RANCANG BANGUN APLIKASI PERANGKAT BERGERAK SISTEM PENERIMAAN LOWONGAN PADA KERJA PADA *MOBILE* BERBASIS ANDROID

SKRIPSI

LABORATORIUM
PEMROGRAMAN APLIKASI PERANGKAT BERGERAK

Untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer

Disusun Oleh :
HUTAMI RISTIANI
NIM. 105060807111023

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
UNIVERSITAS BRAWIJAYA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER
MALANG
2015

LEMBAR PERSETUJUAN

RANCANG BANGUN APLIKASI PERANGKAT BERGERAK SISTEM PENERIMAAN LOWONGAN KERJA PADA *MOBILE* BERBASIS ANDROID

SKRIPSI

LABORATORIUM

PEMROGRAMAN APLIKASI PERANGKAT BERGERAK

Untuk Memenuhi Sebagian Persyaratan Memperoleh Gelar Sarjana Komputer

Disusun Oleh :

HUTAMI RISTIANI

NIM. 105060807111023

Skripsi ini telah disetujui oleh dosen pembimbing pada tanggal 24 Juni 2015

Dosen Pembimbing I	**Dosen Pembimbing II**
Dr. Eng. Herman Tolle, ST., MT	**Aryo Pinandito, ST, M.MT**
NIP. 197408232000121 001	**NIP. 198305192014041001**

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI PERANGKAT BERGERAK SISTEM PENERIMAAN LOWONGAN KERJA PADA *MOBILE* BERBASIS ANDROID

SKRIPSI

PEMROGRAMAN APLIKASI PERANGKAT BERGERAK

Untuk Memenuhi Sebagian Persyaratan Memperoleh Gelar Sarjana Komputer

Disusun Oleh :

HUTAMI RISTIANI

NIM. 105060807111023

Setelah dipertahankan di depan Majelis Penguji pada tanggal 24 Juni 2015
dan dinyatakan memenuhi syarat untuk memperoleh gelar Sarjana dalam bidang
Ilmu Komputer

Penguji I	**Penguji II**
Agi Putra Kharisma, ST., M.T.	**Denny Sagita Rusdianto, S.Kom., M.Kom**
NIK. 2013048604301001	**NIK. 85112406110250**

Penguji III

Wibisono Sukmo Wardhono, ST., MT

NIK. 82040406110091

Mengetahui,
Ketua Program Studi Informatika / Ilmu Komputer

Drs. Marji, M.T.

NIP. 19670801 199203 1 001

PERNYATAAN
ORISINALITAS SKRIPSI

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 dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

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).

Undang-undang No. 20 Tahun 2003 Pasal 25 ayat 2 yang berisi "Lulusan perguruan tinggi yang karya ilmiahnya digunakan untuk memperoleh gelar akademik, profesi, atau vokasi terbukti merupakan jiplakan dicabut gelarnya." Dan Pasal 70 yang berisi "Lulusan yang karya ilmiah yang digunakannya untuk mendapatkan gelar akademik, profesi, atau vokasi sebagaimana dimaksud dalam Pasal 25 ayat (2) terbukti merupakan jiplakan dipidana dengan pidana penjara paling lama dua tahun dan/atau pidana denda paling banyak Rp. 200.000.000,00 (dua ratus juta rupiah)."

Malang, 24 Juni 2015
Mahasiswa,

Hutami Ristiani
NIM. 105060807111023

KATA PENGANTAR

Puji dan syukur Penulis panjatkan kehadirat Allah SWT, karena hanya dengan rahmat dan bimbingannya Penulis dapat menyelesaikan skripsi dengan judul "Rancang Bangun Aplikasi Perangkat Bergerak Sistem Penerimaan Lowongan Kerja Pada *Mobile* Berbasis Android" dengan baik. Tanpa rahmat dan bimbingan dari Tuhan Yang Maha Esa, maka niscaya Penulis tidak akan dapat menyelesaikan skripsi ini dengan baik dan tepat waktu.

Terima kasih pula Penulis sampaikan kepada pihak-pihak yang telah membantu Penulis dalam penyelesaian skripsi ini. Pihak-pihak tersebut antara lain:

1. Herman Tolle, Dr. Eng., ST., MT. dan Aryo Pinandito, ST., M.MT. selaku dosen pembimbing I dan dosen pembimbing II yang telah banyak memberikan bimbingan dan menyalurkan ilmu kepada penulis serta nasehat yang telah diberikan dalam proses pengerjaan skripsi ini.
2. Orang tua Penulis, Karso dan Yayuk Windarti yang telah memberi motivasi, kasih sayang serta dukungan moril dan materil kepada Penulis. Serta kakak saya tercinta Titi Riansari yang telah memberikan semangat dari awal sampai akhir pengerjaan skripsi ini.
3. Bapak R. Arief Setyawan, ST., MT. selaku dosen pembimbing akademik yang telah memberikan bimbingan, ilmu dan saran selama penulis belajar.
4. Teman-teman yang tersayang Amelinda Putri Rizki, Nurwidyaningtyas, Fahima Rahmasari, Achmad Ghozainul Alfa, Gian Rofi, Aga Saputra, Dedy Kurniawan, dan Rizky Bagus Ananta yang selalu memberikan saran, semangat dan dorongan dalam mengerjakan skripsi ini.
5. Semua teman-teman Informatika angkatan 2009, 2010 dan 2011 lainnya terimakasih atas segala bantuannya selama menjadi mahasiswa.
6. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya skripsi ini.

Semoga jasa dan amal baik mendapatkan balasan dari Allah SWT. Ibarat tak ada gading yang tak retak, dengan segala kerendahan hati, penulis menyadari sepenuhnya bahwa skripsi ini masih jauh dari kata sempurna. Oleh karena itu kritik dan saran untuk kesempurnaan skripsi ini, senantiasa penulis harapkan dari berbagai pihak.

Malang, 24 Juni 2015

Penulis

ABSTRAK

Hutami Ristiani. 2015. Rancang Bangun Aplikasi Perangkat Bergerak Sistem Penerimaan Lowongan Kerja Pada *Mobile* Berbasis Android. Fakultas Ilmu Komputer, Universitas Brawijaya, Malang. Dosen Pembimbing : Dr. Eng. Herman Tolle, ST., MT. dan Aryo Pinandito, S.T., M.MT.

Semakin banyak jumlah lulusan perguruan tinggi semakin banyak pula lapangan pekerjaan yang dibutuhkan masyarakat saat ini. Serta peluang mendapatkan pekerjaan di dunia bisnis sekarang semakin kecil. Banyaknya pencari kerja serta minimnya info-info tentang lapangan pekerjaan, membuat para pencari kerja terlambat mengetahui atau bahkan melewatkan lowongan pekerjaan yang ada. Saat ini, proses advertising lowongan pekerjaan masih melalui media koran, website, radio, serta proses lamaran pekerjaan dilakukan secara manual. Sehingga para pelamar kerja membutuhkan banyak waktu untuk mengirimkan berkas lamaran pekerjaan melalui media e-mail maupun pos ke beberapa penyedia lowongan kerja.

Dari penjabaran permasalahan yang dialami oleh para pencari kerja tersebut, dapat disimpulkan hal ini dianggap tidak efisien dan terlalu banyak menyita waktu para pelamar kerja. Sehingga para pencari kerja maupun penyedia lowongan kerja memerlukan aplikasi yang mampu menangani masalah mulai dari proses mengunggah iklan lowongan pekerjaan yang dilakukan oleh penyedia lowongan kerja dan iklan tersebut dapat diunduh oleh pencari kerja, hingga pencari kerja dapat mengirimkan CV secara langsung di aplikasi secara efisien. Sistem ini menggunakan beberapa pengujian diantaranya menggunakan sistem pengujian blackbox yang mengamati tampilan luar dan fungsionalitas dari sistem.

Kata kunci : Lapangan pekerjaan, Lowongan pekerjaan, Black Box

ABSTRACT

Hutami Ristiani. 2015. Design of Mobile Devices System Application Acceptance Jobs On Mobile-Based Android. Faculty of Computer Science, Brawijaya University, Malang. Adviser : Dr. Eng. Herman Tolle, ST., MT. and Aryo Pinandito, S.T., M.MT.

The more count of college graduates the more too jobs needed by society today. And the opportunities to get a job in business are now getting smaller. Many job seekers and the lack of information about jobs, make the job seekers out too late knows or even miss the job vacancies there. Currently, the process job vacancies advertising is still through newspapers, websites, radio, and job application was done manually. So the job seekers need a lot of time to send a file job applications via e-mail or post to some employeer.

From the explanation of the problems endured by job seekers could conclude this is considered inefficiently and over much time-consuming job seekers. So job seekers and employeer need applications that are able to handle problems from job advertisements uploading process done by employeer and it can be downloaded by job seekers, until the job seekers can send their CV directly in applications efficiently.This system uses several tests them using blackbox testing system that observe outer appearance and functionality of the system.

Keywords : Jobs, Job vacancy, Black Box

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE PROGRAM

BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring dengan perkembangan zaman yang semakin pesat, kebutuhan akan efektifitas dan efisiensi sangat diutamakan dalam berbagai bidang. Hal tersebut telah mendorong manusia untuk berkreasi dan berinovasi dalam bidang teknologi untuk menciptakan suatu alat yang lebih efektif dan efisien. Perkembangan teknologi saat ini dapat dilihat sudah banyak alat yang diciptakan agar memberikan kemudahan pada masyarakat dalam melaksanakan pekerjaan.

Semakin banyak jumlah lulusan perguruan tinggi semakin banyak pula lapangan pekerjaan yang dibutuhkan masyarakat saat ini. Serta peluang mendapatkan pekerjaan di dunia bisnis sekarang semakin kecil. Banyaknya pencari kerja serta minimnya info-info tentang lapangan pekerjaan, membuat para pencari kerja terlambat mengetahui atau bahkan melewatkan lowongan pekerjaan yang ada. Saat ini, proses *advertising* lowongan pekerjaan masih melalui media koran, *website*, radio, proses lamaran pekerjaan juga masih dilakukan dengan cara membuat surat lamaran kerja dan CV yang kemudian dikirimkan melalui pos ataupun dikirim sendiri. Apabila sebuah prusahaan penyedia lowongan kerja ingin membuat beberapa iklan lowongan kerja atau seorang pencari kerja ingin melamar pekerjaan ke beberapa perusahaan pasti akan banyak menyita waktu dan biaya dalam prosesnya.

Dalam upaya untuk mengurangi angka pengangguran, peran layanan penyedia informasi lowongan kerja sangat besar. Layanan penyedia informasi ini selalu dituntut untuk dapat memberikan infomasi yang benar secara efektif dan efisien. *Mobile* merupakan teknologi yang telah berkembang menjadi kebutuhan primer manusia, suatu teknologi yang selalu dibutuhkan dan dibawa kemanapun seseorang itu berada. Dengan sifatnya yang *portable* inilah teknologi *mobile* sesuai digunakan untuk suatu layanan penyedia lowongan kerja karena dapat bekerja lebih efektif dan efisien. Para penyedia lowongan kerja dapat menyalurkan iklan lowongan kerja yang sedang dibutuhkan kepada para pencari kerja. Selain menyalurkan iklan, dengan fitur-fitur yang dimiliki perangkat *mobile*

layanan penyedia lowongan kerja juga dapat menyediakan suatu layanan untuk menerima dan melihat CV seorang pelamar kerja, sehingga para pencari kerjapun dapat mencari lowongan kerja serta mengirimkan CV/berkas lamaran kerja secara lebih mudah. Dengan fitur tersebut proses pendaftaran dan seleksi awal para pelamar kerja dapat berlangsung lebih mudah dan efisien.

Sekarang ini banyak jenis sistem operasi yang mendukung perangkat *mobile*, salah satu sistem operasi yang mendukung perangkat *mobile* adalah Android. Penggunaan perangkat *mobile* berbasis android di masyarakat semakin lama semakin banyak [LUA-14]. Hal inilah yang melatar belakangi dibuatnya suatu **aplikasi perangkat bergerak sistem penerimaan lowongan kerja pada *mobile* berbasis android**. Sebuah aplikasi yang dapat memberikan layanan informasi terkait lowongan pekerjaan sekaligus dapat melakukan pendaftaran dengan cara mengirimkan langsung *curriculum vitae* kepada penyedia lowongan kerja tersebut.

1.2 Rumusan Masalah

Sesuai dengan latar belakang masalah di atas, maka rumusan masalah pada penelitian ini adalah:

1. Bagaimana menganalisis kebutuhan aplikasi sistem penerimaan lowongan kerja pada perangkat *mobile*?
2. Bagaimana mengimplementasikan aplikasi *mobile* yang dapat memberi kemudahan bagi para pemberi lowongan kerja dalam melakukan seleksi awal para pelamar kerja dengan membangun sistem ini?
3. Bagaimana mengukur validitas dan *usability* sistem penerimaan lowongan kerja pada perangkat *mobile*?

1.3 Batasan Masalah

Berdasarkan rumusan masalah penelitian di atas, maka batasan masalah dalam pembuatan sistem ini sebagai berikut:

1. Aplikasi penerimaan lowongan kerja diimplementasikan pada perangkat bergerak berbasis android.

2. Iklan lowongan kerja yang tersedia pada *database* server aplikasi diunggah langsung oleh perusahaan/penyedia lowongan kerja tanpa diverifikasi admin terlebih dahulu.

1.4 Tujuan

Mendesain, menguji, menganalisis, serta mengimplementasikan aplikasi perangkat bergerak sistem penerimaan lowongan kerja *mobile* berbasis android sehingga menjadi sebuah sistem yang dapat memberikan informasi berupa lowongan pekerjaan dari perusahaan.

1.5 Manfaat

Penelitian yang telah dilakukan ini diharapkan dapat memberikan manfaat bagi pihak-pihak yang berkepentingan, antara lain yaitu:

- **Bagi Pencari Kerja atau *Jobseeker***

Bagi masyarakat yang sedang membutuhkan pekerjaan untuk mendapatkan info-info mengenai berbagai macam lowongan pekerjaan terkini (*up-to-date*) serta dapat juga mengirimkan langsung profil biodata terhadap perusahaan yang diminati.

- **Bagi Perusahaan atau *Employeer***

Bagi Perusahaan atau *Employeer* yang membutuhkan dan/atau mencari tenaga untuk pegawai baru guna mempertemukan pencari kerja dengan pengguna tenaga kerja yang lebih efektif dan efisien, karena berbagai kriteria yang diminta pengguna tenaga kerja bisa dipilih sesuai dengan kriteria yang dimiliki oleh pencari kerja maupun para penyedia kerja.

1.6 Sistematika Penulisan

BAB I : Pendahuluan

Bab pendahuluan berisi hal-hal sebagai berikut : Latar belakang, menjelaskan rasional atau justifikasi penelitian dilihat dari latar belakang pemilihan permasalahan yang dibahas. Perumusan Masalah, dirumuskan secara lugas dan jelas. Batasan Masalah, dibuat sesuai dengan ruang lingkup pembahasan yang akan dilakukan. Tujuan,

menyatakan target penelitian yang akan dicapai. Manfaat, berisi manfaat yang diperoleh bagi *stakeholder* yang terlibat. Sistematika Penulisan, berisi tentang sistematika penulisan laporan skripsi.

BAB II : Kajian Pustaka dan Dasar Teori

Bagian ini berisi analisis berbagai teori dan hasil penelitian yang relevan dengan masalah yang akan dibahas. Dalam bagian ini melakukan sintesis terhadap teori yang relevan dengan permasalahan tentang android.

BAB III : Metodologi Penelitian

Bab ini berisi tentang metodologi penelitian dan perancangan dari sistem yang dibuat. Metodologi berupa metode-metode yang digunakan seperti studi literatur dan metode pengujian.

BAB IV : Analisis dan Perancangan

Perancangan pada bab ini dibagi menjadi dua yaitu Analisis kebutuhan dan perancangan lunak. Analisis kebutuhan direpresentasikan dengan menggunakan *use case* dan *activity* diagram. Sedangkan untuk perancangan perangkat lunak, berdasarkan pada analisis kebutuhan akan dirancang arsitektur sistem, perancangan basis data, diagram *sequence*, *class* diagram, dan perancangan antarmuka.

BAB V : Implementasi dan Pengujian

Pada bab ini membahas tentang implementasi perangkat lunak. *Operating system* yang digunakan disini yaitu Android, serta prosedur-prosedur dilakukan pada pembuatan perangkat lunak tersebut.

Memuat teknik-teknik serta metode-metode pengujian yang dapat dilakukan pada sistem yang telah dibuat.

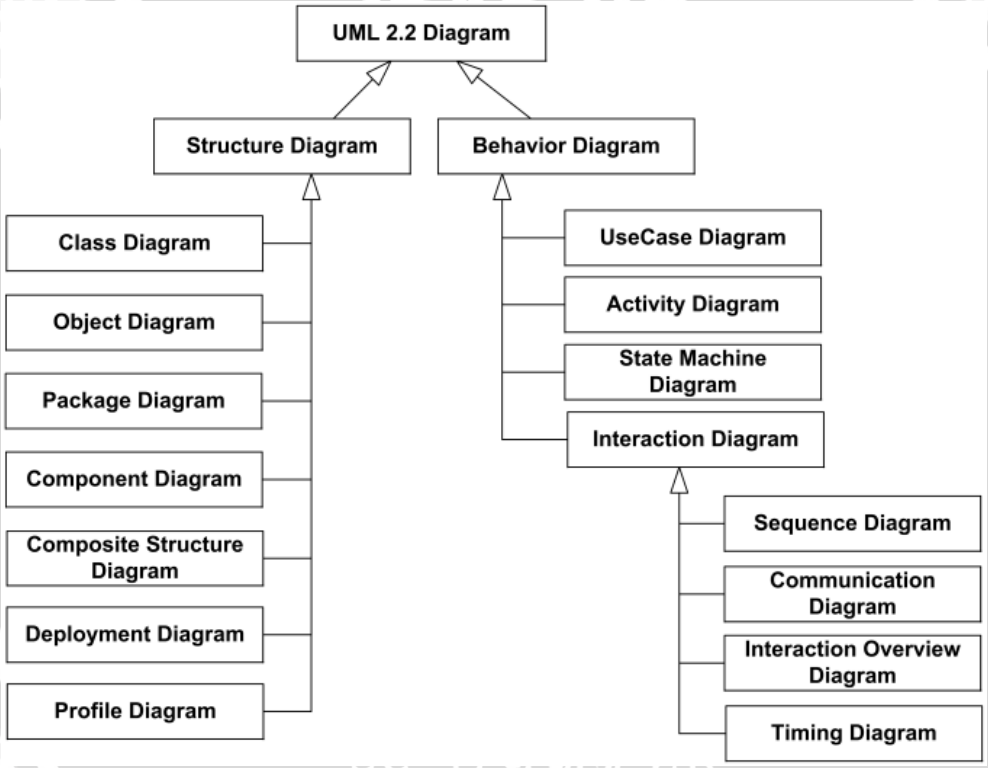
BAB VI : Penutup

Berisi Kesimpulan dan Saran. Dalam bagian ini menyimpulkan hasil secara tegas dan lugas, sesuai dengan permasalahan. Setelah hasil penelitian disimpulkan, juga harus mampu memberikan saran yang operasional.

BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 *Unified Modelling Language*

Unified Modelling Language (UML) merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram teks–teks pendukung UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan mendokumentasikan dari sistem perangkat lunak [ASR-11].



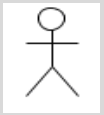
Gambar 2.1 Diagram Unified Modeling Language
Sumber : [ASR-11]

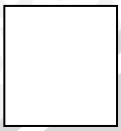
Structure diagram yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem. *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem [ASR-11].

2.1.1 *Use Case Diagram*

Usecase diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

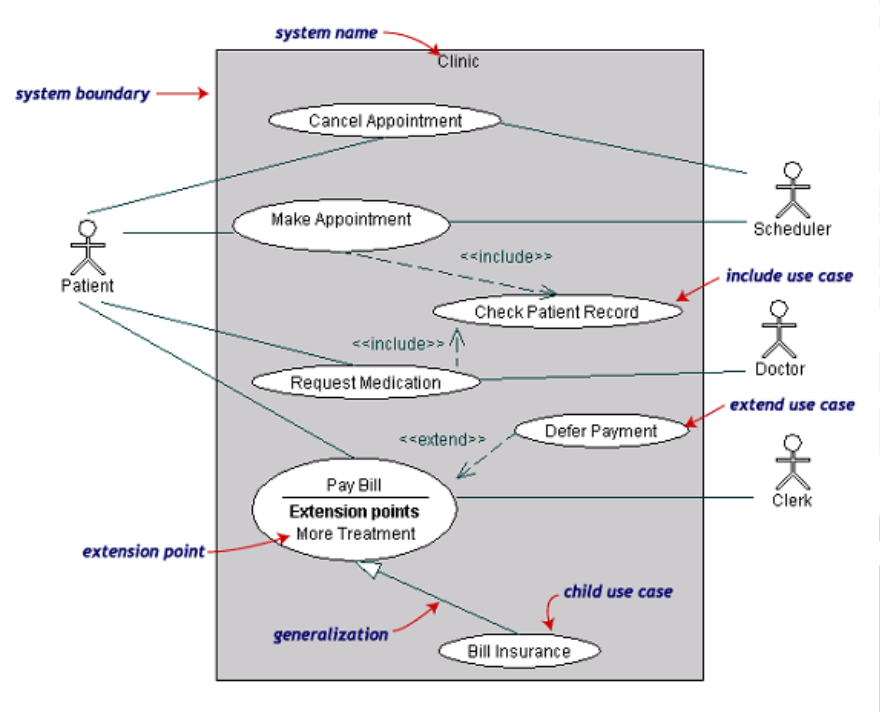
Tabel 2.1 Keterangan Simbol – Simbol *Use Case* Diagram
Sumber : [ASR-11]

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya sebagai elemen yang tidak mandiri (*independent*).
3		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya (objek induk / *ancestor*).
4		*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.

5		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu *actor*.
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal.

Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extenduse case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.



Gambar 2.2 Contoh *Use Case Diagram*
Sumber : [ASR-11]

2.1.2 *Class* Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok :

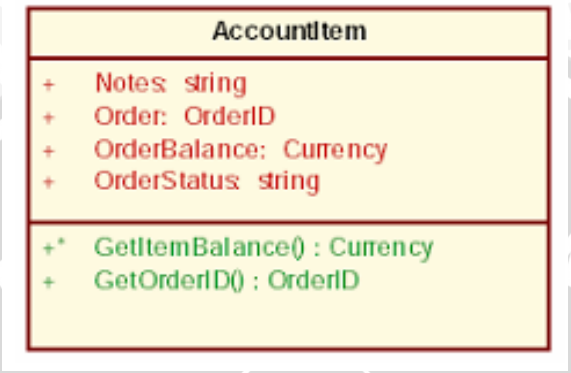
1. Nama (dan *stereotype*)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

- *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
- *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.

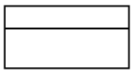
- *Public*, dapat dipanggil oleh siapa saja

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*.



Gambar 2.3 Contoh sebuah *Class*
Sumber : [ASR-11]

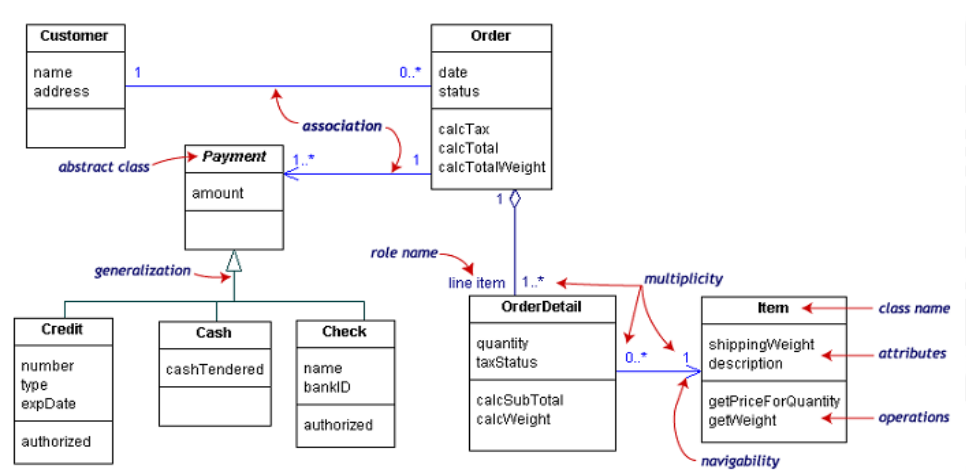
Tabel 2.2 Keterangan Simbol - Simbol *Class* Diagram
Sumber : [ASR-11]

NO	GAMBAR	NAMA	KETERANGAN
1		*Generalization*	Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya (objek induk / *ancestor*).
2		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang

			menghasilkan suatu hasil yang terukur bagi suatu aktor.
5		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
6		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
7		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
8	1..*, 0...*, 1...1	*Multiplicity*	Menunjukkan jumlah objek yang diperbolehkan menurut logika.

Hubungan Antar *Class*

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan class yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas..").
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari class lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari class yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan sequence diagram yang akan dijelaskan kemudian.



Gambar 2.4 Contoh *Class Diagram*
Sumber : [ASR-11]

2.1.3 Activity Diagram

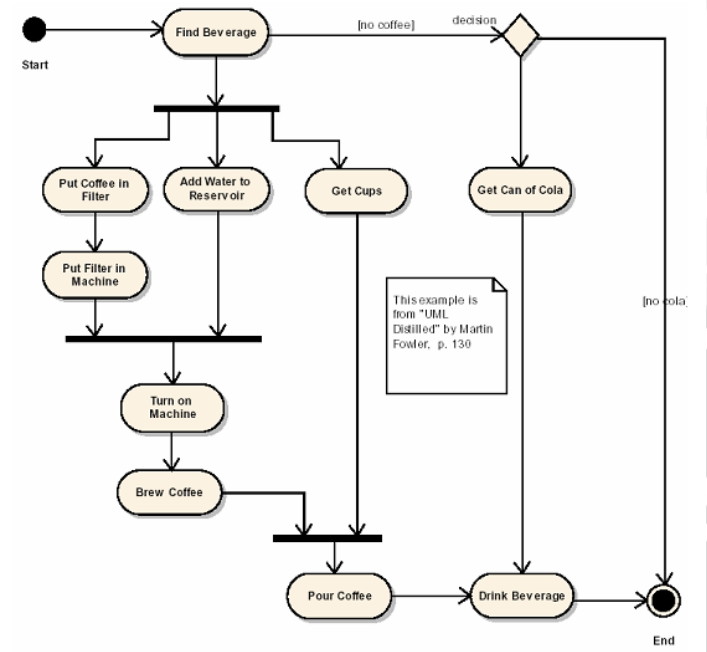
Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state* diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity* diagram tidak menggambarkan *behaviour* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu*use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa

object swimlane untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.



Gambar 2.5 Contoh Activity Diagram Tanpa Swimlane
Sumber : [ASR-11]

Tabel 2.3 Keterangan simbol - simbol *Activity Diagram*

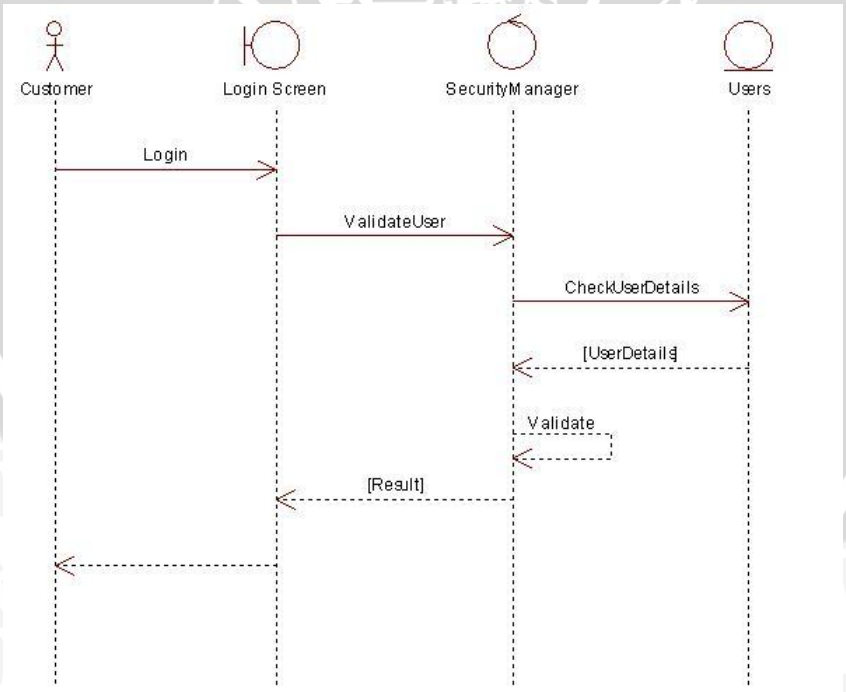
NO	GAMBAR	NAMA	KETERANGAN
1		*Activity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi.
3		*Initial Node*	Bagaimana objek dibentuk atau diawali.
4		*Activity Final Node*	Bagaimana objek dibentuk dan dihancurkan.
5		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran.
6		*Decision*	Menunjukkan proses pemilihan keputusan / seleksi kondisi.

2.1.4 *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

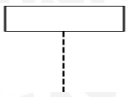
Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai *respons* dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary*, *controller* dan *persistent entity*.



Gambar 2.6 Contoh *Sequence Diagram*
Sumber : [ASR-11]

Tabel 2.4 Keterangan simbol - simbol *Sequence Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		*Life Line*	Objek *entity*, antarmuka yang saling berinteraksi.
2		*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi
3		*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi
4		*Activation*	Menunjukkan saat dimana *entity* diaktifkan di sepanjang *Life Line.*

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary*, *controller* dan *persistent entity*.

2.2 *Web Service*

Web service dapat diartikan juga sebuah metode pertukaran data, tanpa memperhatikan dimana sebuah *database* ditanamkan, dibuat dalam bahasa apa sebuah aplikasi yang mengkonsumsi data, dan di *platform* apa sebuah data itu dikonsumsi. *Web service* mampu menunjang interoperabilitas. Sehingga *web service* mampu menjadi sebuah jembatan penghubung antara berbagai sistem yang ada [DKB-12].

Web service adalah aplikasi sekumpulan data (*database*), perangkat lunak (*software*) atau bagian dari perangkat lunak yang dapat diakses secara *remote* oleh berbagai piranti dengan sebuah perantara tertentu. Secara umum, *web service*

dapat diidentifikasikan dengan menggunakan URL seperti hanya web pada umumnya. Namun yang membedakan *web service* dengan web pada umumnya adalah interaksi yang diberikan oleh *web service*. Berbeda dengan URL web pada umumnya, URL *web service* hanya mengandung kumpulan informasi, perintah, konfigurasi atau sintaks yang berguna membangun sebuah fungsi-fungsi tertentu dari aplikasi [GOD-13].

Web pada umumnya digunakan untuk melakukan *respon* dan *request* yang dilakukan antara *client* dan *server*. Sebagai contoh, seorang pengguna layanan *web* tertentu mengetikan alamat *url web* untuk membentuk sebuah *request*. *Request* akan sampai pada *server*, diolah dan kemudian disajikan dalam bentuk sebuah respon. Dengan singkat kata terjadilah hubungan *client-server* secara sederhana [GOD-13].

Sedangkan pada *web service* hubungan antara *client* dan *server* tidak terjadi secara langsung. Hubungan antara *client* dan *server* dijembatani oleh file *web service* dalam format tertentu. Sehingga akses terhadap *database* akan ditanggani tidak secara langsung oleh *server*, melainkan melalui perantara yang disebut sebagai *web service*. Peran dari *web service* ini akan mempermudah distribusi sekaligus integrasi *database* yang tersebar di beberapa *server* sekaligus [GOD-13].

2.3 *Google Cloud Messanging* (GCM)

Google Cloud Messanging (GCM) adalah layanan yang memungkinkan pengembang mengirim data pesan singkat kepada pengguna mereka pada perangkat android tanpa perlu solusi *sync proprietary*.

GCM menangani semua rincian pesan antrian dan memberikan mereka efisiensi untuk perangkat android yang ditargetkan. Mendukung pesan *multicasting* dan dapat mencapai hingga 1000 perangkat terhubung secara bersamaan dengan satu permintaan. Ini juga mendukung pesan muatan yang berarti selain mengirimkan pesan untuk sebuah aplikasi pada perangkat, pengembangjuga dapat mengirim hingga 4Kb data *paload* (sehingga aplikasi seperti *instant messaging* dapat mengkonsumsi pesan langsung). Layanan GCM menangani semua aspek antrian pesan dan pengiriman ke aplikasi target android berjalan pada perangkat target [ABV-14].

2.3.1 Cara Kerja GCM

Pertama aplikasi perlu mengirim “query” (permintaan) untuk server GCM. "Query" ini memerlukan informasi tertentu seperti ID Pengirim dan ID Aplikasi ini (nama paket aplikasi). Dan server GCM ini akan mengirim kembali intent dengan baik pesan gagal atau ID Registrasi.

Kemudian perlu untuk mengirim ID Pendaftaran ke server Web Aplikasi, sehingga Web server dapat menyimpan informasi ini untuk digunakan nanti. Untuk “Tanda Daftar ID” dapat gagal/batal sehingga aplikasi Anda harus mempertimbangkan hal ini. Dari titik pandang aplikasi, selain menerima pesan, tidak ada lagi yang perlu dilakukan. Langkah-langkah lain dari proses ini adalah pada sisi server Web.

Web server kemudian harus menggunakan ID Pengirim dan *password* untuk mengambil Otentikasi ‘Masuk’ Client dari server ‘Masuk’ Google Klien. Dengan ini Web server sekarang memiliki semua informasi yang diperlukan untuk mengirim pesan ke aplikasi Anda.

Web server kemudian akan menggunakan ID Pendaftaran Anda dan Otentikasi Klien Login dengan pesan ke server yang dituju adalah GCM. Setelah ini Google akan menangani mengirimkan pemberitahuan ke aplikasi [RSP-13].

2.4 Pengujian Perangkat Lunak

Pengujian perangkat lunak memerlukan perancangan kasus uji (*test case*) agar dapat menemukan kesalahan dalam waktu singkat dan usaha minimum. Berbagai macam metode perancangan kasus uji telah berevolusi. Metode-metode ini menyediakan *developer* pendekatan sistematis untuk pengujian. Terlebih lagi metode metode ini menyediakan mekanisme yang dapat membantu memastikan kelengkapan dari pengujian dan menyediakan kemungkinan tertinggi untuk menemukan kesalahan kesalahan dalam perangkat lunak [PRE-12]. Teknik atau metode perancangan kasus uji yang digunakan adalah *black-box testing.*

2.4.1 Black-Box Testing

Black-box testing atau *behavioral testing* berfokus pada persyaratan fungsional perangkat lunak [PRE-12]. Dengan demikian, pengujian *black-box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi

input yang sepenuhnya menggunakan semua persyaratan fungsional untuk semua program. Pengujian *black-box* merupakan pendekatan komplementer yang kemungkinan besar mampu mengungkap kelas kesalahan.

Pengujian *black-box* berusaha menemukan kesalahan dalam kategori berikut:

1. Fungsi-fungsi yang tidak benar atau hilang.
2. Kesalahan *interface*.
3. Kesalahan dalam struktur data atau akses basis data eksternal.
4. Kesalahan kinerja.
5. Inisialisasi dan kesalahan terminasi.

Pengujian *black-box* cenderung diaplikasikan selama tahap akhir pengujian. Pengujian *black-box* memperhatikan struktur kontrol, maka perhatian berfokus pada domain informasi.

2.4.2 Strategi Pengujian

Strategi untuk pengujian perangkat lunak mengintegrasikan metode desain *test case* perangkat lunak ke dalam sederetan langkah yang direncanakan dengan baik, dan hasilnya adalah konstruksi perangkat lunak yang berhasil [PRE-12]. Sejumlah strategi pengujian perangkat lunak telah diusulkan di dalam literatur. Strategi pengujian harus mengakomodasi pengujian tingkat rendah yang diperlukan untuk membuktikan bahwa segmen kode sumber yang kecil telah diimplementasikan dengan tepat, demikian juga pengujian tingkat tinggi yang memvalidasi fungsi-fungsi sistem mayor yang berlawanan dengan kebutuhan pelanggan. Proses pengujian dimulai dengan pengujian yang berfokus pada setiap modul secara individual (*unit testing*), dilanjutkan dengan pengujian integrasi (*integration testing*) dan berakhir pada pengujian validasi (*validation testing*) [PRE-12].

