawabanSalah++;
loadLatihan(++index);
}
} else {
score += (checkJawaban.getJawabanUser().equals(Model_Latihan.latihanTebakOutputPGListPW.get(index).getKunciJawaban())) ? +1 : +0;
jawabanBenar += (checkJawaban.getJawabanUser().equals(Model_Latihan.latihanTebakOutputPGListPW.get(index).getKunciJawaban())) ? +1 : +0;
jawabanSalah += (checkJawaban.getJawabanUser().equals(Model_Latihan.latihanTebakOutputPGListPW.get(index).getKunciJawaban())) ? +0 : +1;
Intent intentLv2 = new Intent(Activity_Level1.this, Activity_Level2.class);
intentLv2.putExtra("scoreLv1", score);
startActivity(intentLv2);
finish();
} } if (Model_Latihan.materiLatihan.equals("Perulangan Do-While")){
if(index < totalPertanyaan-1) {
if(checkJawaban.getJawabanUser().equals(Model_Latihan.latihanTebakOutputPGListPDW.get(index).getKunciJawaban())){
score+=1;
jawabanBenar++;
loadLatihan(++index);
}
else {
score+=0;
jawabanSalah++;
loadLatihan(++index);
}
} else {
score += (checkJawaban.getJawabanUser().equals(Model_Latihan.latihanTebakOutputPGListPDW.get (index).getKunciJawaban())) ? +1 : +0;
jawabanBenar += (checkJawaban.getJawabanUser().equals(Model_Latihan.latihanTebakOutputPGListPDW.get(index).getKunciJawaban())) ?
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

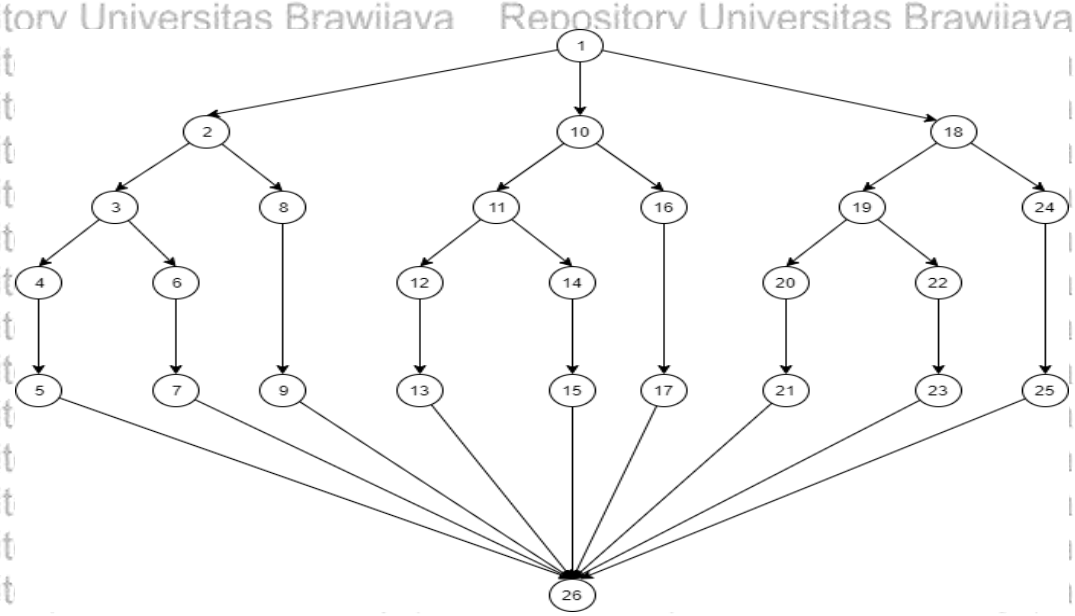
```
+1 : +0;
            jawabanSalah +=
(checkJawaban.getJawabanUser().equals(Model_Latihan.latih
anTebakOutputPGListPDW.get(index).getKunciJawaban())) ?
+0 : +1;
            Intent intentLv2 = new
Intent(Activity_Level1.this, Activity_Level2.class);
            intentLv2.putExtra("scoreLv1", score);
            startActivity(intentLv2);

            finish();
        }
    }
}
```

25

26

Gambar 6.6 Kode Program *Method* jawabLatihan

Berdasarkan kode program pada Gambar 6.6, dibuat *flow graph* dari *method* `jawabLatihan`. *Flow* graph *method* `buatMateriSingkat` terdapat pada Gambar 6.7.



Gambar 6.7 *Flow Graph Method* jawabLatihan

Setelah ditentukan model algoritma dalam bentuk *flow graph*, kemudian dilakukan penghitungan *cyclomatic complexity*. Berdasarkan gambar 6.15, jumlah *cyclomatic complexity* dari *method* `jawabLatihan` adalah sebagai berikut :

V(G) = 33 – 26 + 2
= 7 + 2
= 9

Berdasarkan nilai *cyclomatic complexity* yang telah ditentukan, didapatkan tiga jalur independen, yaitu :

Jalur 1 : 1 – 2 – 3 – 4 – 5 – 26

Jalur 2 : 1 – 2 – 3 – 6 – 7 – 26

Jalur 3 : 1 – 2 – 8 – 9 – 26

Jalur 4 : 1 – 10 – 11 – 12 – 13 – 26

Jalur 5 : 1 – 10 – 11 – 14 – 15 – 26

Jalur 6 : 1 – 10 – 16 – 17 – 26

Jalur 7 : 1 – 18 – 19 – 20 – 21 – 26

Jalur 8 : 1 – 18 – 19 – 22 – 23 – 26

Jalur 9 : 1 – 18 – 24 – 25 – 26

Berdasarkan jumlah jalur independen yang telah didapatkan, akan dilakukan pengujian tiap jalur independen tersebut yang terdapat di Tabel 6.3.