2.4.2.1 *Pengujian Validasi*

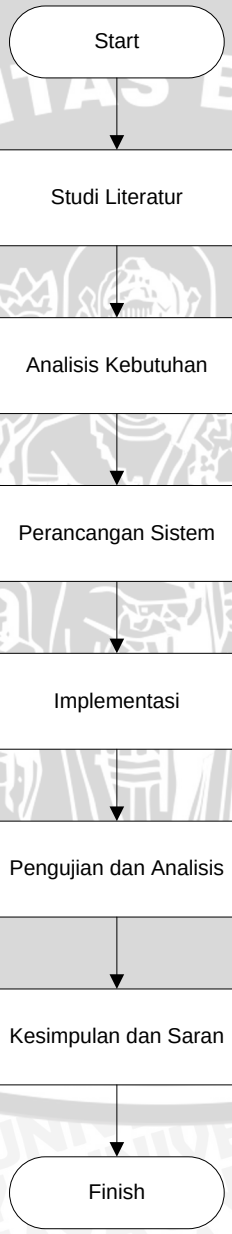
Pada kulminasi pengujian terintegrasi, perangkat lunak secara lengkap dirakit sebagai suatu paket; kesalahan *interfacing* telah diungkap dan dikoreksi, dan seri akhir dari pengujian perangkat lunak, yaitu pengujian validasi dapat dimulai. Validasi dapat ditentukan dengan berbagai cara, tetapi definisi yang

sederhana adalah bahwa validasi berhasil bila perangkat lunak berfungsi dengan cara yang dapat diharapkan secara bertanggung jawab oleh pelanggan. Validasi perangkat lunak dicapai melalui sederetan pengujian *black-box* yang memperlihatkan konformitas dengan persyaratan.

Rencana pengujian menguraikan kelas-kelas pengujian yang akan dilakukan, dan prosedur pengujian menentukan *test case* spesifik yang akan digunakan untuk mengungkap kesalahan dalam konformitas dengan persyaratan. Baik rencana dan prosedur didesain untuk memastikan apakah semua persyaratan fungsional dipenuhi; semua persyaratan kinerja dicapai; dokumentasi betul dan direkayasa oleh manusia: dan persyaratan lainnya dipenuhi (transportabilitas, kompatibilitas, pembetulan kesalahan, maintanabilitas).

BAB III
METODOLOGI PENELITIAN

Pada bab ini dijelaskan langkah-langkah yang akan dilakukan dalam perancangan, implementasi dan pengujian dari aplikasi perangkat lunak yang akan dikembangkan. Kesimpulan dan saran disertakan sebagai catatan atas aplikasi dan kemungkinan arah pengembangan perangkat lunak selanjutnya. Berikut ini merupakan diagram alir runtutan pengerjaan penelitian ini:



Gambar 3.1 Diagram Alir Metode Penelitian

3.1 Studi Literatur

Studi literatur menjelaskan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori-teori pendukung tersebut meliputi:

a. *Unifed Modelling Language*
b. *Web Service*
c. *Google Cloud Messaging*
d. Pengujian Perangkat Lunak
e. *Database* MySQL

3.2 Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan dari sistem yang akan dibangun. Analisis kebutuhan dilakukan dengan mengidentifikasi semua kebutuhan (*requirements*) sistem dan siapa saja yang terlibat di dalamnya. Analisis juga dilakukan untuk mengetahui kondisi lapangan yang ada sehingga dapat diketahui implementasi perangkat lunak yang akan digunakan. Metode analisis yang digunakan adalah *Object Oriented Analysis. Use case diagram* digunakan untuk mendeskripsikan kebutuhan-kebutuhan dan fungsionalitas sistem dari perspektif *end-user*. Analisis kebutuhan dilakukan dengan mengidentifikasi semua kebutuhan (*requirements*) perangkat lunak yang kemudian akan dimodelkan dalam *use case diagram*.

Tahap analisis kebutuhan terdiri atas empat langkah yaitu melakukan penjabaran tentang gambaran umum aplikasi, melakukan proses identifikasi aktor yang terlibat dalam aplikasi, membuat daftar kebutuhan pengguna menggunakan tabel *Software Requirement Spesification*. Proses analisis kebutuhan ini diawali dengan penjabaran gambaran umum aplikasi *mobile* sistem penerimaan lowongan kerja, identifikasi aplikasi *mobile* sistem penerimaan lowongan kerja dan kemudian memodelkannya ke dalam diagram *use case.* Analisis kebutuhan ini bertujuan untuk menggambarkan kebutuhan-kebutuhan yang harus disediakan oleh sistem agar dapat memenuhi kebutuhan *user*.

3.3 Perancangan Sistem

Perancangan perangkat lunak dilakukan setelah semua kebutuhan perangkat lunak didapatkan melalui tahap analisis kebutuhan. Perancangan perangkat lunak berdasarkan *object oriented analysis* dan *object-oriented design* yaitu menggunakan pemodelan UML (*Unified Modeling Language*). Perancangan dimulai dari perancangan alur atau aktifitas yang dilakukan *user* secara prosedural yang dimodelkan dalam *activity diagram*. Selanjutnya, dilakukan perancangan interaksi antar elemen (objek) dalam sistem yang telah teridentifikasi dan dimodelkan dalam *sequence diagram* yang menggambarkan interaksi antar objek yang disusun dalam urutan waktu. Perancangan kelas-kelas dan *interface-interface* dalam sistem dimodelkan dalam class diagram. Diagram kelas digunakan untuk mengidentifikasi kelas-kelas serta paket-paket yang terdapat dalam sistem. Setelah paket-paket dalam sistem didapatkan, kemudian dilakukan perancangan basis data yang dimodelkan dengan ERD. Tahap yang terakhir adalah merancang antarmuka pengguna.

3.4 Implementasi

Implementasi perangkat lunak mengacu kepada perancangan perangkat lunak yang dilakukan pada tahap sebelumnya. Implementasi perangkat lunak diawali dengan penjabaran spesifikasi lingkungan perancangan perangkat lunak. Selanjutnya dijabarkan *mapping class* dengan *layout* saat implementasi perangkat lunak. Implementasi perangkat lunak dilakukan dengan menggunakan bahasa pemrograman Java dan PHP. Implementasi antarmuka berdasarkan perancangan yang telah dilakukan. Pada tahap akhir dilakukan implementasi simulasi pada *hardware* secara langsung dari Eclipse menggunakan *Android Development Tools* (ADT).

3.5 Pengujian dan Analisis

Pengujian perangkat lunak pada skripsi ini dilakukan agar dapat menunjukkan bahwa perangkat lunak telah mampu bekerja sesuai dengan spesifikasi dari kebutuhan yang telah dianalisa sebelumnya. Strategi pengujian perangkat lunak yang akan digunakan yaitu pengujian unit (*unit testing*),

pengujian validasi (*validation testing*), strategi perangkat lunak yang akan diuji dengan cara *black box testing*.

Estimasi waktu yang diperlukan untuk menguji aplikasi ini yaitu dua bulan atau 8 minggu. Pada bulan pertama, dilakukan pengujian secara *validation* testing yang mengacu pada sekumpulan aktifitas yang berbeda untuk memastikan bahwa perangkat lunak telah mengimplementasikan fungsi yang sesuai dengan spesifikasi (*requirement*) dan dapat memenuhi harapan pengguna sistem. Bulan kedua dilakukan pengujian secara *usability* testing yang merupakan pengujian terhadap fitur yang tersedia apakah bisa digunakan dengan mudah oleh *user* terutama pada penamaan tombol, lalu penempatan tombol aplikasi yang mudah sehingga *user* bisa memahami serta mengingat dalam perangkat lunak.

3.6 Pengambilan Kesimpulan dan Saran

Pada tahap pengambilan kesimpulan dilakukan saat semua tahapan mulai dari perancangan perangkat lunak, implementasi perangkat lunak, kemudian apabila semua tes telah dilakukan maka perlu adanya bentuk evaluasi yaitu melakukan pengecekan ulang pada tiap metode testing yang digunakan apakah sudah sesuai dengan criteria pengujian sehingga dapat menghasilkan data yang valid. Analisis juga dilakukan untuk mengetahui hasil dari pengujian perangkat lunak sehingga dapat didapatkan kesimpulan dari pengembangan perangkat lunak yang telah dilakukan. Setelah kesimpulan didapatkan, maka tahap akhir terakhir adalah saran yang berguna untuk memperbaiki celah atau kesalahan yang terjadi dari penelitian yang telah dibuat sehingga menyempurnakan penulisan dan dapat berfungsi sebagai landasan dalam mengembangkan serta memperbaiki kelemahan perangkat lunak selanjutnya.

BAB IV
ANALISIS DAN PERANCANGAN

Pada bab ini akan membahas proses perancangan analisis kebutuhan dan proses perancangan perangkat lunak. Berikut adalah tahap–tahap perancangan analisis dan perancangan perangkat lunaknya:

4.1 Analisis Kebutuhan

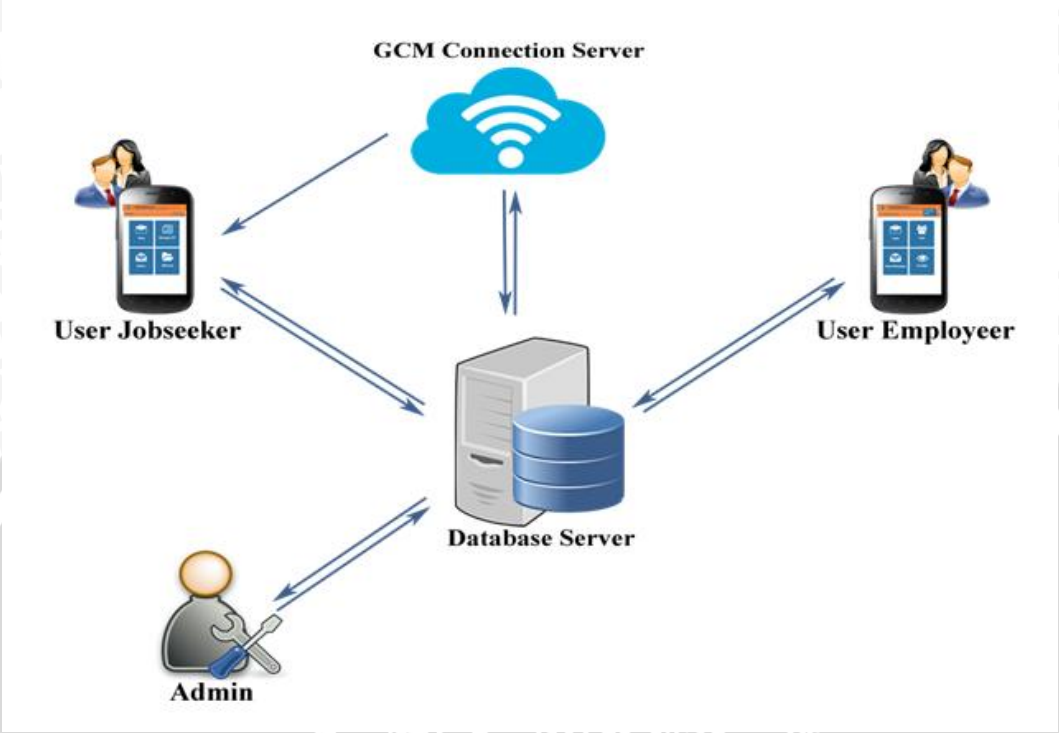
Proses analisis kebutuhan ini diawali dengan penjabaran gambaran umum aplikasi, penjabaran tentang daftar kebutuhan dan kemudian dimodelkan kedalam diagram *use case*. Analisis kebutuhan ini bertujuan untuk menggambarkan kebutuhan-kebutuhan yang harus disediakan oleh sistem agar dapat memenuhi kebutuhan pengguna.

4.1.1 Gambaran Umum Sistem

Perancangan umum sistem merupakan tahapan awal dari perancangan perangkat lunak. Perancangan sistem dilakukan untuk merepresentasikan arsitektur sistem yang akan dibuat secara umum. Sistem ini dibuat untuk mempertemukan sebuah perusahaan penyedia lowongan kerja dengan pencari kerja. Selain memberikan layanan untuk memasang iklan lowongan kerja ataupun mencari lowongan kerja, sistem ini juga memiliki layanan untuk melakukan pendaftaran pada suatu lowongan dengan mengirimkan CV yang sebelumnya telah dibuat.

Sistem ini memiliki 2 kategori *user* yakni user sebagai pencari kerja yang dikategorikan sebagai *jobseeker* dan *user* sebagai penyedia lowongan kerja yang akan dikategorikan sebagai *employeer*. Kategori *user* ditentukan melalui verifikasi id pada saat *login*, tiap-tiap kategori memiliki fitur yang berbeda yang mendukung kebutuhan kategori tersebut. Data info lowongan kerja, detail info lowongan kerja, profil penyedia lowongan kerja, profil pencari kerja dan CV pencari kerja disimpan pada *web server*. Data dapat diakses dan ditampilkan sesuai fungsi yang dipanggil/dijalankan oleh *user*. GCM digunakan untuk push *notification* pada sistem, *push notification* ditujukan pada *user* kategori *jobseeker* untuk info iklan baru dan pesan masuk dari *user* kategori *employeer*. GCM juga digunakan untuk *push notification* pada saat ada pembaruan sistem.

Tampilan/*User Interface* didesain sederhana sehingga mudah untuk digunakan oleh masyarakat umum khususnya para pencari kerja. Gambar 4.1 berikut menunjukan perancangan umum sistem.



Gambar 4.1 Diagram Arsitektur Sistem

4.1.2 Proses Bisnis Sistem Penerimaan Lowongan Kerja

Proses kerja aplikasi dimulai dari *user employeer* membuat dan meng-*upload* iklan lowongan kerja pada jendela menu *create job* yang terdapat di aplikasi ini. Kemudian sistem menampilkan iklan lowongan kerja yang telah dibuat *user employeer* pada jendela menu *search job* milik *user jobseeker*. Ketika *user jobseeker* memilih iklan, sistem kemudian menampilkan detail dari iklan lowongan kerja yang dipilih, mulai dari jabatan, gaji, persyaratan dan batas akhir pendaftaran. Kemudian ketika *user jobseeker* memilih tombol daftar *user jobseeker* akan diarahkan pada halaman CV yang telah dibuat sebelumnya oleh *user jobseeker*, kemudian *user jobseeker* dapat mengirimkan CV untuk mengakhiri proses pendaftaran. CV *user jobseeker* yang telah dikirim dapat dilihat oleh *user employeer* pada bagian menu *list registrant* sesuai dengan iklan lowongan kerja yang dibuat, nama dan pendidikan terakhir *registrant* akan ditampilkan secara list dihalaman halaman *list registrant* ini. Untuk menindak

lanjuti calon *registrant* yang sesuai pilihan kriteria *user employeer*, *user employeer* dapat mengirimkan pesan sesuai CV yang dikehendaki untuk dikirim ke *user jobseeker*.

4.1.3 Identifikasi Aktor

Tahap ini adalah tahap untuk melakukan identifikasi terhadap aktor-aktor yang akan berinteraksi dengan sistem atau aplikasi yang dibuat. Penjelasan dari masing-masing aktor dapat dilihat pada Tabel 4.1.

Tabel 4.1 Identifikasi Aktor

Aktor	Deskripsi
User	*User* merupakan seorang pengguna awal yang baru menjalankan aplikasi ini untuk membuat akun (*SignUp*) dan kemudian *Login*.
Job Seeker	*Job Seeker* merupakan seorang *user*yang ketika *SignUp* ia mendaftar sebagai seorang pencari kerja. *Jobseeker* dapat melihat berbagai lowongan yang tersedia, dapat mendaftar pada beberapa lowongan kerja yang tersedia.
Employeer	*Employeer* merupakan seorang *user*yang ketika *SignUp* ia mendaftar sebagai seorang pemberi kerja. *Employeer* dapat melihat lowongan yang tersedia, membuat lowongan kerja baru, serta dapat melihat para pelamar kerja yang mendaftar pada lowongan kerja yang dibuat *employeer* tersebut.
Admin	Berperan sebagai pemantau perkembangan Sistem, Admin dapat menghapus sebuah iklan maupun menghapus akun pengguna.

4.1.4 Analisis Kebutuhan Fungsional

Daftar kebutuhan terdiri dari kebutuhan fungsional. Pada daftar kebutuhan fungsional akan dispesifikasikan yaitu spesifikasi kebutuhan fungsional pengguna yang di tunjukkan pada Tabel 4.2 dengan pernomer mengunakan SRS (*Software Requirement Spesification*).

Tabel 4.2 User Requirement

Identifikasi	Kebutuhan	*Use Case*
SRS_01	Aplikasi menampilkan form pendaftaran dan menyimpan data *user*	*Sign Up*
SRS_02	Aplikasi mampu mengidentifikasi / mengenali seorang *user* berdasarkan *username* yang dimasukkan	*Login*
SRS_03	Aplikasi mampu menampilkan berbagai lowongan kerja yang tersedia, yang berada dalam *database system*	*List Job Advert*
SRS_04	Aplikasi mampu menampilkan secara detail tentang lowongan kerja yang tersedia	*View Detail Job Advert*
SRS_05	Aplikasi mampu menampilkan daftar lowongan kerja yang tersedia berdasarkan *keyword* / kualifikasi yang diberikan	*Search Job Advert*
SRS_06	Aplikasi mampu menampilkan *form* tambah iklan lowongan kerja, menambah iklan lowongan kerja baru, serta menyimpan iklan lowongan kerja yang telah dibuat	*Create Job Adverts*
SRS_07	Aplikasi mampu melakukan *editing* /*update* iklan lowongan kerja yang telah dibuat sebelum iklan di-*launching*	*Update Job Advert*
SRS_08	Aplikasi mampu menampilkan *history* iklan lowongan kerja yang telah diiklankan oleh suatu penyedia lowongan kerja	*History*

SRS_09	Aplikasi mampu menampilkan detail dari iklan lowongan kerja lama yang pernah dilakukan pendaftaran	*Detail History View*
SRS_10	Aplikasi mampu menerima pesan dan menampilkan isi pesan (*Broadcast*)	*View Message*
SRS_11	Aplikasi mampu menampilkan formpengisian *curriculum vitae* dan menyimpan data *curriculum vitae user*	*Create CV*
SRS_12	Aplikasi mampu mengambil data *curriculum vitae* pada *database* dan menampilkan data *curriculum vitae* yang telah dibuat	View CV
SRS_13	Aplikasi mampu menampilkan *form update curriculum vitae* sesuai data *curriculum vitae* sebelumnya dan menyimpan data *curriculum vitae* pengguna setelah di edit	*Edit CV*
SRS_14	Aplikasi mampu menampilkan Profil diri ataupun profil perusahaan	Profil
SRS_15	Aplikasi mampu memanipulasi data profil diri *user* maupun profil perusahaan	Edit Profil
SRS_16	Aplikasi mampu menampilkan form pengaduan iklan lowongan maupun akun bermasalah	Pengaduan
SRS_17	Aplikasi mampu menampilkan *list* data lowongan pekerjaan yang telah dibuat buat sebelumnya	*List History Job Advert*

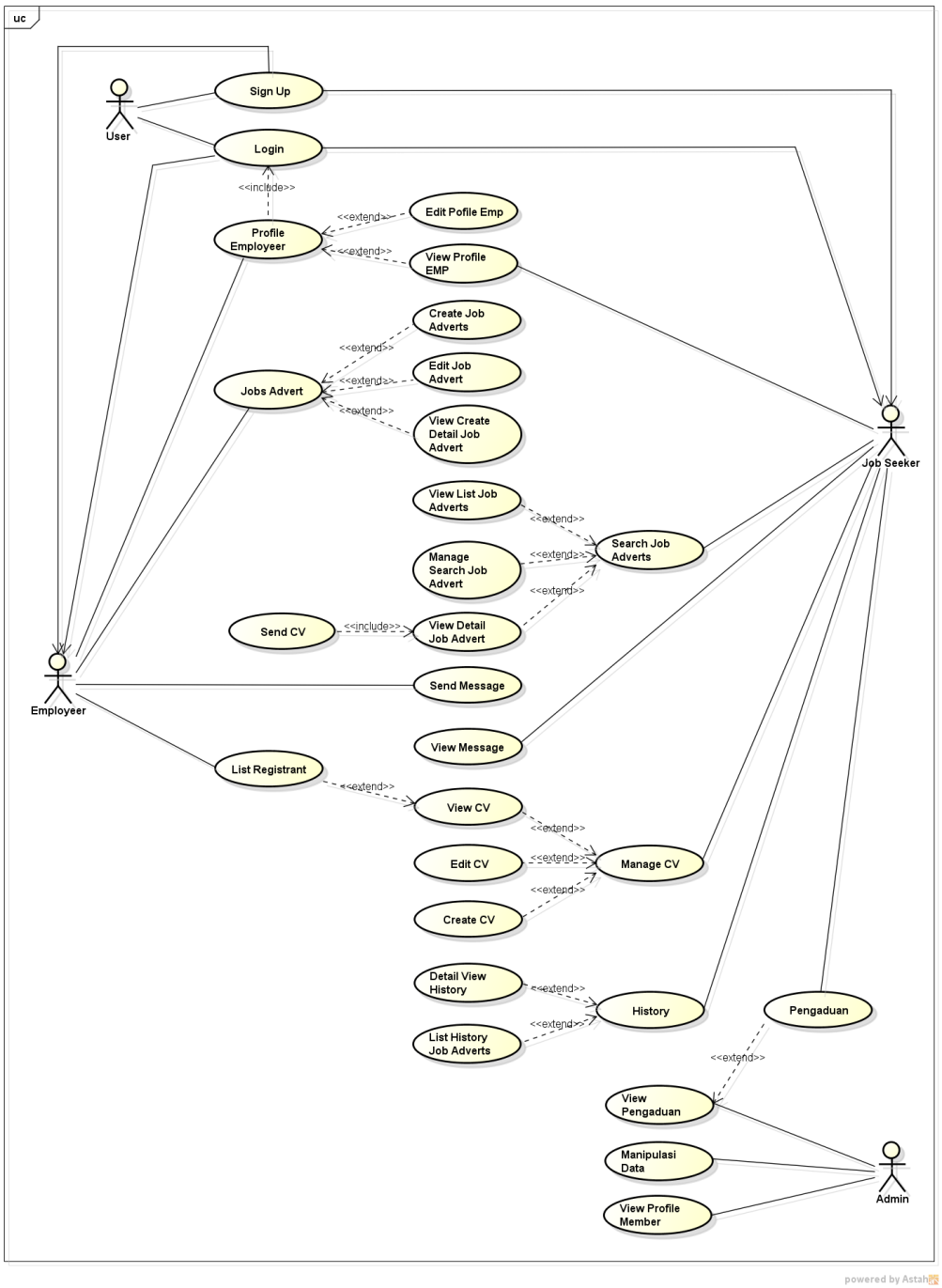
SRS_18	Aplikasi mampu menampilkan *list registrant* lowongan kerja pada suatu lowongan kerja yang telah diiklankan	*List Registrant*
SRS_19	Aplikasi mampu menampilkan data *curriculum vitae user* lain dengan ketentuan sebagai pemberi kerja dan pelamar kerja	*View CV Registrant*
SRS_20	Aplikasi mampu menampilkan *form* untuk membuat pesan dan mengirimkannya secara *broadcast*	*Send Message*
SRS_21	Aplikasi mampu menampilkan berbagai macam isi pengaduan yang ada	*View* Pengaduan
SRS_22	Aplikasi mampu menampilkan seluruh data *member*	*View Profile Member*
SRS_23	Aplikasi mampu memanipulasi data iklan lowongan maupun data pengguna aplikasi ini	Manipulasi Data

4.1.5 Diagram *Use Case*

Diagram *use case* adalah salah satu diagram yang digunakan untuk memodelkan aspek perilaku sistem. Diagram *use case* memuat sekumpulan nama *use case*, *actor* dan hubungannya. *Use case* merupakan fungsionalitas dari sistem yang diinisialisasi oleh aktor.

Pemodelan *use case* sistem diperoleh dari kebutuhan fungsional yang digunakan untuk menggambarkan interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Fungsi atau fitur yang akan dibuat pada aplikasi sistem penerimaan lowongan kerja berbasis *mobile* android antara lain daftar, *login*, *logout*, mengelola iklan lowongan kerja, mengelola CV, mengelola profil, mengirim pesan, menerima pesan, melakukan pendaftaran pada iklan lowongan

kerja, melihat daftar/*list registrant* pada suatu iklan lowongan kerja, dan beberapa fitur lainnya. Untuk lebih jelasnya dapat dilihat pada Gambar 4.2.



Gambar 4.2 Diagram *Use Case*

4.1.5.1 *Skenario Use Case*

Secara lebih mendetail, *use case create job advert*, *use case send CV*, dan *use case list registrant* yang terdapat pada diagram *use case* dijabarkan dalam

skenario *use case*. Di dalam skenario *use case*, akan diberikan uraian nama *use case*, aktor yang berhubungan dengan *use case* tersebut, tujuan dari *use case*, deskripsi global tentang *use case*, kondisi awal yang harus dipenuhi dan kondisi akhir yang diharapkan setelah berjalannya fungsional *use case*. Selain itu juga akan diberikan ulasan yang berkaitan dengan tanggapan dari sistem atas suatu aksi yang diberikan oleh aktor.

Skenario *use case* yang akan dijabarkan antara lain *use case create job advert*, *use case send* CV, *use case list registrant*. *Use case* tersebut dipilih karena *use case* tersebut mewakili penjelasan-penjelasan proses fitur utama pada aplikasi perangkat bergerak sistem penerimaan lowongan kerja pada *mobile* berbasis android ini.

Tabel 4.3 Skenario *Use Case Create Job Advert*

Nama	*Create Job Advert*
Tujuan	Untuk membuat iklan lowongan pekerjaan
Deskripsi	*Use case* ini menjelaskan *user* dengan kategori *employeer* dapat membuat iklan lowongan pekerjaan yang diinginkan
Aktor	*Employeer*
Kondisi Awal	Aplikasi telah ter-*install*, pengguna membuka aplikasi dan telah *login*
Alur Usecase	• Masuk kehalaman *home* pada aplikasi • Pengguna memilih menu *Jobs* • Pengguna memilih *icon create job* • Pengguna membuat iklan lowongan kerja baru
Kondisi Akhir	Aplikasi menampilkan *form* untuk membuat iklan lowongan kerja baru

Tabel 4.8 menjelaskan mengenai proses yang dilakukan pengguna dengan kategori *employeer* pada saat *Create Advert* pada Gambar 4.2 *Use case* **jobNgalam**

Tabel 4.4 Skenario Use Case Send CV

Nama	*Send* CV
Tujuan	Untuk mengirim CV
Deskripsi	*Use case* ini menjelaskan *user* dengan kategori *jobseeker* mengirimkan CV yang telah dibuat sebelumnya kepada *user* dengan kategori *employeer* berdasarkan iklan lowongan kerja yang dikehendaki user *jobseeker*
Aktor	*Jobseeker*
Kondisi Awal	Aplikasi telah ter-*install*, *user* membuka aplikasi dan telah *login*
Alur Usecase	• Masuk kehalaman *home* pada aplikasi • *User* memilih menu *Jobs* • *User* melihat-lihat iklan yang tersedia • *User* memilih salah satu iklan yang tersedia • *User* melihat detail isi/ketentuan iklan lowongan kerja • *User* memilih/melakukan pendaftaran pada iklan
Kondisi Akhir	Aplikasi menampilkan detail dari iklan lowongan kerja tertentu

Tabel 4.4 menjelaskan mengenai proses yang dilakukan *user* dengan kategori *jobseeker* pada saat *Send* CV pada Gambar 4.2 *Use Case* **jobNgalam**.

Tabel 4.5 Skenario Use Case List Registrant

Nama	*List Registrant*
Tujuan	Untuk melihat *list registrant*
Deskripsi	*Use case* ini menjelaskan *user* dengan kategori *Employeer* dapat melihat *list registrant* yang mendaftar pada iklan lowongan pekerjaan yang telah

	dibuat *employeer* tersebut
Aktor	*Employeer*
Kondisi Awal	Aplikasi telah ter-*install*, pengguna membuka aplikasi dan telah *login*
Alur Usecase	• Masuk kehalaman *home* pada aplikasi • *user me*milih menu *List* • *user* melihat-lihat *list* iklan yang pernah dibuat • *user*memilih salah satu iklan yang pernah dibuat • *user* melihat *list registrant* pada iklan tersebut
Kondisi Akhir	Aplikasi menampilkan *list history* lowongan pekerjaan

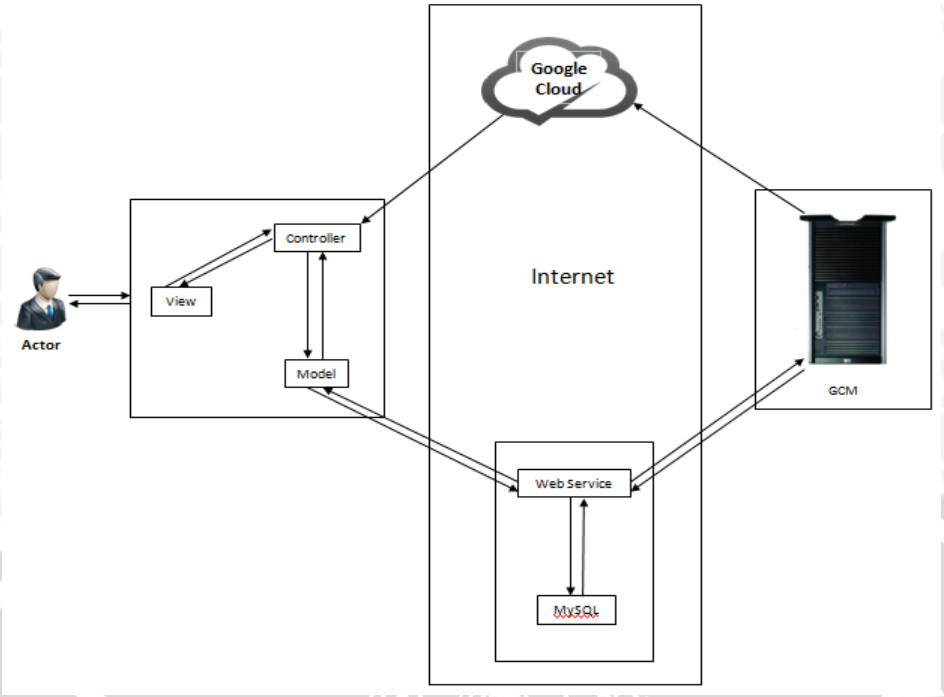
Tabel 4.5 menjelaskan proses yang dilakukan *user* dengan kategori *Employeer* pada saat *List Registrant* pada gambar 4.2 *Use case* **jobNgalam.**

4.2 Perancangan Perangkat lunak

4.2.1 Perancangan Arsitektur

MVC (*Model View Control*) merupakan metode yang digunakan untuk perancangan arsitektur pada sistem penerimaan lowongan kerja ini. *View* merupakan antarmuka yang digunakan untuk interaksi antara *user* dan sistem, sedangkan *controller* merupakan jembatan yang menghubungkan antara *view* dan *model*, kemudian *model* berisikan data yang dibutuhkan untuk mengakses sistem.

User akan berinteraksi dengan sistem menggunakan *view* (*layout*) pada perangkat bergerak, *controller* (*activity*) akan melakukan pengontrolan antara data yang diambil oleh model (JSON) dari server dengan *view*. Setiap data yang ter-*update* oleh server maka akan diambil oleh web service. Kemudian *web service* terkoneksi dengan *server* GCM (*Google Cloud Messaging*) dan menampilkan pemberitahuan kepada *user*. Perancangan arsitektur penerimaan lowongan kerja ini dapat dilihat dalam Gambar 4.6.



Gambar 4.6 Perancangan Arsitektur

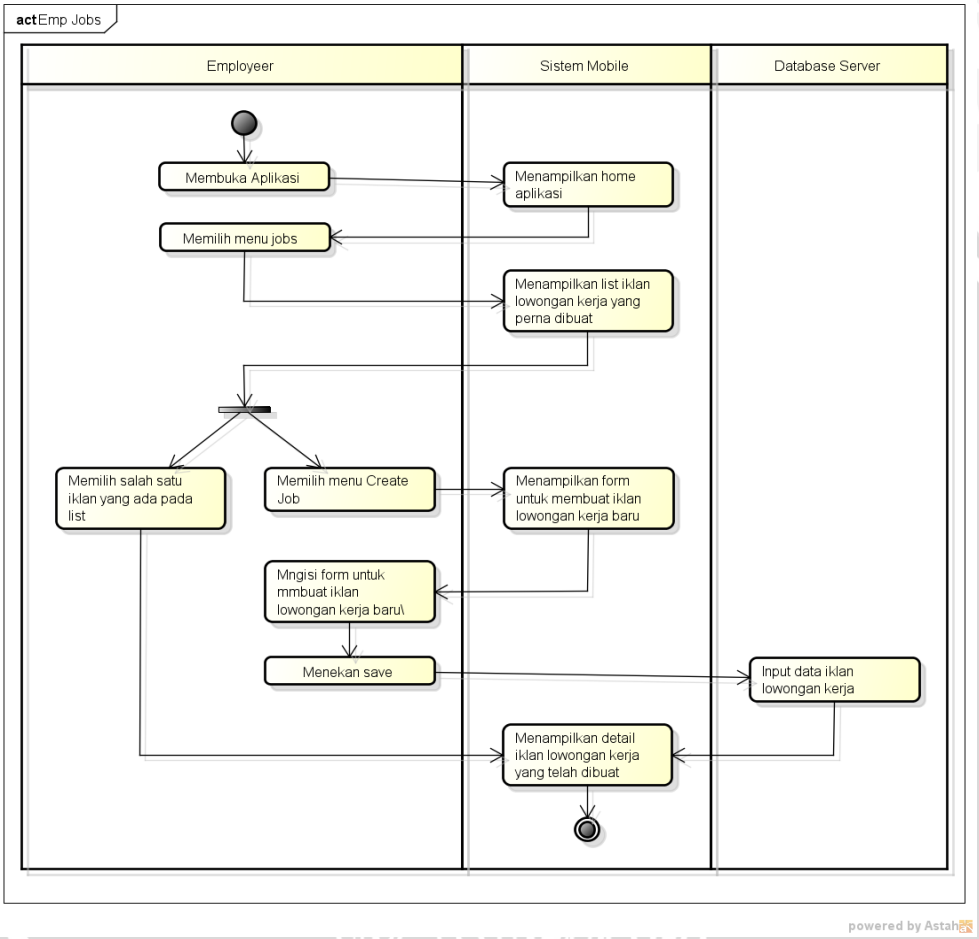
4.2.2 Diagram *Activity*

Diagram *activity* memodelkan aktivitas antara *user* dan sistem secara langkah demi langkah yang berjalan sesuai dengan deskripsi *use case* yang telah dibuat sebelumnya. Diagram *activity* disini akan dibedakan menjadi 3 bagian diantaranya diagram *activity user*, yakni diagram *activity* yang memodelkan aktivitas antara *user* dengan sistem dimana *user* belum memilki akun atau *user* belum melakukan *login*, diagram *activity employeer* yakni diagram *activity* yang memodelkan aktivitas antara *user* dengan sistem dimana *user* yang *login* dikenali sebagai *user* dengan kategori *employeer* dan diagram *activity jobseeker* yakni diagram *activity* yang memodelkan aktivitas antara *user* dengan sistem dimana *user* yang login dikenali sebagai *user* dengan kategori *jobseeker*. Berikut merupakan gambar dari masing-masing *activity diagram*.

4.2.2.1 *Activity Diagram Create Job Ads*

Pada gambar 4.3 di bawah ini menunjukkan *activity diagram Create Job Ads*, yang akan menjelaskan interaksi *user employeer* dengan sistem ketika *user* melakukan aktivitas membuat iklan lowongan kerja baru. Ketika *user employeer* memilih menu *jobs*, pada sistem menampilkan *list* iklan lowongan kerja yang

pernah dibuat jika sebelumnya sudah pernah membuat iklan lowongan, dan jika belum *user employeer* akan memilih menu *Create Job*. Setelah itu *user employeer* mengisi *form* untuk membuat iklan lowongan kerja baru yang nantinya data akan tersimpan pada *database server*, jika proses sudah dilakukan semua maka akan menampilkan detail iklan lowongan kerja yang telah dibuat.