Tabel 6.3 Kasus Uji Pengujian Unit *Method* jawabLatihan

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Cek jawaban materi perulangan *for* kondisi benar dan masih terdapat pertanyaan lain	Variabel score, jawabanBenar bertambah, dan memanggil method loadLatihan	Variabel score, jawabanBenar bertambah, dan memanggil method loadLatihan
2	Cek jawaban materi perulangan for kondisi salah dan masih terdapat pertanyaan lain	Variabel score tidak bertambah, variabel jawabanSalah bertambah, dan memanggil method loadLatihan	Variabel score tidak bertambah, variabel jawabanSalah bertambah, dan memanggil method loadLatihan
3	Cek jawaban materi perulangan for ketika tidak terdapat pertanyaan lagi	Cek jawaban materi perulangan for terakhir yang kemudian variabel score, jawabanBenar, jawabanSalah dapat bertambah atau tidak sesuai jawaban yang salah atau benar. Kemudian pemanggilan activity Level 2	Cek jawaban materi perulangan for terakhir yang kemudian variabel score, jawabanBenar, jawabanSalah dapat bertambah atau tidak sesuai jawaban yang salah atau benar. Kemudian pemanggilan activity Level 2
4	Cek jawaban materi perulangan while	Variabel score, jawabanBenar bertambah, dan	Variabel score, jawabanBenar bertambah,

	kondisi benar dan masih terdapat pertanyaan lain	memanggil method loadLatihan	dan memanggil method loadLatihan
5	Cek jawaban materi perulangan while kondisi salah dan masih terdapat pertanyaan lain	Variabel score tidak bertambah, variabel jawabanSalah bertambah, dan memanggil method loadLatihan	Variabel score tidak bertambah, variabel jawabanSalah bertambah, dan memanggil method loadLatihan
6	Cek jawaban materi perulangan while ketika tidak terdapat pertanyaan lagi	Cek jawaban materi perulangan while terakhir yang kemudian variabel score, jawabanBenar, jawabanSalah dapat bertambah atau tidak sesuai jawaban yang salah atau benar. Kemudian pemanggilan activity Level 2	Cek jawaban materi perulangan while terakhir yang kemudian variabel score, jawabanBenar, jawabanSalah dapat bertambah atau tidak sesuai jawaban yang salah atau benar. Kemudian pemanggilan activity Level 2
7	Cek jawaban materi perulangan do-while kondisi benar dan masih terdapat pertanyaan lain	Variabel score, jawabanBenar bertambah, dan memanggil method loadLatihan	Variabel score, jawabanBenar bertambah, dan memanggil method loadLatihan
8	Cek jawaban materi perulangan do-while kondisi salah dan masih terdapat pertanyaan lain	Variabel score tidak bertambah, variabel jawabanSalah bertambah, dan memanggil method loadLatihan	Variabel score tidak bertambah, variabel jawabanSalah bertambah, dan memanggil method loadLatihan
9	Cek jawaban materi perulangan do-while ketika tidak terdapat pertanyaan lagi	Cek jawaban materi perulangan do-while terakhir yang kemudian variabel score, jawabanBenar, jawabanSalah dapat bertambah atau tidak sesuai jawaban yang	Cek jawaban materi perulangan do-while terakhir yang kemudian variabel score, jawabanBenar, jawabanSalah dapat bertambah atau tidak sesuai jawaban yang salah

		salah atau benar. Kemudian pemanggilan activity Level 2	atau benar. Kemudian pemanggilan activity Level 2

6.1.2 Pengujian Validasi

Kebutuhan yang telah dirumuskan ke dalam spesifikasi kebutuhan akan dijadikan dasar dalam melakukan pengujian validasi. Pengujian validasi dilakukan juga untuk mengetahui apakah ada fitur yang tidak bisa dilakukan atau terdapat *error*/*bug* pada pelaksanaannya.

6.1.2.1 Kasus Uji Validasi *Guest*

a. Daftar pengguna baru

Tabel 6.4 Pengujian Validasi Kasus Uji Daftar Pengguna Baru

Kasus Uji	Daftar pengguna baru
Tujuan Pengujian	Memastikan sistem dapat membuat data pengguna baru
Prosedur Pengujian	1. Masukkan data pengguna 2. Menekan tombol daftar
Hasil yang Diharapkan	Pengguna akan masuk ke dalam sistem

b. *Login*

Tabel 6.5 Pengujian Validasi Kasus Uji *Login*

Kasus Uji	*Login*
Tujuan Pengujian	Memastikan pengguna yang datanya telah terdaftar dalam sistem dapat melakukan autentifikasi
Prosedur Pengujian	1. Masukkan data *login* 2. Menekan tombol *login*
Hasil yang Diharapkan	Pengguna dapat melakukan autentifikasi dan masuk ke dalam sistem

c. *Reset password*

Tabel 6.6 Pengujian Validasi Kasus Uji *Reset Password*

Kasus Uji	*Reset password*
Tujuan Pengujian	Membuktikan bahwa pengguna yang lupa *password* dapat memulihkan *password*nya kembali

Prosedur Pengujian	1. Masukkan data *e-mail* yang telah terdaftar dalam sistem 2. Menekan tombol kirim
Hasil yang Diharapkan	*E-mail* pemulihan *password* dikirim

6.1.2.2 Kasus Uji Validasi Mahasiswa

a. Melihat konten materi

Tabel 6.7 Pengujian Validasi Kasus Uji Melihat Konten Materi

Kasus Uji	Melihat konten materi
Tujuan Pengujian	Memastikan pengguna dapat melihat isi konten materi pada sistem
Prosedur Pengujian	1. Pilih menu materi 2. Pilih materi yang ingin ditampilkan
Hasil yang Diharapkan	Konten materi ditampilkan sistem

b. Melihat konten contoh *source code* berdasarkan materi

Tabel 6.8 Pengujian Validasi Kasus Uji Melihat Konten Contoh *Source Code*

Kasus Uji	Melihat konten contoh *source code*
Tujuan Pengujian	Membuktikan bahwa pengguna dapat melihat konten isi contoh *source code*
Prosedur Pengujian	1. Pilih menu materi 2. Pilih materi yang ingin ditampilkan 3. Melakukan *swipe* ke bagian *source code*
Hasil yang Diharapkan	Sistem menampilkan isi konten contoh *source code*

c. Mengerjakan latihan berdasarkan materi

Tabel 6.9 Pengujian Validasi Kasus Uji Mengerjakan Latihan

Kasus Uji	Mengerjakan latihan
Tujuan Pengujian	Memastikan pengguna dapat menggunakan fitur latihan untuk mengerjakan latihan yang telah disediakan oleh sistem
Prosedur Pengujian	1. Pilih materi 2. Isi latihan sampai pertanyaan selesai

Hasil yang Diharapkan	Skor latihan akan ditampilkan oleh sistem

d. Lihat skor

Tabel 6.10 Pengujian Validasi Kasus Uji Lihat Skor

Kasus Uji	Lihat skor
Tujuan Pengujian	Memastikan pengguna dapat melihat skor latihan
Prosedur Pengujian	1. Pilih menu lihat skor
Hasil yang Diharapkan	Skor latihan ditampilkan sistem

e. Ubah data profil

Tabel 6.11 Pengujian Validasi Kasus Uji Ubah Data Profil

Kasus Uji	Ubah data profil
Tujuan Pengujian	Memastikan pengguna dapat mengubah data pribadi terkait profil
Prosedur Pengujian	1. Pilih menu ubah profil 2. Isi form profil 3. Tekan tombol ubah
Hasil yang Diharapkan	Terdapat notifikasi bahwa data telah diubah oleh sistem

f. *Logout*

Tabel 6.12 Pengujian Validasi Kasus Uji *Logout*

Kasus Uji	*Logout*
Tujuan Pengujian	Memastikan pengguna dapat membatalkan autentifikasi dan keluar dari sistem
Prosedur Pengujian	1. Pilih menu *logout*
Hasil yang Diharapkan	Sistem kembali ke halaman *login*