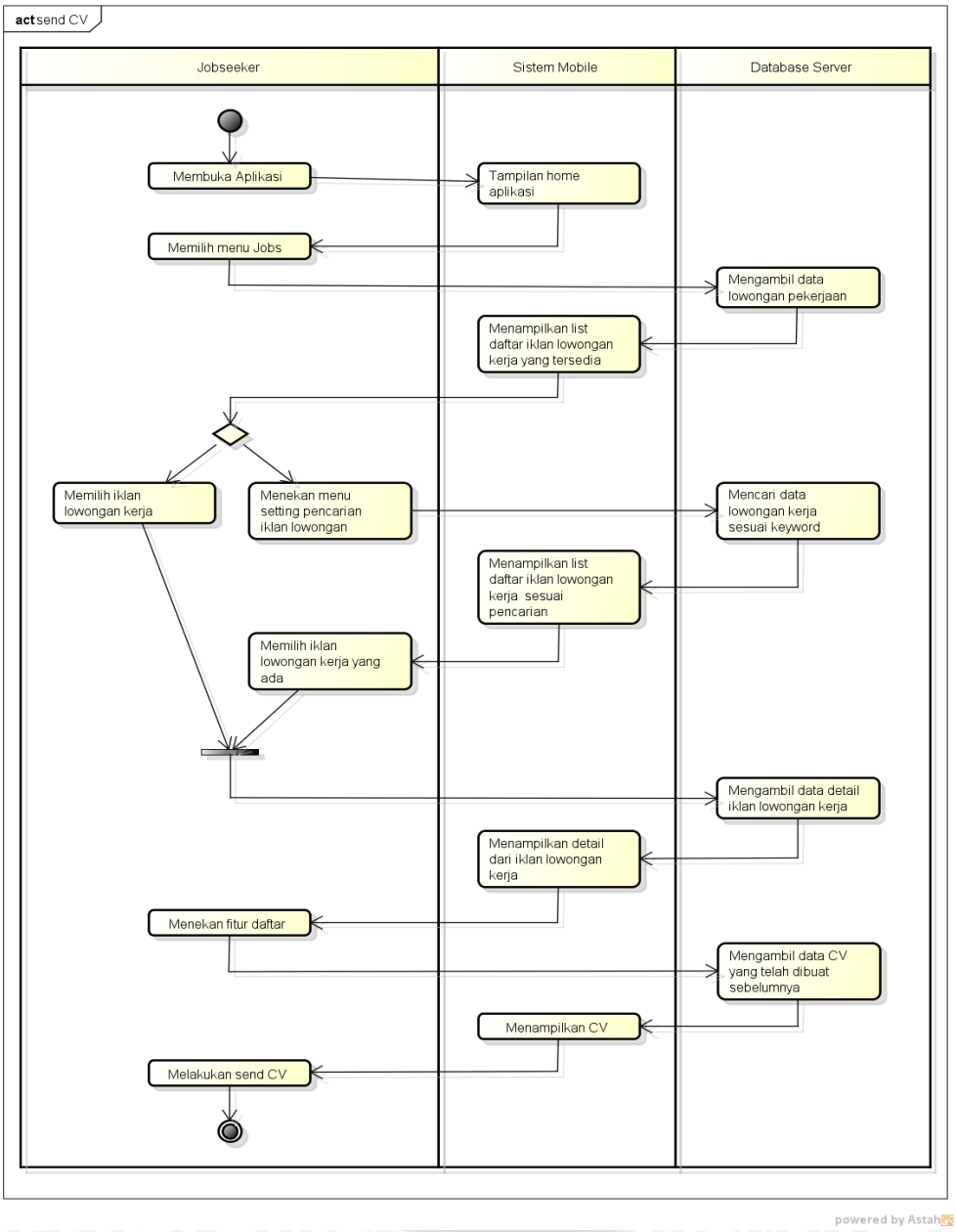


Gambar 4.3 Diagram Activity Create Job Ads

4.2.2.2 *Activity Diagram Send* CV

Pada gambar 4.5 di bawah ini menunjukkan *activity diagram Send CV*, yang akan menjelaskan interaksi yang terjadi antara *user jobseeker* dengan sistem ketika *user jobseeker* akan mengirimkan CV kepada *user employeer*. Ketika *user jobseeker* memilih menu *Jobs*, data lowongan pekerjaan akan diambil pada *database server* dan pada sitem akan menampilkan *list* daftar iklan lowongan kerja yang tersedia. Jika *user jobseeker* memilih iklan lowongan kerja maka *database server* akan mengambil data detail iklan lowongan kerja. Sedangkan, jika menekan menu *setting* pencarian iklan lowongan, di *database server* akan

melakukan pencarian data lowongan kerja sesuai *keyword.* Apabila pada sistem telah menampilkan detail iklan lowongan pekerjaan, *user jobseeker* akan menekan tombol daftar yang data CV yang telah dibuat sebelumnya akan diambil di *database server.* Setelah itu *user jobseeker* baru bisa melakukan pengiriman CV pada waktu sistem menampilkan CV.

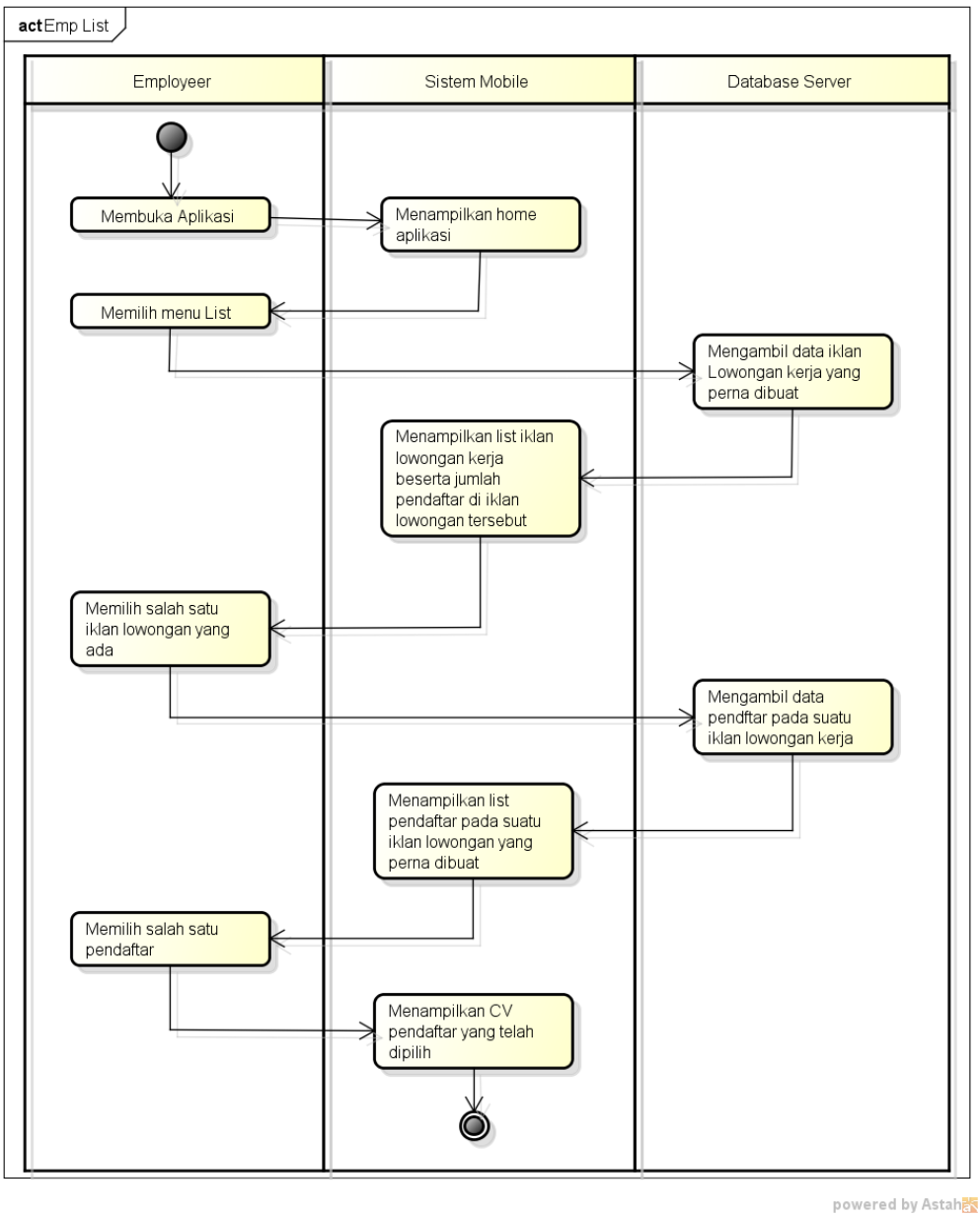


Gambar 4.4 Diagram *Activity Send* CV

4.2.2.3 *Activity Diagram List Registrant*

Pada gambar 4.5 di bawah ini menunjukkan *activity diagram List Registrant*, yangakan menjelaskan interaksi yang terjadi antara *user employeer*

dengan sistem ketika *user employeer* akan melihat *registrant* pada iklan lowongan kerja yang telah dibuat oleh *employeer*. Ketika *user employeer* memilih menu *list registrant*, pada *database server* mengambil data iklan lowongan kerja yang pernah dibuat, lalu pada sistem akan menampilkan *list* iklan lowongan kerja beserta jumlah *registrant* di iklan tersebut. Ketika *user employeer* memilih salah satu iklan, pada *database server* akan mengambil data *registrant* pada iklan tersebut, yang kemudian *user employeer* akan memilih salah satu *registrant* pada list *registrant* dan dapat melihat CV *registrant*.

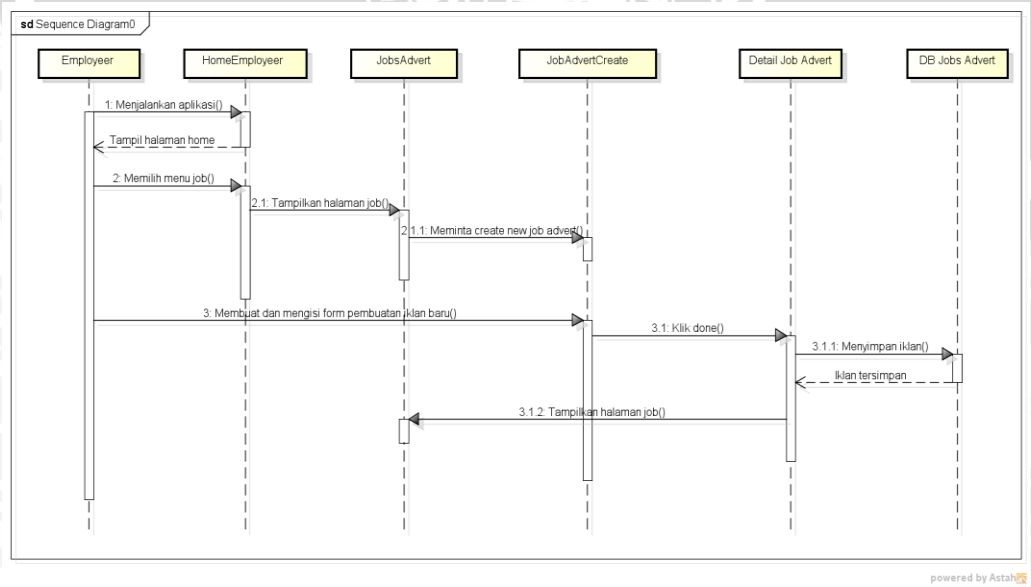


Gambar 4.5 Diagram Activity List Registrant

4.2.3 Perancangan Diagram Sequence

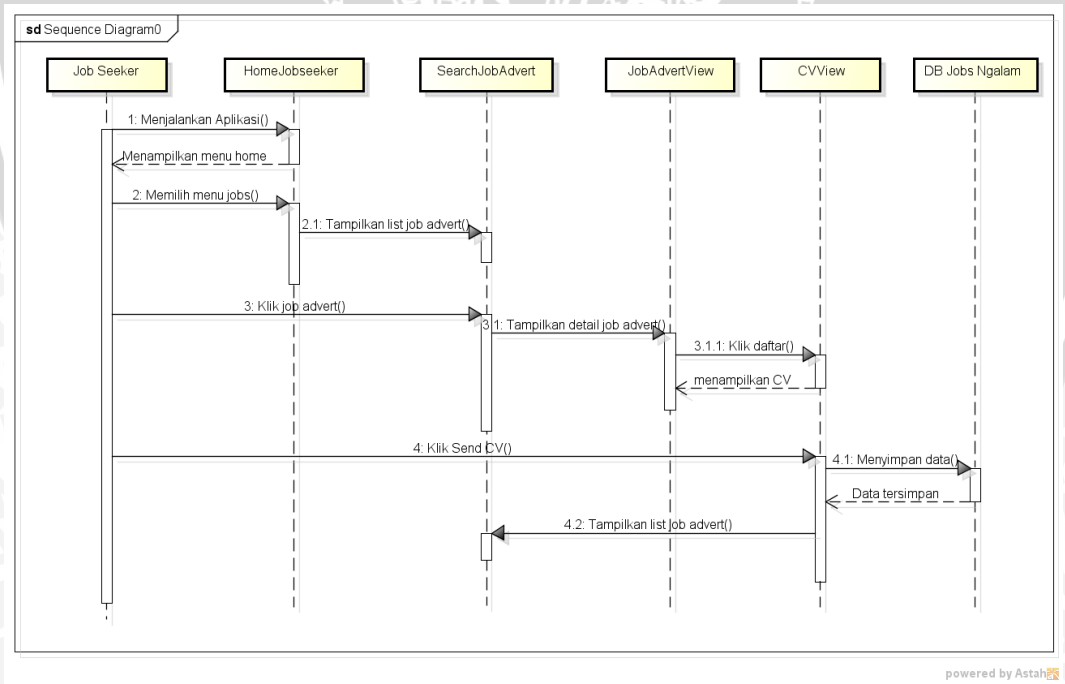
Diagram *sequence* digunakan untuk menggambarkan interaksi antar user dan sistem yang disusun dalam urutan waktu. Diagram *sequence* berupa deskripsi atau rangkaian langkah-langkah yang dilakukan dari sebuah kejadian untuk menghasilkan keluaran tertentu.

Pada Gambar 4.7 diagram sequence *create job advert* menjelaskan interaksi menampilkan daftar iklan lowongan kerja yang pernah dibuat, dimulai dari membuka aplikasi kemudian *user* (kategori *employeer*) melihat halaman *home*. Setelah tampilan *home* selesai dimuat oleh sistem, kemudian *user* (kategori *employeer*) memilih menu *jobs* untuk melihat daftar iklan yang pernah dibuat, kemudian pada halaman menu *jobs user* (kategori *employeer*) memilih fungsi *button create new* untuk membuat iklan lowongan kerja baru. Setelah sistem menampilkan *form* untuk membuat iklan lowongan kerja, *user* (kategori *employeer*) kemudian mengisi semua kolom pada *form* untuk membuat iklan lowongan kerja. Setelah *user* (kategori *employeer*) menekan tombol *save* sistem kemudian akan menampilkan detail iklan lowongan kerja yang telah dibuat sebagai konfirmasi sebelum iklan lowongan kerja tersebut dikirim ke server. Setelah *user* (kategori *employeer*) melakukan konfirmasi iklan yang baru dibuat, kemudian sistem akan mengirimkan iklan lowongan kerja ke server, kemudian sistem akan menampilkan halaman menu *jobs*.



Gambar 4.7 Diagram Sequence *Create Job Advert*

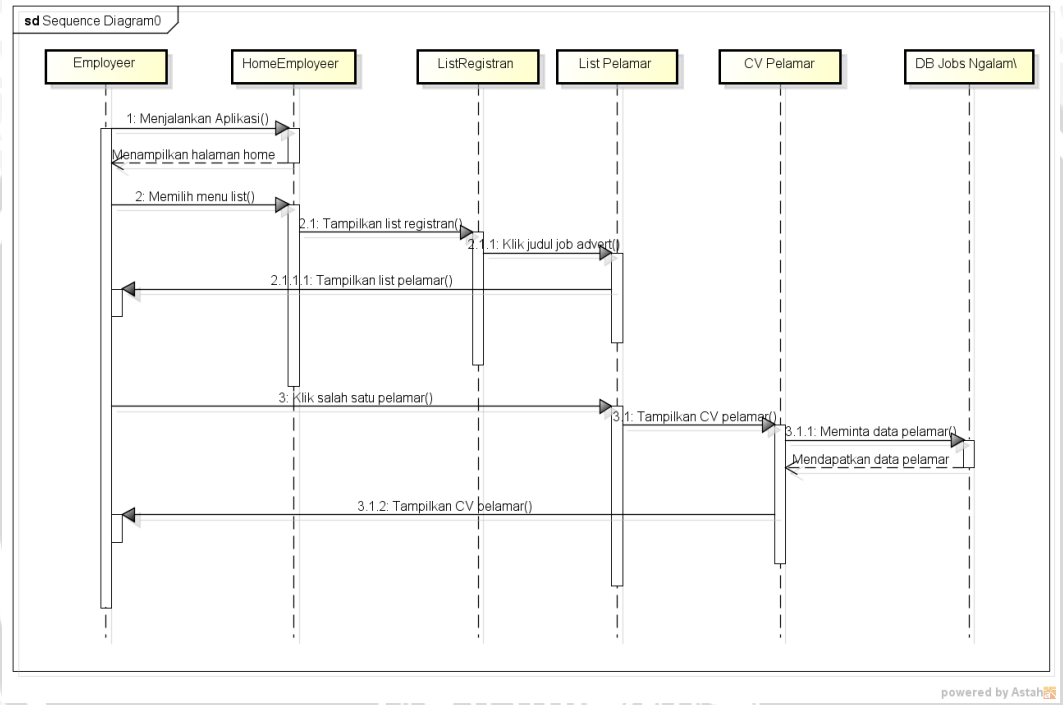
Pada Gambar 4.8 diagram *sequence send* CV menjelaskan proses pengiriman CV yang dilakukan pada saat akan mendaftar pada iklan lowongan kerja pada aplikasi ini. Dimulai dari membuka aplikasi kemudian *user* (kategori *jobseeker*) melihat halaman *home*. Setelah tampilan *home* selesai dimuat oleh sistem, kemudian *user* (kategori *jobseeker*) memilih menu *jobs* untuk melihat daftar iklan lowongan kerja yang ada pada server (secara *default* iklan ditampilkan secara list berdasarkan iklan lowongan kerja terbaru), setelah pada halaman menu *jobs user* (kategori *jobseeker*) memilih salah satu iklan yang diminati sistem akan menampilkan detail dari iklan yang dipilih. Setelah sistem menampilkan detail dari iklan lowongan kerja user (kategori *jobseeker*) memilih fungsi *button* daftar untuk melakukan pendaftaran pada iklan lowongan kerja yang dipilih. Setelah itu sistem akan menampilkan CV dari *user* (kategori *jobseeker*) untuk dikirimkan ke server sesuai pada iklan lowongan kerja yang dipilih. Setelah *user* (kategori *jobseeker*) menekan fungsi *button send*, sistem langsung mengirimkan CV *user* (kategori *jobseeker*) ke server. Kemudian setelah proses selasai sistem kembali menampilkan daftar iklan lowongan kerja yang ada pada server.



Gambar 4.8 Diagram *Sequence Send* CV

Pada Gambar 4.9 diagram sequence *list registrant* menjelaskan proses menampilkan *registrant* yang mendaftar pada suatu iklan lowongan kerja yang

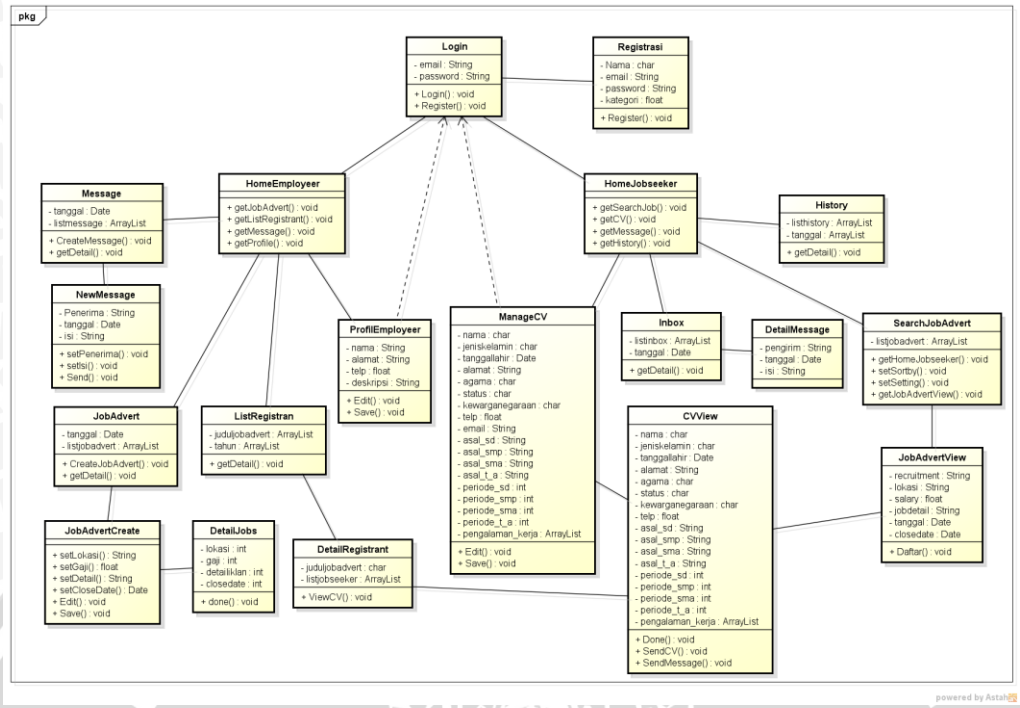
telah dibuat pada aplikasi ini. Dimulai dari membuka aplikasi kemudian *user* (kategori *employeer*) melihat halaman *home*. Setelah tampilan *home* selesai dimuat oleh sistem, kemudian *user* (kategori *employeer*) memilih menu *list* untuk melihat daftar iklan lowongan kerja telah dibuat sebelumnya, setelah *user* (kategori *employeer*) memilih salah satu iklan lowongan kerja yang pernah dibuat kemudian sistem menampilkan daftar pelamar pada iklan lowongan kerja tersebut. Kemudian ketika memilih salah satu nama pelamar yang ada sistem akan menampilkan CV pelamar sesuai data yang ada pada server.



Gambar 4.9 Diagram Sequence List Registrant

4.2.4 Perancangan Diagram *Class*

Diagram *class* digunakan untuk menggambarkan pemodelan elemen-elemen class yang membentuk sebuah sistem. Selain itu, diagram *class* juga menggambarkan relasi-relasi *class* pada sistem. Berikut ini merupakan rancangan diagram *class* untuk sistem yang akan dibuat.

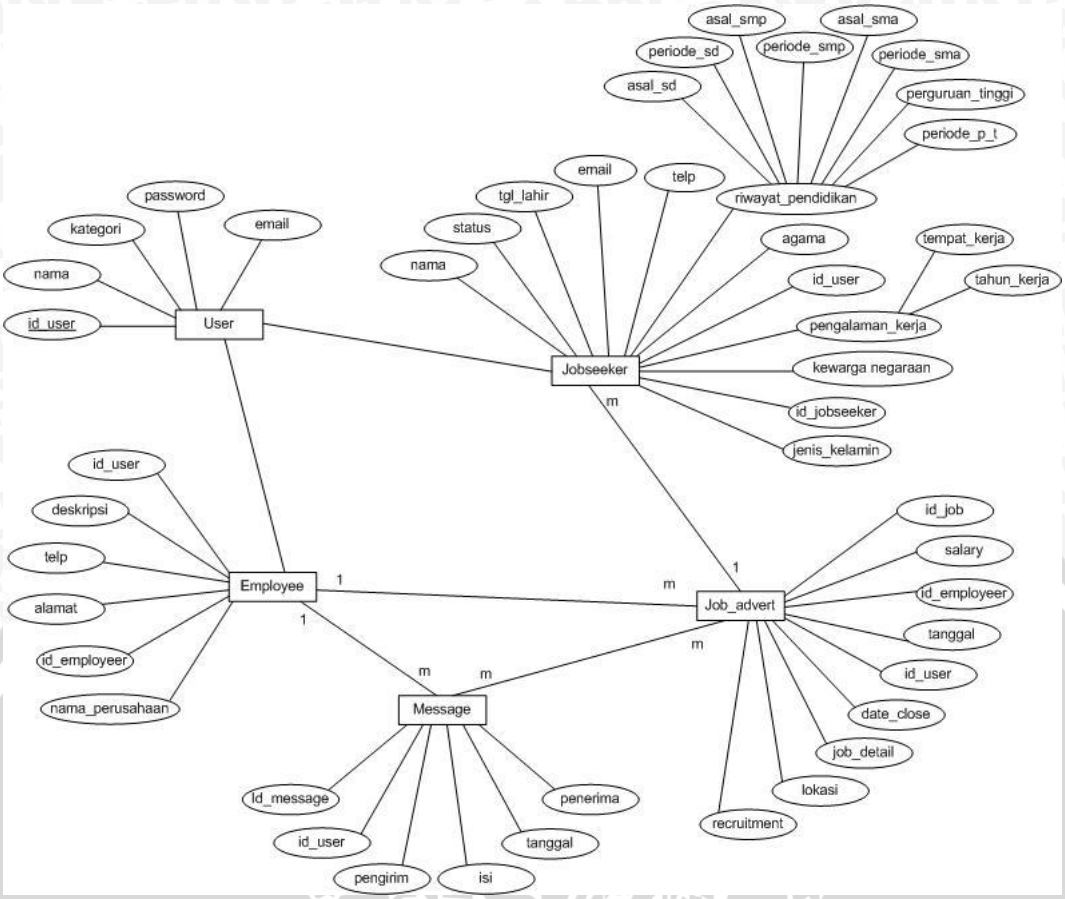


Gambar 4.10 Diagram *Class* Sistem

Dalam Gambar 4.10 diagram class sistem dijelaskan relasi-relasi antar class pada sistem yang akan dibuat. dimulai dari class *login* yang memiliki atribut *email, password* dan memiliki fungsi *login* dan registrasi. Class *login* berelasi dengan class registrasi diperlukan karena tidak semua *user* yang ingin menjalankan aplikasi telah memiliki akun pada aplikasi. *Class login* juga berelasi dengan *class home employeer* dan *home jobseeker* hal ini ketika *user* telah melakukan *login* akan diarahkan pada *class home employeer/home jobseeker*, tujuan class setelah *login* sesuai dengan id (*username*) yang digunakan untuk *login*. Pada class *home employeer* berelasi dengan *class list registrant, job advert, message* dan *profile employeer*, sedangkan pada *class home jobseeker* berelasi dengan class *manage CV, inbox, search job advert* dan *history*.

4.2.5 Perancangan Basis Data

Basis data diperlukan untuk menyimpan data yang dibutuhkan oleh sistem. Perancangan basis data digambarkan dalam bentuk ERD (Entity Relationship Diagram). Pada sistem ini, basis data terdiri dari 5 entity yang digunakan untuk menyimpan data *user, jobseeker, employeer, job advert, message*.



Gambar 4.11 Perancangan basis data sistem

Berdasarkan Gambar 4.11. dapat diketahui bahwa dalam ERD sistem terdapat 5 entitas, yaitu entitas *user*, *jobseeker*, *employeer*, *job advert*, *message*. Entitas *user* digunakan untuk menyimpan data *user*. Struktur entitas pengguna akan dijelaskan pada Tabel 4.6.

Tabel 4.6 Struktur entitas pengguna

Nama Kolom	Tipe Data	Keterangan	Contoh
id_user	Int	*Primary Key*	1,2,3
Nama	Varchar	-	Hutami
Kategori	Double	-	*Jobseeker/employeer*
Password	Varchar	-	asd123
e-mail	Varchar	-	hutami@ymail.com

Entitas *jobseeker* digunakan untuk menyimpan data *jobseeker*. Struktur entitas *jobseeker* akan dijelaskan pada Tabel 4.7.

Tabel 4.7 Struktur entitas *jobseeker*

Nama Kolom	Tipe Data	Keterangan	Contoh
Nama	Varchar	-	Hutami
Status	Varchar	-	Belum Kawin
tgl_lahir	Date	-	1991/11/03
e-mail	Varchar	-	hutami@ymail.com
Telp	Varchar	-	085334599***
Agama	Varchar	-	Islam
id_user	Int	*Foreign Key*	1,2,3
pengalaman_kerja	Varchar	-	Tomatech/*Mobile Programming*
tahun_kerja	Int	-	2014
asal_sd	Varchar	-	SDN 3 Pasuruan
periode_sd	Int	-	2004
asal_smp	Varchar	-	SMPN 2 Pasuruan
periode_smp	Int	-	2007
asal_sma	Varchar	-	SMAN 1 Pasuruan
periode_sma	Int	-	2010
Asal_p-t	Varchar	-	Univ. Brawijaya
Periode_t_a	Int	-	2014
Kewarganegaraan	Varchar	-	Indonesia
id_jobseeker	Int	*Primary Key*	1,2,3
jenis_kelamin	Varchar	-	Wanita

Entitas *employeer* digunakan untuk menyimpan data *employeer*. Struktur entitas *jobseeker* akan dijelaskan pada Tabel 4.8.

Tabel 4.8 Struktur entitas *employeer*

Nama Kolom	Tipe Data	Keterangan	Contoh
id_user	Int	*Foreign Key*	1,2,3
Deskripsi	Varchar	-	Perusahaan di bidang IT…
Telp	Varchar	-	085334599***
Alamat	Varchar	-	Jln Kahuripan 48 Malang
id_employeer	Int	*Primary Key*	1,2,3
nama_perusahaan	Varchar	-	CV. Skripsi Sukses

Entitas *job_advert* digunakan untuk menyimpan data iklan lowongan pekerjaan. Struktur entitas *job_advert* akan dijelaskan pada Tabel 4.9.

Tabel 4.9 Struktur entitas *job_advert*

Nama Kolom	Tipe Data	Keterangan	Contoh
id_job	Int	*Primary Key*	1,2,3
Salary	Varchar	-	2.500.000
id_employeer	Int	*Foreign Key*	1,2,3
Tanggal	Date	-	2015/05/10
id_user	Int	*Foreign Key*	1,2,3
date_close	Date	-	2015/06/10
job_detail	Varchar	-	Kualifikasi
Lokasi	Varchar	-	Malang
Recruitment	Varchar	-	*Programmer/Designer*

Entitas *message* digunakan untuk menyimpan data pesan. Struktur entitas *message* akan dijelaskan pada Tabel 4.10.

Tabel 4.10 Struktur entitas *message*

Nama Kolom	Tipe Data	Keterangan	Contoh
id_message	Int	*Primary Key*	1,2,3
id_user	Int	*Foreign Key*	1,2,3
Pengirim	Varchar	-	CV. Skripsi Sukses
Isi	Varchar	-	Selamat Anda Diterima
Tanggal	date	-	2015/06/11

Penerima	varchar	-	Hutami

4.2.6 Perancangan *Wireframe* Antarmuka

Perancangan antarmuka merupakan kerangka pembuatan sebuah aplikasi. Tujuan dari perancangan antarmuka untuk membuat sebuah tampilan dari aplikasi yang sederhana agar memudahkan *user* dalam mengoperasikan aplikasi. Antarmuka halaman daftar barang merupakan antarmuka untuk menampilkan sistem penerimaan lowongan kerja secara keseluruhan. Pada antarmuka ini terdapat tampilan foto masing-masing logo dan nama menu.

1. Halaman *Jobs Employeer*

Halaman *Jobs* akan ditampilkan *ketika user* (kategori *employeer*) memilih menu *Jobs* pada halaman home. Halaman *jobs* ini menampilkan list iklan lowongan kerja yang pernah dibuat serta juga untuk ketika ingin membuat iklan lowongan kerja baru.

Gambar 4.11 Antarmuka Halaman Jobs *Employeer*

Keterangan :

1. *Field ActionBar* : berisi logo dan nama aplikasi serta *ActionBar* untuk logout.
2. *Header* aplikasi : berisi nama dan kategori pengguna *login* serta tanggal saat ini.
3. Menu *button create job advert*: berfungsi untuk membuat iklan lowongan kerja baru.
4. List iklan lowongan kerja : berisi daftar iklan-iklan lowongan kerja yang pernah dibuat.
5. Menu *button back to home* : berfungsi untuk kembali ke halaman beranda.

2. Halaman *Create Job Advert*

Halaman *create job advert* akan ditampilkan ketika *user* (kategori *employeer*) memilih menu *create job* pada halaman *job employeer*. Halaman *create job* ini berisi *form* untuk membuat iklan lowongan kerja baru.

Gambar 4.12 Antarmuka Halaman *Create Job Advert*

Keterangan :

1. *Field ActionBar* : berisi logo dan nama aplikasi serta *ActionBar* untuk logout.
2. *Header* aplikasi : berisi nama dan kategori pengguna *login* serta tanggal saat ini.
3. *Form* isian iklan lowongan kerja : berisi *field-field* yang diperlukan untuk membuat iklan lowongan kerja baru.
4. Menu *button save* : untuk menyimpan iklan lowongan kerja yang baru dibuat.

3. Halaman *View Job*

Halaman *view job* akan ditampilkan ketika *user* (kategori *employeer*) selesai membuat iklan lowongan kerja (memilih *save* pada halaman *create job*). Halaman *view job* ini digunakan untuk melihat atau memvalidasi iklan lowongan kerja yang telah dibuat sebelum di iklankan ke pengguna *jobseeke*r.

Gambar 4.13 Antarmuka Halaman *View Job*

Keterangan :

1. *Field Action Bar* : berisi logo dan nama aplikasi serta ActionBar untuk *logout*.
2. *Header* aplikasi : berisi nama dan kategori pengguna *login* serta tanggal saat ini.
3. *Field* data iklan lowongan kerja : berisi data iklan lowongan kerja yang telah dibuat sebelumnya sebelum dikirim ke server.
4. Menu *button Edit* iklan : untuk melakukan edit data iklan yang telah dibuat sebelumnya sebelum dikirim ke server.
5. Menu *button Done* : untuk mengakhiri pembuatan iklan lowongan kerja baru dan mengirimkan iklan lowongan kerja ke server.

4. Halaman *Search Jobs Advert*

Halaman *Search Jobs Advert* akan ditampilkan ketika *user* (kategori *jobseeker*) memilih menu *Jobs* pada halaman *home*. Halaman *Search Jobs Advert* ini menampilkan iklan-iklan lowongan kerja yang ada pada *database* dan ditampilkan secara *list* dari iklan terbaru.

Gambar 4.14 Antarmuka Halaman *Jobs Jobseeker*

Keterangan :

1. *Field Action Bar* : berisi logo dan nama aplikasi serta *ActionBar* untuk *logout*.
2. *Header* aplikasi : berisi nama dan kategori pengguna *login* serta tanggal saat ini.
3. Menu *button back to home* : untuk kembali ke beranda
4. *Field function sort by* iklan : untuk mengatur urutan iklan yang ingin ditampilkan.
5. Menu *button setting*
6. *Field setting search advanced*
7. *List* iklan lowongan kerja : daftar iklan lowongan kerja
8. Menu button search: untuk memulai pencarian iklan lowongan kerja sesuai ketentuan yang telah diberikan.

5. Halaman *Jobs Detail*

Halaman *Jobs Detail* akan ditampilkan ketika *user* (kategori *jobseeker*) memilih salah satu iklan lowongan kerja yang ada pada pada halaman *Search Jobs Advert*. Halaman *Jobs Detail* ini menampilkan detail dari iklan lowongan kerja yang dipilih.

Gambar 4.15 Antarmuka Halaman *Search Jobs Advert*

Keterangan:

1. *Field Action Bar* : berisi logo dan nama aplikasi serta *ActionBar* untuk logout.
2. *Header* aplikasi : berisi nama dan kategori pengguna *login* serta tanggal saat ini.
3. Detail iklan lowongan kerja : berisi info iklan lowongan kerja secara lebih terperinci.
4. Menu *button* daftar : digunakan untuk melakukan pendaftaran pada lowongan kerja.

6. Halaman *List Registrant*

Halaman *List Registrant* akan ditampilkan ketika *user* (kategori *employeer*) memilih menu *List* pada halaman *home*. Halaman *list* ini menampilkan *list* iklan yang pernah dibuat yang didasarkan pada tahun pembuatannya.

Gambar 4.16 Antarmuka Halaman *List*

Keterangan :

1. *Field Action Bar* : berisi logo dan nama aplikasi serta *ActionBar* untuk *logout*.
2. *Header* aplikasi : berisi nama dan kategori pengguna *login* serta tanggal saat ini.
3. List tahun pembuatan iklan : berisi daftar tahun-tahun ketika pengguna membuat iklan lowongan kerja pada aplikasi ini.
4. List iklan lowongan kerja : berisi daftar iklan lowongan kerja yang pernah dibuat berdasarkan tahun yang tertera.

7. Halaman *List View*

Halaman *list view* akan ditampilkan ketika pengguna (*user employeer*) memilih salah satu iklan lowongan yang pernah dibuat pada halaman *list*. Halaman *list view* ini menampilkan *list registrant* pada suatu iklan yang pernah dibuat.

Gambar 4.17 Antarmuka Halaman *List View*

Keterangan :

1. *Field Action Bar* : berisi logo dan nama aplikasi serta *ActionBar* untuk *logout*.
2. *Header* aplikasi : berisi nama dan kategori pengguna *login* serta tanggal saat ini.
3. Judul iklan lowongan kerja : berisi judul iklan lowongan kerja berdasarkan iklan lowongan kerja yang dipilih.
4. *List registrant* : berisi *registrant* iklan lowongan kerja terkait yang ditampilkan dalam *list*.

8. Halaman *View* CV

Halaman *view* CV akan ditampilkan ketika pengguna (*user employeeer*) memilih salah satu nama pada *list registrant* pada halaman *List View*.