6.1.2.3 Kasus Uji Validasi Dosen

a. Buat konten materi

Tabel 6.13 Pengujian Validasi Kasus Uji Buat Konten Materi

Kasus Uji	Buat konten materi
Tujuan Pengujian	Memastikan sistem dapat menambah isi konten materi
Prosedur Pengujian	1. Pilih menu buat materi 2. Isi *form* 3. Tekan tombol buat 4. Masuk menu materi untuk melihat materi yang baru dibuat 5. Pilih materi yang telah dibuat 6. Pilih menu tambah sub materi baru 7. Isi form
Hasil yang Diharapkan	Notifikasi bahwa konten materi telah dibuat ditampilkan oleh sistem

b. Ubah konten materi

Tabel 6.14 Pengujian Validasi Kasus Uji Ubah Konten Materi

Kasus Uji	Ubah konten materi
Tujuan Pengujian	Memastikan sistem dapat mengubah isi konten materi
Prosedur Pengujian	1. Pilih konten yang akan diubah 2. Isi *form* 3. Tekan tombol ubah
Hasil yang Diharapkan	Notifikasi bahwa konten materi telah diubah ditampilkan oleh sistem

c. Hapus konten materi

Tabel 6.15 Pengujian Validasi Kasus Uji Hapus Konten Materi

Kasus Uji	Hapus konten materi
Tujuan Pengujian	Membuktikan bahwa sistem dapat menghapus konten materi
Prosedur Pengujian	1. Pilih konten yang akan diubah 2. Pilih menu hapus
Hasil yang Diharapkan	Notifikasi bahwa konten materi telah dihapus ditampilkan oleh sistem

d. Buat konten contoh *source code*

Tabel 6.16 Pengujian Validasi Kasus Uji Buat Contoh *Source Code*

Kasus Uji	Buat konten contoh *source code*
Tujuan Pengujian	Membuktikan bahwa sistem dapat menambah konten contoh *source code*
Prosedur Pengujian	1. Masuk menu materi 2. Pilih materi 3. Pilih menu tambah *source code*
Hasil yang Diharapkan	Sistem menampilkan notifikasi bahwa konten contoh *source code* telah dibuat

e. Ubah konten contoh *source code*

Tabel 6.17 Pengujian Validasi Kasus Uji Ubah Contoh *Source Code*

Kasus Uji	Ubah konten contoh *source code*
Tujuan Pengujian	Membuktikan bahwa sistem dapat mengubah konten contoh *source code*
Prosedur Pengujian	1. Pilih konten yang akan diubah 2. Isi form 3. Tekan tombol ubah
Hasil yang Diharapkan	Sistem menampilkan notifikasi bahwa konten contoh *source code* telah diubah

f. Hapus konten contoh *source code*

Tabel 6.18 Pengujian Validasi Kasus Uji Hapus Konten Contoh *Source Code*

Kasus Uji	Hapus konten contoh *source code*
Tujuan Pengujian	Memastikan sistem dapat menghapus isi konten contoh *source code*
Prosedur Pengujian	1. Pilih konten yang akan diubah 2. Pilih menu hapus
Hasil yang Diharapkan	Notifikasi bahwa konten contoh *source code* dihapus ditampilkan oleh sistem

g. Cek konten latihan

Tabel 6.19 Pengujian Validasi Kasus Uji Cek Konten Latihan

Kasus Uji	Cek konten latihan
Tujuan Pengujian	Memastikan sistem dapat melihat isi konten latihan
Prosedur Pengujian	1. Pilih kategori latihan
Hasil yang Diharapkan	Konten isi latihan ditampilkan oleh sistem

h. Buat konten latihan

Tabel 6.20 Pengujian Validasi Kasus Uji Buat Konten Latihan

Kasus Uji	Buat konten latihan
Tujuan Pengujian	Membuktikan sistem dapat menambah konten latihan
Prosedur Pengujian	1. Pilih kategori latihan 2. Tentukan materi latihan 3. Isi form 4. Tekan tombol buat
Hasil yang Diharapkan	Notifikasi bahwa konten latihan telah ditambah ditampilkan oleh sistem

i. Ubah konten latihan

Tabel 6.21 Pengujian Validasi Kasus Uji Ubah Konten Latihan

Kasus Uji	Ubah konten latihan
Tujuan Pengujian	Membuktikan bahwa sistem dapat mengubah data konten latihan
Prosedur Pengujian	1. Pilih konten latihan yang ingin diubah 2. Isi form 3. Tekan tombol ubah
Hasil yang Diharapkan	Notifikasi penambahan konten latihan ditampilkan oleh sistem

j. Hapus konten latihan

Tabel 6.22 Pengujian Validasi Kasus Uji Hapus Konten Latihan

Kasus Uji	Hapus konten latihan
Tujuan Pengujian	Memastikan sistem dapat menghapus isi konten latihan

Prosedur Pengujian	1. Pilih konten yang akan diubah 2. Pilih menu hapus
Hasil yang Diharapkan	Notifikasi penghapusan isi konten latihan ditampilkan oleh sistem

k. Lihat data mahasiswa

Tabel 6.23 Pengujian Validasi Kasus Uji Lihat Data Mahasiswa

Kasus Uji	Lihat data mahasiswa
Tujuan Pengujian	Membuktikan bahwa sistem dapat mengubah data konten latihan
Prosedur Pengujian	1. Pilih menu lihat skor
Hasil yang Diharapkan	Sistem menampilkan data nama mahasiswa dan NIM mahasiswa yang bersangkutan

l. Lihat skor mahasiswa

Tabel 6.24 Pengujian Validasi Kasus Uji Lihat Skor Mahasiswa

Kasus Uji	Lihat data mahasiswa
Tujuan Pengujian	Membuktikan bahwa sistem dapat mengubah data konten latihan
Prosedur Pengujian	1. Pilih menu lihat skor 2. Pilih data mahasiswa yang ingin dilihat skornya
Hasil yang Diharapkan	Sistem menampilkan skor mahasiswa yang bersangkutan

6.1.2.4 Hasil Pengujian Validasi *Guest*

Hasil pengujian validasi *guest* terdapat pada pada Tabel 6.25.

Tabel 6.25 Hasil Pengujian Validasi *Guest*

No	Nama Kasus Uji	Hasil Yang Diharapkan	Hasil Yang Didapatkan	Validitas
1	Daftar pengguna baru	Pengguna akan masuk ke dalam sistem	Pengguna masuk ke dalam sistem	Valid

2	Login	Pengguna akan masuk ke dalam sistem	Pengguna masuk ke dalam sistem	Valid
3	*Reset password*	*E-mail* pemulihan *password* dikirim	Terdapat *e-mail* pemulihan *password*	Valid

6.1.2.5 Hasil Pengujian Validasi Mahasiswa

Hasil pengujian validasi mahasiswa terdapat pada pada Tabel 6.26.

Tabel 6.26 Hasil Pengujian Validasi Mahasiswa

No	Nama Kasus Uji	Hasil Yang Diharapkan	Hasil Yang Didapatkan	Validitas
1	Melihat konten materi	Sistem menampilkan isi konten materi	Sistem menampilkan isi konten materi	Valid
2	Melihat konten contoh *source code*	Sistem menampilkan isi konten contoh *source code*	Sistem menampilkan isi konten contoh *source code*	Valid
3	Mengerjakan latihan	Sistem akan menampilkan skor latihan yang baru saja diselesaikan	Sistem akan menampilkan skor latihan yang baru saja diselesaikan	Valid
4	Lihat skor	Sistem menampilkan skor latihan	Sistem menampilkan skor latihan	Valid
5	Ubah data profil	Sistem menampilkan notifikasi bahwa data diubah	Sistem menampilkan notifikasi bahwa data diubah	Valid
6	Logout	Sistem kembali ke halaman login	Sistem kembali ke halaman login	Valid

6.1.2.6 Hasil Pengujian Validasi Dosen

Hasil pengujian validasi dosen terdapat pada pada Tabel 6.27.

Tabel 6.27 Hasil Pengujian Validasi Dosen

No	Nama Kasus Uji	Hasil Yang Diharapkan	Hasil Yang Didapatkan	Validitas
1	Buat konten materi	Sistem menampilkan notifikasi bahwa konten materi telah dibuat	Sistem menampilkan notifikasi bahwa konten materi telah dibuat	Valid
2	Ubah konten materi	Sistem menampilkan notifikasi bahwa konten materi telah diubah	Sistem menampilkan notifikasi bahwa konten materi telah diubah	Valid
3	Hapus konten materi	Sistem menampilkan notifikasi bahwa konten materi telah dihapus	Sistem menampilkan notifikasi bahwa konten materi telah dihapus	Valid
4	Lihat skor	Sistem menampilkan skor latihan	Sistem menampilkan skor latihan	Valid
5	Buat konten contoh *source code*	Sistem menampilkan notifikasi bahwa konten contoh *source code* telah dibuat	Sistem menampilkan notifikasi bahwa konten contoh *source code* telah dibuat	Valid
6	Ubah konten contoh *source code*	Sistem menampilkan notifikasi bahwa konten contoh *source code* telah diubah	Sistem menampilkan notifikasi bahwa konten contoh *source code* telah diubah	Valid
7	Hapus konten contoh *source code*	Sistem menampilkan notifikasi bahwa konten contoh *source code* telah dihapus	Sistem menampilkan notifikasi bahwa konten contoh *source code* telah dihapus	Valid

8	Cek konten latihan	Sistem menampilkan isi konten latihan	Sistem menampilkan isi konten latihan	Valid
9	Buat konten latihan	Sistem menampilkan notifikasi bahwa konten latihan telah ditambah	Sistem menampilkan notifikasi bahwa konten latihan telah ditambah	Valid
10	Ubah konten latihan	Sistem menampilkan notifikasi bahwa konten latihan telah diubah	Sistem menampilkan notifikasi bahwa konten latihan telah diubah	Valid
11	Hapus konten latihan	Sistem menampilkan notifikasi bahwa konten latihan telah dihapus	Sistem menampilkan notifikasi bahwa konten latihan telah dihapus	Valid
12	Lihat data mahasiswa	Sistem menampilkan data nama mahasiswa dan NIM mahasiswa yang bersangkutan	Sistem menampilkan data nama mahasiswa dan NIM mahasiswa yang bersangkutan	Valid
13	Lihat data mahasiswa	Sistem menampilkan skor mahasiswa yang bersangkutan	Sistem menampilkan skor mahasiswa yang bersangkutan	Valid

6.2 Pengujian Non-Fungsional

Pada pengujian non-fungsional, akan dilakukan pengujian *usability*. Pengujian *usability* bertujuan untuk mengetahui tingkatan kepuasan pengguna.

6.2.1 Pengujian *Usability*

Pengujian *usability* bertujuan untuk mengetahui tingkatan kepuasan pengguna. Dengan menggunakan parameter *usability*, akan diketahui apakah aplikasi mudah digunakan atau tidak. Untuk mengukur tingkat kepuasan pengguna, digunakan metode *System Usability Scale* (SUS). Dilakukan juga latihan terhadap mahasiswa untuk mengetahui apakah sistem yang dikembangkan efektif atau tidak.

Menurut Nielsen (2000), sebuah proyek dapat dilakukan pengujian *usability* ke setidaknya 2 pengguna per studi. Untuk bebepa proyek, juga dapat dilakukan minimal 8 pengguna per studi. Lebih banyak pengguna maka akan makin baik.

6.2.1.1 Identifikasi Pengguna

Pada penelitian ini, pengujian *usability* akan ditujukan kepada 12 responden. Dari total 12 responden tersebut dibagi menjadi 2 kelompok, yakni 10 responden berasal dari mahasiswa, dan 2 responden berasal dari dosen.

Total dosen yang masih atau sedang mengajar materi Pemrograman Java hanya sedikit di FILKOM UB sehingga penulis hanya mendapatkan 2 dosen yang bersedia untuk menjadi *tester*. Sedangkan untuk mahasiswa, penulis membatasi kepada 10 *tester* yang bersedia.

6.2.1.2 *System Usability Scale* (SUS)

Pengujian *usability* dengan menggunakan metode SUS dimana penulis memberikan 10 kriteria pertanyaan item SUS kepada pengguna saat mencoba aplikasi dan akan dijawab oleh pengguna setelah mencoba aplikasi final. Daftar pertanyaan yang akan diberikan kepada pengguna terdapat pada Tabel 6.28.