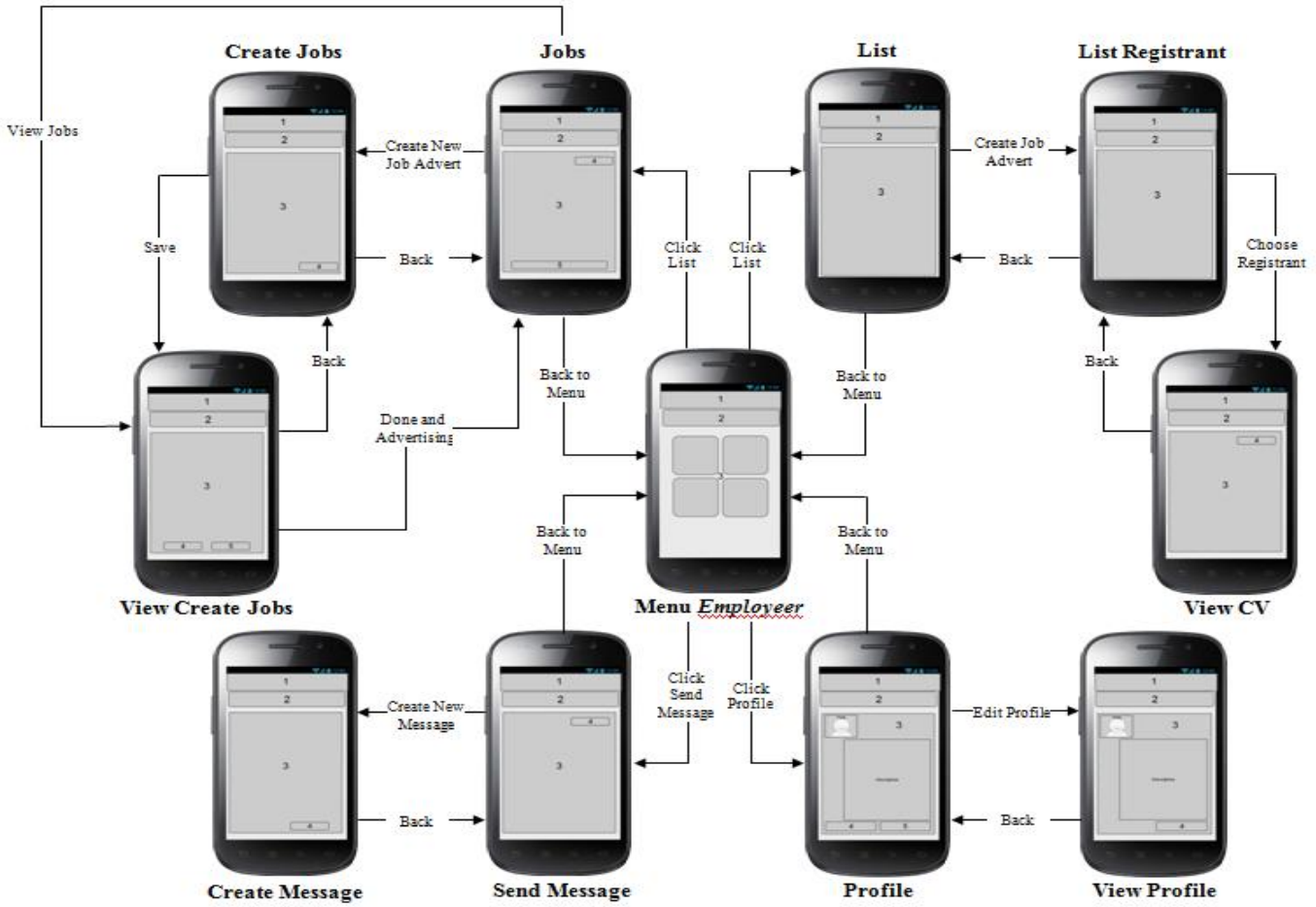
Gambar 4.18 Antarmuka Halaman *View* CV

Keterangan :

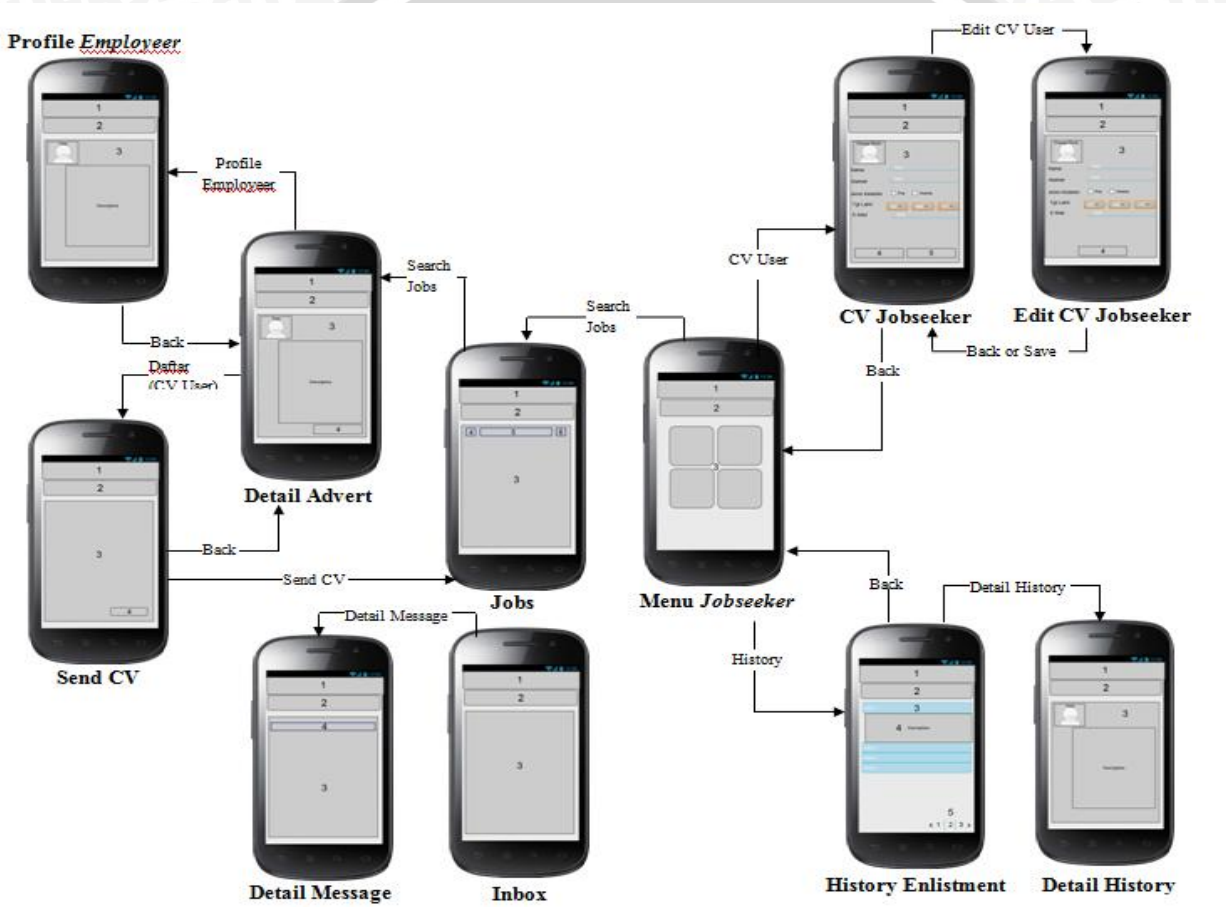
1. *Field Action Bar* : berisi logo dan nama aplikasi serta *ActionBar* untuk logout.
2. *Header* aplikasi : berisi nama dan kategori pengguna *login* serta tanggal saat ini.
3. *Field* CV *registrant* : berisi data *curriculum vitae registrant*.
4. Menu *button like* : untuk menandai *registrant* agar bisa dilihat lagi untuk melakukan balasan dari pendaftaran atas iklan lowongan kerja yang dibuat.

4.2.7 *Process Flow*

Process Flow adalah gambaran proses sederhana dari aplikasi yang akan dibuat. Hal ini dimaksudkan untuk memberikan gambaran tentang halaman – halaman aplikasi yang akan dibuat dan *link-link* yang akan diberikan dalam aplikasi tersebut. Hal tersebut dapat memberikan batasan dalam mengembangkan aplikasi dan mempermudah dalam pembuatan desain aplikasi. Rancangan *process flow* tersebut dapat dilihat pada Gambar 4.20.



Gambar 4.20 ***Process Flow Employeer***



Gambar 4.21 ***Process Flow Jobseeker***

BAB V
IMPLEMENTASI DAN PENGUJIAN

Bab ini membahas mengenai tahapan implementasi dan pengujian aplikasi sistem penerimaan lowongan kerja berdasarkan dari hasil analisis kebutuhan dan perancangan aplikasi. Pembahasan terdiri dari implementasi basis data, implementasi *class*, dan implementasi antarmuka. Kemudian dilanjutkan dengan pengujian validasi pada aplikasi tersebut. Setelah pengujian selesai, akan dilanjutkan dengan proses analisis untuk mendapatkan kesimpulan dari hasil pengujian aplikasi sistem penerimaan lowongan kerja.

5.1 Implementasi

Bagian ini diawali dengan penjelasan tentang spesifikasi lingkungan pengembangan perangkat lunak, dilanjutkan dengan implementasi basis data, implementasi *class diagram*, implementasi *code* program dan antarmuka aplikasi.

5.1.1 Spesifikasi Lingkungan Pengembangan Sistem

Pengembangan sistem penerimaan lowongan kerja *mobile* berbasis android ini menggunakan sebuah komputer *notebook*/laptop dengan *processor* Intel Core i3 dan operasi sistem windows 7 perangkat lunak yang digunakan untuk membuat aplikasi penerimaan lowongan kerja yaitu Eclipse June, ADT (*Android Develeper Tool*) dengan bahasa pemrogram Java untuk aplikasi Android, dan Subline Text 2 dengan bahasa pemrograman PHP untuk *web service*. Uji coba aplikasi penerimaan lowongan kerja *mobile* berbasis android ini mengunakan emulator android dengan OS android 4.2.2 (Jelly Bean), *device mobile* android dengan OS android 2.3.6 (Gingerbread) dan android 4.1.2 (Jelly Bean).

5.1.2 Batasan-Batasan Implementasi

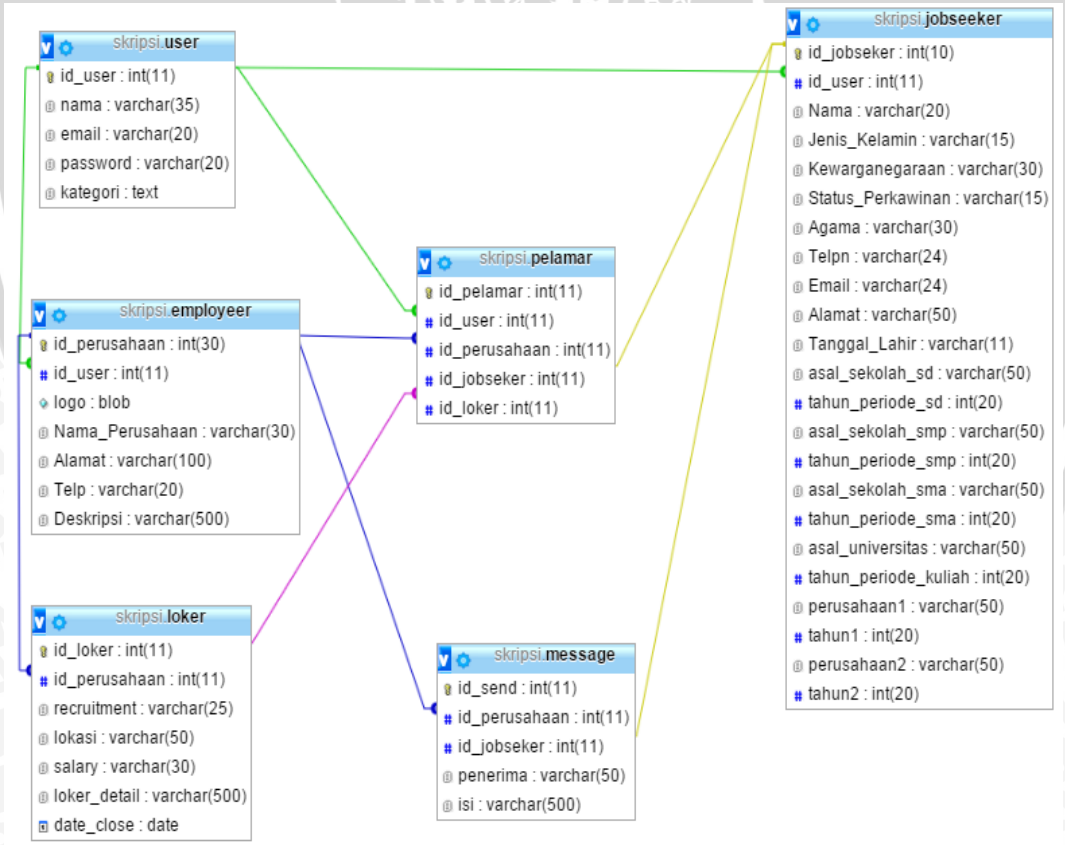
Pada pembuatan sistem penerimaan lowongan kerja *mobile* berbasis android ini terdapat batasan-batasan dalam proses implementasinya yaitu sebagai berikut:

1. Pembuatan sistem dan antarmuka sistem menggunakan bahasa pemrograman .java dan xml.
2. Penyimpanan data yang digunakan adalah MySQL.

3. Aplikasi penerimaan lowongan kerja *mobile* berbasis android dirancang untuk dijalankan pada *smartphone* Android.
4. Aplikasi hanya dapat dijalankan dengan menggunakan koneksi internet.
5. Pengambilan/pembagian data antara aplikasi dengan basis data yang terdapat pada server menggunakan JSON.

5.1.3 BasisData

Implementasi penyimpanan data dilakukan dengan *database management* system MySQL. Hasil implementasi penyimpanan data ini berupa file basis data dengan format ekstensi file sql. Hasil implementasi MySQL pada basis data ini dimodelkan dalam *physical* diagram. Pada *physical* diagram terdapat hubungan relasi antar tabel. Pada sistem ini, data-data disimpan dalam 5 tabel yang berbeda sesuai kebutuhan, antara lain data *user, jobseeker, employeer,* loker, pelamar dan *message.*



Gambar 5.1 *Physical* Diagram

5.1.4 Implementasi *Class*

Setiap *class* yang telah dirancang pada perancangan Diagram *class* dalam Gambar 4.10 direalisasikan pada sebuah *file* program. Dalam Gambar 4.10 *class default* (sistem) direalisasikan dengan ekstensi .java dan *file layout* dengan ekstensi .xml sebagai tampilan dari *class* tersebut. Pada Tabel 5.1 menjelaskan mengenai pasangan antara *class file* program dan *layout* yang digunakan untuk mengimplementasikan tampilan aplikasi .

Tabel 5.1 Implementasi Perancangan Class Diagram

No	Package	Nama *Class*	Nama File Program	Nama Layout
1	com.tam.jobsngalam	*Login*	login.java	login.xml
2	com.tam.jobsngalam	*Register*	registrasi.java	registrasi.xml
3	com.tam.jobsngalam	*Home Employeer*	employeer.java	employeer.xml
4	com.tam.jobsngalam	*Home Jobseeker*	jobseeker.java	jobseeker.xml
5	com.tam.jobsngalam	*Profile Employeer*	employeer_profil.java	employeer_profil.xml
6	com.tam.jobsngalam	*Messsage*	employeer_message.java	employeer_message.xml
7	com.tam.jobsngalam	*List Registrant*	employeer_list.java	employeer_list.xml
8	com.tam.jobsngalam	*Create Job Advert*	employeer_insertjob.java	employeer_insertjob.xml
9	com.tam.jobsngalam	*Job Advert*	detail_job.java	detail_job.xml
10	com.tam.jobsngalam	*Manage CV*	jobseeker_cv.java	jobseeker_cv.xml
11	com.tam.jobsngalam	*Inbox*	jobseeker_inbox.java	jobseeker_inbox.xml

12	com.tam.jobsngalam	*History*	jobseeker_history.java	jobseeker_history.xml
13	com.tam.jobsngalam	*Search Job Advert*	jobseeker_job.java	jobseeker_job.xml
14	com.tam.jobsngalam	*View Job Advert*	joseeker_viewjob.java	joseeker_viewjob.xml

5.1.5 Implementasi *Code* Program

Pada penulisan implementasi *code* program ini hanya dicantumkan algoritma dari beberapa proses saja, sehingga tidak semua *code* program dicantumkan. Algoritma proses yang dicantumkan antara lain proses *login*, proses tambah iklan lowongan kerja, proses mencari iklan lowongan kerja, proses daftar iklan lowongan kerja, proses melihat *registrant* iklan lowongan kerja.

5.1.5.1 Implementasi Algoritma Proses *Login*

Proses *login* dilakukan dengan mengisi *username* dan *password* pada halaman awal aplikasi. Proses *login* dilakukan untuk menentukan halaman *home* yang akan ditampilkan, halaman *home* ditentukan berdasarkan kategori *user* yakni antara kategori *employeer* dan kategori *jobseeker*. Proses *login* ditunjukan pada Kode 5.1 dan Kode 5.2.

Kode 5.1 Algoritma Proses *Login* (java)

```
login.setOnClickListener(new View.OnClickListener() {

      public void onClick(View v) {
            url = "http://"+ip+"/login_jobs/login.php?" + "email="
                  + password.getText().toString() + "&password="
                  + email.getText().toString();

            if (email.getText().toString().trim().length() > 0
                  && password.getText().toString().trim().length()
                  > 0 ) {
                        new Masuk().execute();}
            else {
                  Toast.makeText(getApplicationContext(),
                  "Username dan Password masih kosong!!",
                  Toast.LENGTH_LONG).show(); }
                  }
      });
```

Kode 5.2 Algoritma Proses *Login* (php)

```
$email    = $_GET["email"];
$password = $_GET["password"];

$query = "select * from user where email='$email' and
 password='$password'";

$hasil = mysql_query($query);
if (mysql_num_rows($hasil) > 0) {
$response = array();
$response["login"] = array();
while ($data = mysql_fetch_array($hasil))
{
$h['id']            = $data['id_user'] ;
$h['nama']          = $data['nama'] ;
$h['email']           = $data['email'] ;
$h['password']      = $data['password'];
$h['kategori']      = $data['kategori'];

 array_push($response["login"], $h);
}     $response["success"] = "1";
 echo json_encode($response);
}
else {
    $response["success"] = "0";
    $response["message"] = "Tidak ada data";
 echo json_encode($response);
}
```

5.1.5.2 *Implementasi Algoritma* Proses Tambah Iklan Lowongan Kerja

Proses tambah iklan lowongan kerja dilakukan dengan menekan menu *job* pada halaman *home* (kategori *employeer*) kemudian menekan *icon create job*. Penambahan iklan lowongan kerja akan disimpan pada *database* server dan ditampilakn pada menu *job* sesuai tanggal pembuatannya.