Tabel 6.28 Daftar Pertanyaan Kuisioner SUS

No	Pertanyaan
1	Saya akan menggunakan aplikasi pembelajaran untuk mendukung mahasiswa dalam belajar Pemrograman Java (BelajarJava) untuk mempelajari bahasa pemrograman Java.
2	Saya menilai sistem ini menyediakan alur yang kompleks.
3	Saya menilai sistem ini mudah untuk digunakan.
4	Saya membutuhkan bantuan orang lain untuk menggunakan sistem ini.
5	Saya menilai sistem ini menyediakan fungsi yang saling berintegrasi dengan baik.
6	Saya merasa sistem ini mengandung banyak hal yang tidak konsisten.
7	Saya merasa banyak orang yang akan dengan mudah menggunakan sistem ini untuk mempelajari Bahasa Pemrograman Java.
8	Saya menilai sistem ini sangat rumit untuk digunakan.
9	Saya merasa dapat menggunakan sistem ini dengan baik.
10	Saya butuh belajar lebih banyak untuk menggunakan sistem ini dengan baik.

Hasil dari pengujian *usability* ini terdapat pada Tabel 6.29 dan Tabel 6.30.

Tabel 6.29 Hasil Perhitungan Pengujian SUS pada Mahasiswa

Responden	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Skor SUS
M1	4	1	5	1	5	1	5	1	5	1	97.5
M2	4	3	4	2	3	2	4	2	5	2	72.5
M3	4	1	4	1	2	3	3	2	4	1	77.5
M4	5	3	4	2	4	2	4	2	4	2	75
M5	4	3	5	1	4	2	4	2	5	1	82.5
M6	5	1	5	2	4	2	4	1	5	3	85
M7	4	2	5	1	4	2	5	1	5	1	90
M8	5	1	5	1	5	5	5	1	5	1	92.5
M9	4	1	5	1	4	1	4	1	5	1	92.5
M10	5	3	4	2	3	3	5	2	4	2	75
									Total Skor SUS		840
									Rata-Rata Skor SUS		84

Tabel 6.30 Hasil Perhitungan Pengujian SUS pada Dosen

Responden	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Skor SUS
D1	4	1	4	2	4	2	4	1	4	3	77.5
D2	3	2	4	3	3	3	4	2	4	3	62.5
									Total Skor SUS		140
									Rata-Rata Skor SUS		70

6.3 Analisis Hasil Pengujian

Hasil pengujian kemudian dilakukan analisis untuk menarik kesimpulan. Analisa dilakukan kepada semua hasil pengujian yang telah dilakukan.

6.3.1 Analisis Hasil *Test Driven Development* (TDD)

Berdasarkan pengujian menggunakan teknik TDD, disimpulkan bahwa TDD yang dilakukan terhadap algoritma yang diuji telah sesuai dengan skenario yang telah dirancangkan sebelumnya dan dapat di implementasikan menjadi kode program.

6.3.2 Analisis Hasil Pengujian Unit

Dari hasil pengujian unit terhadap kode program yang telah diimplementasikan dari hasil TDD, didapatkan kesimpulan bahwa modul unit yang diimplementasikan dan telah diujikan sudah memenuhi daftar kebutuhan.

6.3.3 Analisis Hasil Pengujian Validasi

Berdasarkan pengujian validasi yang telah dilakukan, hasil uji dinyatakan valid jika sesuai dengan skenario sistem. Hasil uji dinyatakan tidak valid jika tidak sesuai dengan skenario sistem, tidak sesuai dengan kebutuhan fungsional, atau terdapat *error* atau *bug* yang terjadi. Hasil dari pengujian validasi dapat diketahui bahwa sistem yang telah dirancang sesuai dengan kebutuhan dan tidak ditemukan adanya *error* atau *bug* yang terjadi.

6.3.4 Analisis Hasil Pengujian *Usability*

Berdasarkan pengujian *usability*, hasil yang didapatkan ketika diujikan ke kelompok mahasiswa didapatkan nilai rata-rata SUS sebesar 84 yang termasuk dalam kategori diterima. Sedangkan pengujian yang dilakukan kepada kelompok dosen didapatkan nilai sebesar 70 yang termasuk dalam kategori diterima. Dengan nilai rata-rata SUS sebesar 84 pada sisi mahasiswa dan 70 pada sisi dosen, maka dapat disimpulkan bahwa Aplikasi Pembelajaran untuk Mendukung Mahasiswa Belajar Pemrograman Java Berbasis Android mudah digunakan dan dapat diterima oleh pengguna.

BAB 7 PENUTUP

Bab ini akan menjelaskan konklusi dari penelitian yang telah dijalani dan merupakan bagian akhir laporan. Bab ini akan menjelaskan kesimpulan dari penelitian yang telah dijalani dan memberikan saran kepada penelitian selanjutnya yang mempunyai lingkup pembahasan yang sama.

7.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dijalankan, maka diambillah beberapa kesimpulan sebagai berikut :

1. Berdasarkan hasil pengujian validasi yang telah dilakukan, aplikasi *BelajarJava* dinyatakan valid karena telah sesuai dengan skenario pengujian sistem.
2. Berdasarkan hasil pengujian *usability* yang telah dilakukan, didapatkan hasil skor rata-rata SUS sebesar 84 pada mahasiswa dan 70 pada dosen dimana skor tersebut termasuk dalam kategori diterima. Kesimpulannya, Aplikasi Pembelajaran untuk Mendukung Mahasiswa Belajar Pemrograman Java atau BelajarJava mudah digunakan dan dapat diterima oleh pengguna.

7.2 Saran

Berdasarkan hasil penelitian yang telah dijalankan, penulis menuliskan saran untuk penelitian selanjutnya yang mempunyai lingkup pembahasan yang sama sebagai berikut :

1. Pada fungsi lihat data mahasiswa, sebaiknya ditambahkan fungsi untuk pencarian data mahasiswa.
2. Penambahan materi pembahasan pada konten materi, contoh *source code*, dan latihan.
3. Penelitian selanjutnya dapat membahas implementasi sistem pembelajaran dengan menggunakan *platform* yang berbeda.
4. Penelitian selanjutnya dapat membahas implementasi pembelajaran bahasa pemrograman yang lain dikarenakan kebutuhan akan pemrograman tidak hanya Bahasa Java saja, dan preferensi bahasa yang dipakai para *programmer* berbeda-beda.
5. Penelitian selanjutnya dapat membahas implementasi M-Learning pada perangkat lunak dengan lingkungan sistem operasi yang berbeda, seperti perangkat yang memakai sistem operasi iOS atau implementasi *M-Learning* dengan menggunakan *Mobile Web*.
6. Penelitian selanjutnya dapat menggunakan metode pengembangan aplikasi *mobile* yang lain sehingga dapat dapat digunakan sebagai referensi untuk membandingkan penelitian yang telah dilakukan dengan penelitian selanjutnya.

DAFTAR REFERENSI

Abrahamsson, P., Hanhineva, A. & Hulkko, H., 2004. *Mobile-D : An Agile Approach for Mobile Application Development*. Companion to the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming, System, Languages, and Applications (OOPSLA). Vancouver, Canada, 24-28 Oktober 2004. pp. 174–175.

Adamchik, V.S., 2009. *Array Data Structure*. [Online] Tersedia di : <https://www.cs.cmu.edu/~adamchik/15-121/lectures/Arrays/arrays.html> [Diakses 1 Oktober 2018]

Al-Fahad, F.N., 2009. *Students' Attitudes and Perceptions Towards the Effectiveness of Mobile Learning in King Saud University, Saudi Arabia*. The Turkish Online Journal of Educational Technology (TOJET), 8(2), pp. 111-119.

Ambler, S., 2013. *Introduction to Test Driven Development (TDD)*. [Online] Tersedia di : <http://agiledata.org/essays/tdd.html#WhatIsTDD> [Diakses 10 Oktober 2018]

Bangor, A., Kortum, P. & Miller, J. 2009. *Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale*. Journal of Usability Studies (JUS), 4(3), pp. 114-123.

Brata, K.C., Brata, A.H., 2018. *An Idea of Interactive Japanese Language M-Learning Application to Support Listening and Speaking Exercise*. 2018 International Conference on Sustainable Information Engineering and Technology (SIET). Malang, Indonesia, 10-12 November 2018. pp. 186-191.

Brooke, J., 2011. *SUS - A Quick and Dirty Usability Scale*. [pdf] NSW Agency for Clinical Innovation. Tersedia di : <http://www.tbistafftraining.info/smartphones/documents/b5_during_the_trial_usability_scale_v1_09aug11.pdf> [Diakses 14 Desember 2019]

Brind, S., 2018. *What are Data Structures? Arrays*. STEM, [Online] Tersedia di : <https://owlcation.com/stem/Arrays> [Diakses 1 Oktober 2018]

Collofello, J., S., 1988. *Introduction to Software Verification and Validation*. [pdf] Software Engineering Institute. Tersedia di : <https://apps.dtic.mil/dtic/tr/fulltext/u2/a236117.pdf> [Diakses 13 Desember 2019]

Crompton, H., 2013. *A historical overview of mobile learning: Toward learner-centered education*. [e-book]. Informa UK Limited. Tersedia melalui: Routledge Handbooks Online <https://www.routledgehandbooks.com/> [Diakses 12 November 2019]

DataCamp Incorporated, 2020. *DataCamp - Learn R, Python & SQL Coding* (27.0.0). [program komputer] Google Play. Tersedia di: <https://play.google.com/store/apps/details?id=com.datacamp&hl=en> [Diakses 22 Februari 2020]

Desantha, D., 2018. *Aplikasi Native yang Sangat Optimal untuk Perangkat Mobile*. [Online] Tersedia di : <http://www.plimbi.com/article/169105/aplikasi-native-yang-sangat-optimal-untuk-perangkat-mobile> [Diakses 21 Maret 2018]

Fajar, R., 2017. *Peluang-Peluang Menjadi Java Developer*. [Online] Tersedia di : <https://www.codepolitan.com/peluang-peluang-menjadi-java-developer> [Diakses 1 September 2018]

Firmansyah, Yulianto, A. & Wigandi, D.P., 2018. *Implementasi Mobile-D dalam Pengembangan Aplikasi Mobile Berbasis Android*. Seminar Nasional Inovasi dan Tren (SNIT). Bekasi, Indonesia, 25 Juli 2018. pp. A 1-6.

Flora, H. K., Chande, S.V., 2013. *A Review and Analysis on Mobile Application Development Processes Using Agile Methodologies*. International Journal of Research in Computer Science (IJORCS), 3(4), pp. 9-18.

Fodor, A.G., Covaci, B.V, 2016. *eLearning Mobile App for Android and Ios "English Grammar Learn&Test"*. Database Systems Journal, 7(2), pp. 10-18.

Google, 2018. *Firebase Authentication*. [Online] (2020-07-01 UTC) Tersedia di: <https://firebase.google.com/docs/auth> [Diakses 16 Juli 2020]

Google, 2018. *Firebase Realtime Database*. [Online] (2020-07-01 UTC) Tersedia di: <https://firebase.google.com/docs/database> [Diakses 16 Juli 2020]

Google LLC, 2020. *Grasshopper: Learn to Code for Free* (2.35.2). [program komputer] Google Play. Tersedia di: <https://play.google.com/store/apps/details?id=com.area120.grasshopper&hl=en> [Diakses 22 Februari 2020]

Hanhineva, A., 2004. *Mobile-D*: *Test-Driven Development*. [pdf] VTT Electronics. Tersedia di: <http://virtual.vtt.fi/virtual/agile/mobile-d_docs/stabilize/2_working_day/tasks/mobiledpattern_tdd.pdf> [Diakses 16 Oktober 2018]

Ihme, T., 2004a. *Mobile-D: Stabilize Phase*. [pdf] VTT Electronics. Tersedia di: <http://agile.vtt.fi/mobiled.html> [Diakses 16 Oktober 2018]

Ihme, T., 2005b. *Mobile-D: Initialize Phase*. [pdf] VTT Electronics. Tersedia di: <http://agile.vtt.fi/mobiled.html> [Diakses 15 Oktober 2018]

Jäälinoja, J.,, Hanhineva, A. & Koskela, J., 2004. *Mobile-D: System Test & Fix*. [pdf] VTT Electronics. Tersedia di: <http://agile.vtt.fi/mobiled.html> [Diakses 16 Oktober 2018]

Jordine, T., Liang, Y. & Ihler, E., 2015. *A Mobile Device Based Serious Gaming Approach for Teaching and Learning Java Programming*. International Journal of Interactive Mobile Technologies (iJIM), 9(1), pp. 53-59.

Koran Sindo, 2018. *Sudah Waktunya Ada Mata Pelajaran Coding di Sekolah*. Sindo News. [Online] Tersedia di :

<https://nasional.sindonews.com/read/1293593/144/sudah-waktunya-ada-mata-pelajaran-coding-di-sekolah-1522297314> [Diakses 13 Juli 2018]

Koskela, J., Kyllönen, P., 2004. *Mobile D: Productionize*. [pdf] VTT Electronics. Tersedia di: <http://agile.vtt.fi/mobiled.html> [Diakses 16 Oktober 2018]

Kumar, K.N.M., Akhi, K., Gunti, S.K. & Reddy, M.S.P., 2016. *Implementing Smartphone Using Firebase*. International Journal of Research in Engineering and Applied Sciences (IJREAS), 6(10), pp. 193-198.

Melnik, G., Maurer, F. & Chiasson, M., 2006. *Executable Acceptance Test for Communicating Business Requirements: Customer Perspective*. 2006 Agile Development Conference (AGILE'06) Minneapolis, United States, 23-28 Juli 2006. pp. 35-46.

Miller, R., Collins, C., 2002. *Acceptance Testing*. [pdf] RoleModel Software Incorporated. Tersedia di : <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/recursos/Testing05.pdf > [Diakses 14 Desember 2019]

Nielsen, J., 2000. *How Many Test Users in a Usability Study*. [Online] Tersedia di : <https://www.nngroup.com/articles/how-many-test-users/> [Diakses 15 Desember 2019]

Oracle. *What is Java Technology and Why do I Need it?.* [Online] Tersedia di : <https://www.java.com/en/download/faq/whatis_java.xml> [Diakses 24 September 2018]

Rahman, A.F, Kharisma, A.P. & Dewi, R.K., 2017. *Rancang Bangun Aplikasi Geofence Marketing Cafe Berbasis Android Studi Kasus: Ice Ah!*. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 2(3), pp. 978-987.

Rumbaugh, J., Jacobson, I. & Booch, G. 2005. *The Unified Modeling Language Reference Manual Second Edition*. Canada: Addison-Wesley.

Salo, O., Hulkko, H., 2004. *Mobile-D: Explore*. [pdf] VTT Electronics. Tersedia di: <http://agile.vtt.fi/mobiled.html> [Diakses 15 Oktober 2018]

Sedgewick, R., Wayne, K., 2000-2018. *Computer Science: An Interdisciplinary Approach*. United States: Addison-Wesley.

Shanmugapriya, M., Tamilarasi, A., 2011. *Designing An M-Learning Application For a Ubiquitous Learning Environment in the Android Based Mobile Devices Using Web Services*. Indian Journal of Computer Science and Engineering (IJCSE), 2(1), pp. 22-30.

Sommerville, I., 2011. *Software Engineering* 9th Edition. London:Addison-Wesley.

SoloLearn, 2020. *SoloLearn: Learn to Code for Free*. [program komputer] Google Play. Tersedia di: <https://play.google.com/store/apps/details?id=com.sololearn&hl=en> [Diakses 16 Juli 2020]

Spataru, A. C., 2010. *Agile Development Methods for Mobile Applications.* [Online] Tersedia di: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.186.1292&rep=rep1&type=pdf> [Diakses 24 September 2018]

Susanto, H., Widiartin, T. & Pratama, F.H.S., 2016. *Aplikasi Pembelajaran Berbasis Android (E-Learning) di MA.Daruttaqwa Gresik*. Information Technology Journal, Universitas Wijaya Kusuma Surabaya, 2(2), pp. 81-88.

Tillmann, N., Moskal, M. & Halleux, J. 2012. *The Future of Teaching Programming is on Mobile Devices*. 17th ACM Annual Conference on Innovation and Technology in Computer Science Education. Haifa, Israel, Juli 2012. pp. 156-61.

Traxler, J. 2005. *Defining Mobile Learning*. IADIS International Conference Mobile Learning. Qawra, Malta, 2005. pp. 261-266.

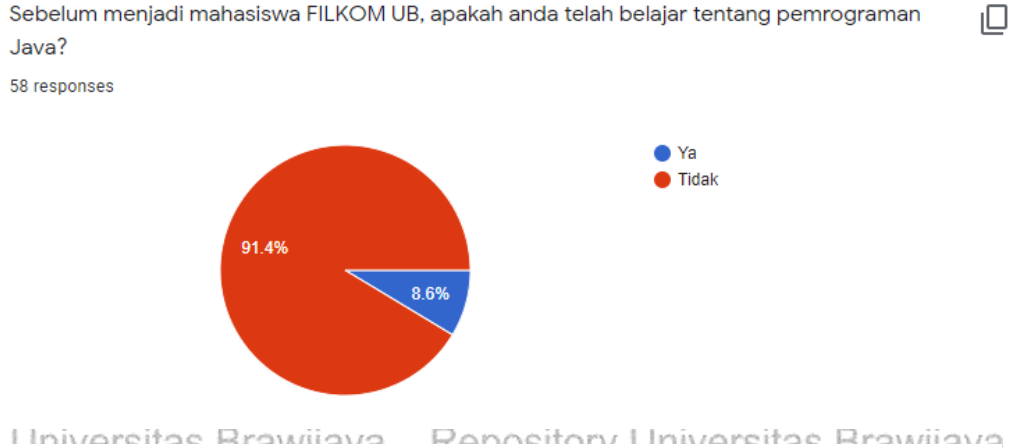
Viswanathan, 2017. *What Is a Mobile Application*. [Online] Tersedia di : <https://www.lifewire.com/what-is-a-mobile-application-2373354> [Diakses 17 April 2018].

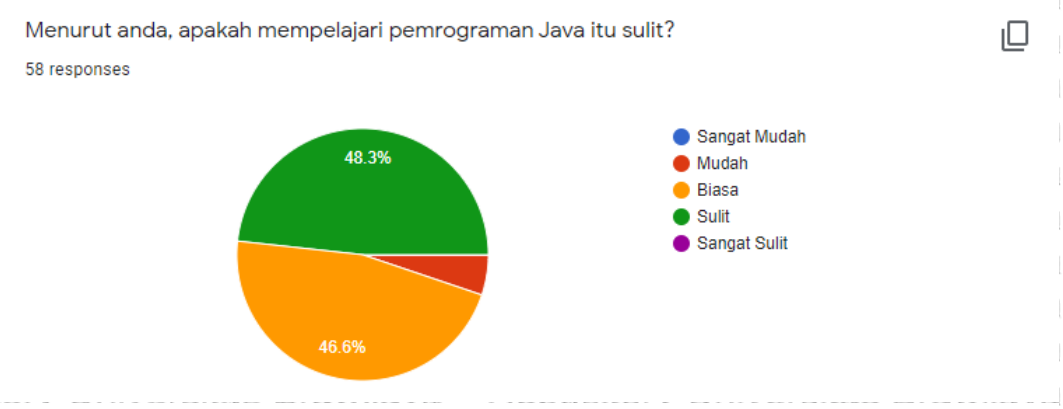
VTT Electronics, 2006. *Portal of Agile Software Development Methodologies*. [Online] Tersedia di : <http://virtual.vtt.fi/virtual/agile/mobiled.html> [Diakses 19 September 2018]

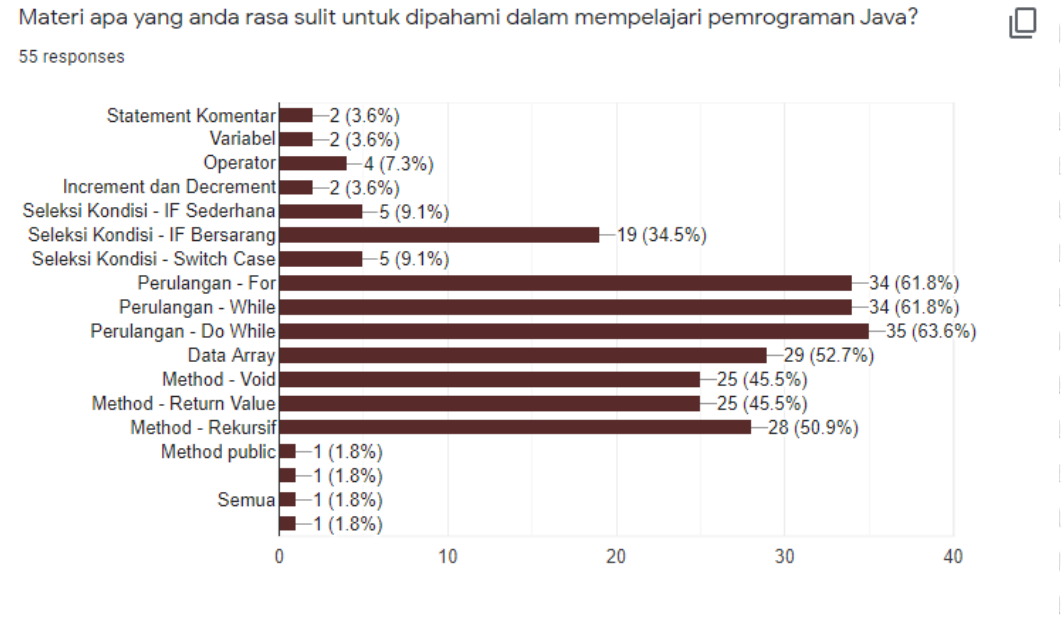
LAMPIRAN A PENGGALIAN KEBUTUHAN

A.1 Penggalian Kebutuhan Mahasiswa

Berikut adalah hasil kuisioner dari 58 Mahasiswa Fakultas Ilmu Komputer Universitas Brawijaya (FILKOM UB) yang penulis berikan kuisioner untuk keperluan penggalian kebutuhan.

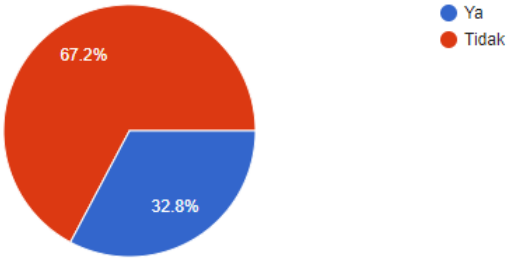






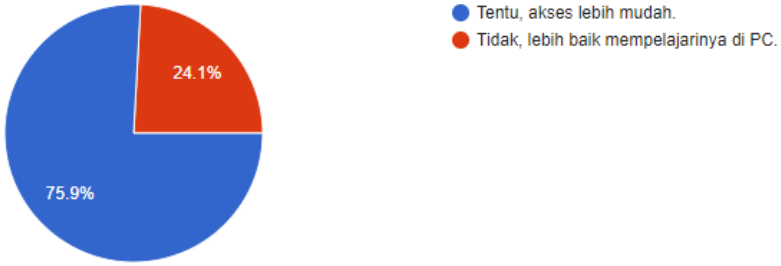
Apakah anda pernah menggunakan tools/software belajar Pemrograman Java?

58 responses



Apakah anda setuju jika mempelajari pemrograman di perangkat mobile akan meningkatkan kemampuan pemrograman anda?

58 responses

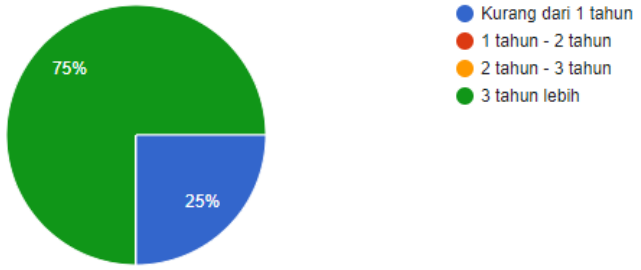


A.2 Penggalian Kebutuhan Dosen

Berikut adalah hasil kuisioner dari 4 Dosen Fakultas Ilmu Komputer Universitas Brawijaya (FILKOM UB) yang penulis berikan kuisioner untuk keperluan penggalian kebutuhan.

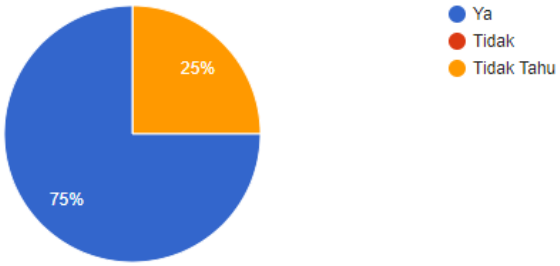
Sudah berapa lama Bapak/Ibu mengajar mata kuliah terkait dengan Pemrograman Java?

4 responses



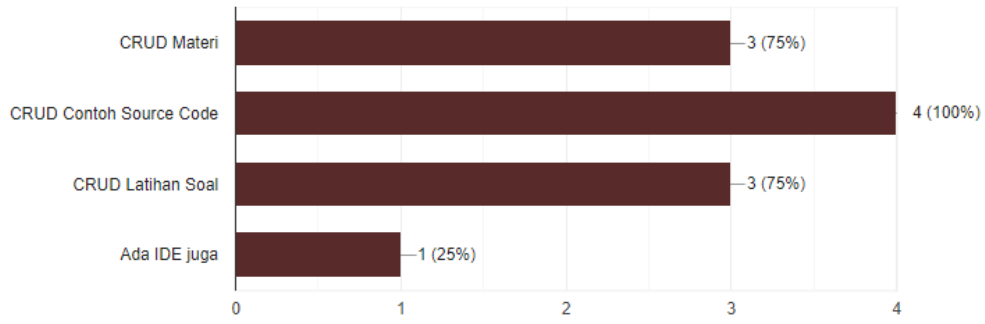
Apakah Bapak/Ibu sepakat jika media Mobile Learning (M-Learning) diimplementasikan untuk mendukung dalam proses pembelajaran Pemrograman Java?

4 responses



Jika sepakat, fitur apa yang Bapak/Ibu harapkan pada aplikasi M-Learning tersebut?

4 responses



Menurut bapak/Ibu, materi apa yang dirasa sulit untuk dipahami oleh mahasiswa?

4 responses