Kode 5.3 Algoritma Tambah Iklan (java)

```
private void updateUi() {
     String[] from = {
"data_perusahaan","data_recruitment","data_salary","data_date" };
     int[] to = {
R.id.id_perusahaan,R.id.recruitment,R.id.salary,R.id.date_close};
     SimpleAdapter adapter = new
SimpleAdapter(getApplication().getBaseContext(), aList,
R.layout.list_item, from, to);
     lv.setAdapter(adapter);
     lv.setOnItemClickListener(new OnItemClickListener() {
public void onItemClick(AdapterView<?> arg0, View arg1, int
position,
      long id) {
            final String id_data = data_loker[position];

     Intent(getBaseContext(),employeer_inserjobPV.class);
                  intent.putExtra("id_loker", id_data);
```

```
                    startActivity(intent);
                }
            });
      protected Void doInBackground(Void... params) {
       WeServiceHandler webservice = new WeServiceHandler();
       String jsonstr = webservice.getJsonData(url+id);
      try {
            JSONObject jsonObject = new JSONObject(jsonstr);
            JSONArray jsonArray =
jsonObject.getJSONArray(lowongan);
          data_loker = new String[jsonArray.length()];
          data_perusahaan = new String[jsonArray.length()];
          data_recruitment = new String[jsonArray.length()];
          data_salary = new String[jsonArray.length()];
          data_date = new String[jsonArray.length()];
         for (int i = 0; i < jsonArray.length(); i++) {
          JSONObject testjson = jsonArray.getJSONObject(i);
          data_loker[i] = testjson.getString(Vid_loker);
          data_perusahaan[i] = testjson.getString(Vid_perusahaan);
          data_recruitment[i] = testjson.getString(Vrecruitment);
          data_salary[i] = testjson.getString(Vsalary);
          data_date[i] = testjson.getString(Vdate_close);
          HashMap<String, String> hm = new
HashMap<String,String>();
          hm.put("data_perusahaan", data_perusahaan[i]);
          hm.put("data_recruitment", data_recruitment[i]);
          hm.put("data_salary", data_salary[i] );
          hm.put("data_date", data_date[i] );
          aList.add(hm);
       }
          }catch (Exception e){
      e.printStackTrace();
      }
      return null;
}
```

Kode 5.4 Algoritma Proses Tambah Iklan (php)

```
          $recruitment     = $_POST['recruitment'];
          $lokasi          = $_POST['lokasi'];
          $salary          = $_POST['salary'];
          $loker_detail    = $_POST['loker_detail'];
          $date_close      = $_POST['date_close'];
       $tanggal         = $_POST['tanggal'];
          $id_perusahaan   = $_POST['id_perusahaan'];
          $id_user         = $_POST['id_user'];

        include "koneksi.php";

       $query2 = "INSERT INTO t_loker
      (recruitment,lokasi,salary,loker_detail,date_close,tanggal,i
d_perusahaan,id_user)VALUES('$recruitment','$lokasi','$salary','$l
oker_detail','".date('Y-m-d',
strtotime($date_close))."','".date('Y-m-d',
strtotime($tanggal))."','$id_perusahaan','$id_user')";
      $hasil2 = mysql_query($query2);
```

```
      if($hasil2)
       {
       $response["success"] = "1";
              $response["message"] = "Data sukses diinput";
              echo json_encode($response);
       }
       else
       {$response["success"] = "0";
           $response["message"] = "Maaf , terjadi kesalahan";

              // echoing JSON response
              echo json_encode($response);
       }
```

5.1.5.3 *Implementasi Algoritma Proses Menampilkan Iklan Lowongan Kerja*

Proses menampilkan iklan lowongan kerja dilakukan dengan menekan menu *job* pada halaman *home* (kategori *jobseeker*). Pada halaman ini iklan-iklan lowongan kerja yang terdapat pada *database* server akan ditampilkan dalam bentuk *list* dimana iklan terbaru akan ditampilkan pada posisi paling atas.

Kode 5.5 Algoritma Proses Menampilkan Iklan (java)

```
protected String doInBackground(String... arg0) {

      String url = "http://"+ip+"/login_jobs/tampilkan.php";

      JSONParser jParser = new JSONParser();
      JSONObject json = jParser.getJSONFromUrl(url);
      try {
            contacts = json.getJSONArray("lowongan");
            for (int i = 0; i < contacts.length(); i++) {
                  JSONObject c = contacts.getJSONObject(i);
                  HashMap<String, String> map = new
HashMap<String, String>();
                  String id = c.getString("id_loker").trim();
                  String recruitment =
c.getString("recruitment").trim();
                  String  salary= c.getString("salary").trim();
                  String close = c.getString("date_close").trim();

                  map.put("id_loker", id);
                  map.put("recruitment", recruitment);
                  map.put("salary", salary);
                  map.put("date_close", close);

                  contactList.add(map);
                  }
            } catch (JSONException e) {
                  }
            return null;
     }

      protected void onPostExecute(String result) {
     // TODO Auto-generated method stub
            super.onPostExecute(result);
```

```
            pDialog.dismiss();
            ListAdapter adapter2 = new
SimpleAdapter(getApplicationContext(),contactList,
R.layout.insert_list_employee,
            new String[] { "recruitment", "salary", "date_close"
}, new int[] { R.id.E_list_recruitment, R.id.EP_recruitment,
            R.id.E_list_close });
            list.setAdapter(adapter2);
            list.setOnItemClickListener(new OnItemClickListener()
{
            @Override
      public void onItemClick(AdapterView<?> parent, View view,int
position, long id) {
            String id_loker = ((TextView)
view.findViewById(R.id.EP_recruitment)).getText().toString();
            Intent in = new Intent(getApplicationContext(),
employeer_inserjobPV.class);
            in.putExtra("id_loker", id_loker);

            startActivity(in);
      }
});
```

Kode 5.6 Algoritma Proses Menampilkan Iklan (php)

```
      response = array();
      $result = mysql_query("SELECT * FROM t_loker Order By
id_loker desc limit 1") or die (mysql_error());

      $response["t_loker"] = array();
      while ($row = mysql_fetch_array($result)) {
          $loker = array();
          $loker["id_loker"] = $row["id_loker"];
          $loker["recruitment"] = $row["recruitment"];
          $loker["location"] = $row["lokasi"];
          $loker["salary"] = $row["salary"];
          $loker["loker_detail"] = $row["loker_detail"];
          $loker["date_close"] = $row["date_close"];
          array_push($response["t_loker"], $loker);
      }
      echo json_encode($response);
```

5.1.5.4 *Implementasi Algoritma* Proses Mengirim Pesan

Proses mengirim pesan dilakukan dengan menekan menu *message* pada halaman *home* (kategori *employeer*). Pada halaman ini *user* (kategori *employeer*) dapat membuat dan mengirim pesan. Penerima pesan dibatasi hanya kepada *user* (kategori *jobseeker*) yang telah melakukan pendaftaran pada sebuah iklan lowongan kerja yang telah dibuat dan disimpan pada *database* server.

Kode 5.7 Algoritma Mengirim Pesan (java)

```
protected Void doInBackground(Void... params) {
      WeServiceHandler webservice = new WeServiceHandler();
      String jsonstr = webservice.getJsonData(url1+nama);
      try {
      JSONObject jsonObject = new JSONObject(jsonstr);
      SONArray jsonArray = jsonObject.getJSONArray("send");
      JSONObject testjson = jsonArray.getJSONObject(0);
       sendto=testjson.getString("penerima");
       sendisi=testjson.getString("isi");
       }
      catch (Exception e){
      e.printStackTrace();
      }
      return null;
}

 protected String doInBackground(String... params) {

       String strpenerima = to.getText().toString();
              String strisi = isi.getText().toString();
              String strtgl = tgl.getText().toString();
              String strid  = nama;
              String stridu = id;

              List<NameValuePair> nvp = new
ArrayList<NameValuePair>();
              nvp.add(new BasicNameValuePair("penerima",
strpenerima));
              nvp.add(new BasicNameValuePair("isi",strisi));
              nvp.add(new BasicNameValuePair("tanggal",strtgl));
            nvp.add(new BasicNameValuePair("perusahaan", strid));
            nvp.add(new BasicNameValuePair("id_user", stridu));

              JSONObject json = jsonParser.makeHttpRequest(url,
"POST", nvp);

              try {
                  success = json.getString("success");

              } catch (Exception e) {
                  Toast.makeText(getApplicationContext(), "Error",
                          Toast.LENGTH_LONG).show();
              }
      return null;
}
```

Kode 5.8 Algoritma Mengirim Pesan (php)

```
      $id_send       = $_POST['id_send'];
          $penerima     = $_POST['penerima'];
          $isi           = $_POST['isi'];
       $tanggal       =$_POST['tanggal'];
          $id_perusahaan = $_POST['id_perusahaan'];

      include "koneksi.php";
```

```
      $query = "INSERT INTO t_send
      (id_send,penerima,isi,tanggal,id_perusahaan)VALUES('$id_send
','$penerima','$isi','".date('Y-m-d',
strtotime($tanggal))."','$id_perusahaan')";
      $hasil = mysql_query($query);

      if($hasil)
       {
       $response["success"] = "1";
             $response["message"] = "Data sukses diinput";
             echo json_encode($response);
       }
       else
       {$response["success"] = "0";
           $response["message"] = "Maaf , terjadi kesalahan";

             // echoing JSON response
             echo json_encode($response);
       }
```

5.1.5.5 *Implementasi Algoritma JSON*

JSON adalah data berbentuk string yang untuk *parsing-parsing* data. Pada aplikasi ini JSON digunakan untuk menghubungkan data pada aplikasi android dengan *database* MySQL.

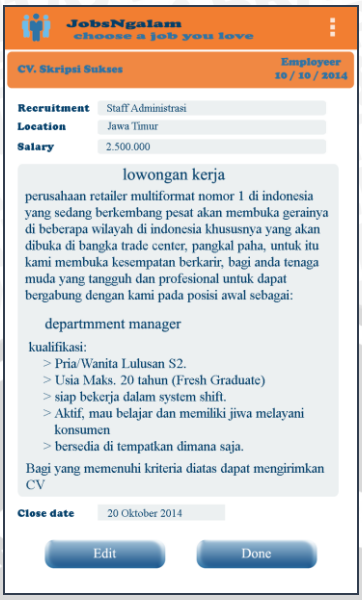
Kode 5.9 Algoritma JSON

```
public JSONObject makeHttpRequest(String url, String method,
                  List<NameValuePair> params) {

            try {
                  if (method == "POST") {
                        DefaultHttpClient httpClient = new
DefaultHttpClient();
                        HttpPost httpPost = new HttpPost(url);
                        httpPost.setEntity(new
UrlEncodedFormEntity(params));

                        HttpResponse httpResponse =
httpClient.execute(httpPost);
                        HttpEntity httpEntity =
httpResponse.getEntity();
                        is = httpEntity.getContent();

                  } else if (method == "GET") {
                        // request method is GET
                        DefaultHttpClient httpClient = new
DefaultHttpClient();
                        String paramString =
URLEncodedUtils.format(params, "utf-8");
                        url += "?" + paramString;
                        HttpGet httpGet = new HttpGet(url);
```

```
                        HttpResponse httpResponse =
httpClient.execute(httpGet);
                        HttpEntity httpEntity =
httpResponse.getEntity();
                        is = httpEntity.getContent();
                  }

            } catch (UnsupportedEncodingException e) {
                  e.printStackTrace();
            } catch (ClientProtocolException e) {
                  e.printStackTrace();
            } catch (IOException e) {
                  e.printStackTrace();
            }

            try {
                  BufferedReader reader = new BufferedReader(new
InputStreamReader(
                              is, "iso-8859-1"), 8);
                  StringBuilder sb = new StringBuilder();
                  String line = null;
                  while ((line = reader.readLine()) != null) {
                        sb.append(line + "\n");
                  }
                  is.close();
                  json = sb.toString();
            } catch (Exception e) {
                  Log.e("Buffer Error", "Error converting result "
+ e.toString());
            }

            // try parse the string to a JSON object
            try {
                  jObj = new JSONObject(json);
            } catch (JSONException e) {
                  Log.e("JSON Parser", "Error parsing data " +
e.toString());
            }

            // return JSON String
            return jObj;

      }
    }
```

5.1.6 ImplementasiAntarmuka

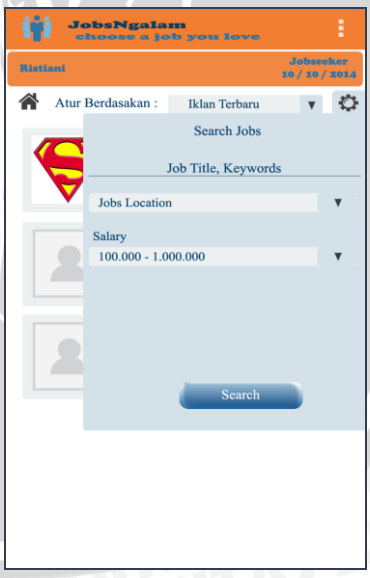
Implementasi antarmuka aplikasi sistem penerimaan lowongan kerja terdiri dari *create job advert*, *list registrant*, *view CV*, *send CV*. *Create job* merupakan implementasi antarmuka dari *use case* kelola *job advert* yang menampilkan *create job advert*, *edit job advert*, *view create detail job advert*. Tampilan *create job advert* ditunjukan dalam Gambar 5.1.



Gambar 5.1 Tampilan Antarmuka *Create Job Advert*

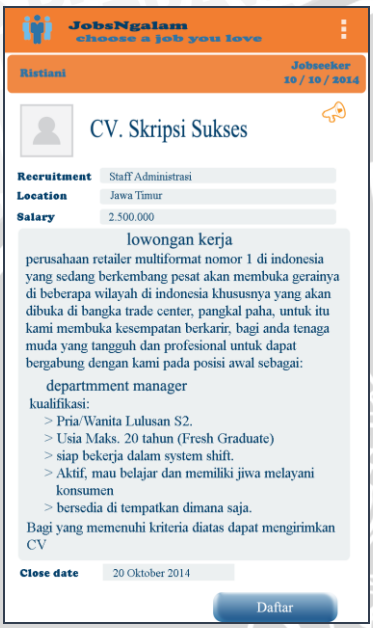
Search Job merupakan implementasi antarmuka dari *use case search job advert* yang menampilkan beberapa iklan lowongan kerja yang berada pada *database*. Pada halaman ini *user* dapat mengatur beberapa iklan lowongan kerja yang ingin ditampilkan. Tampilan *Search job* ditunjukan dalam Gambar 5.2 dan Gambar 5.3.

Gambar 5.2 Tampilan Antarmuka *Search Job*



Gambar 5.3 Tampilan Antarmuka *Search Job Setting*

Proses pendaftaran iklan lowongan kerja pada aplikasi ini dilakukan dengan mengirimkan CV. Proses ini diimplementasi antarmuka dari *use case view* detail *job* dan *send* CV dimana aplikasi menampilkan detail kriteria dari iklan lowongan kerja yang kemudian diarahkan ke halaman untuk mengirimkan CV. Tampilan *detail job* dan *send* CV ditunjukan dalam Gambar 5.4 dan Gambar 5.5.



Gambar 5.4 Tampilan Antarmuka *Detail Job Advert*

Gambar 5.5 Tampilan Antarmuka *List Job Advert*

List registrant merupakan implementasi antarmuka dari *use case list registrant* yang menampilkan *view CV*. Tampilan *list registrant* ditunjukan dalam Gambar 5.6.

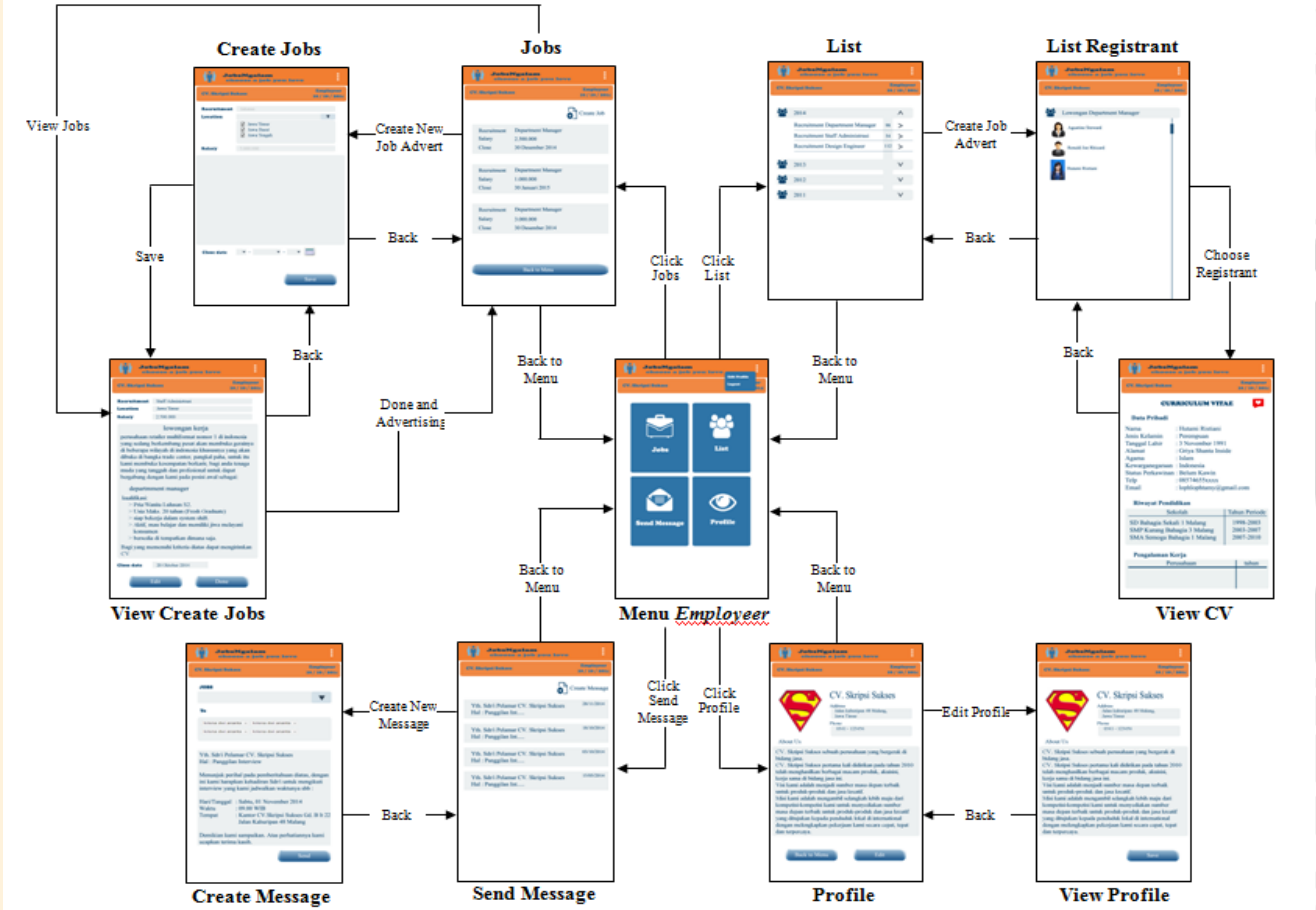
Gambar 5.6 Tampilan Antarmuka *List Registrant*

View CV merupakan implementasi antarmuka dari *use case view* CV yang menampilkan *view* CV. Tampilan *view* CV ditunjukan dalam Gambar 5.7.

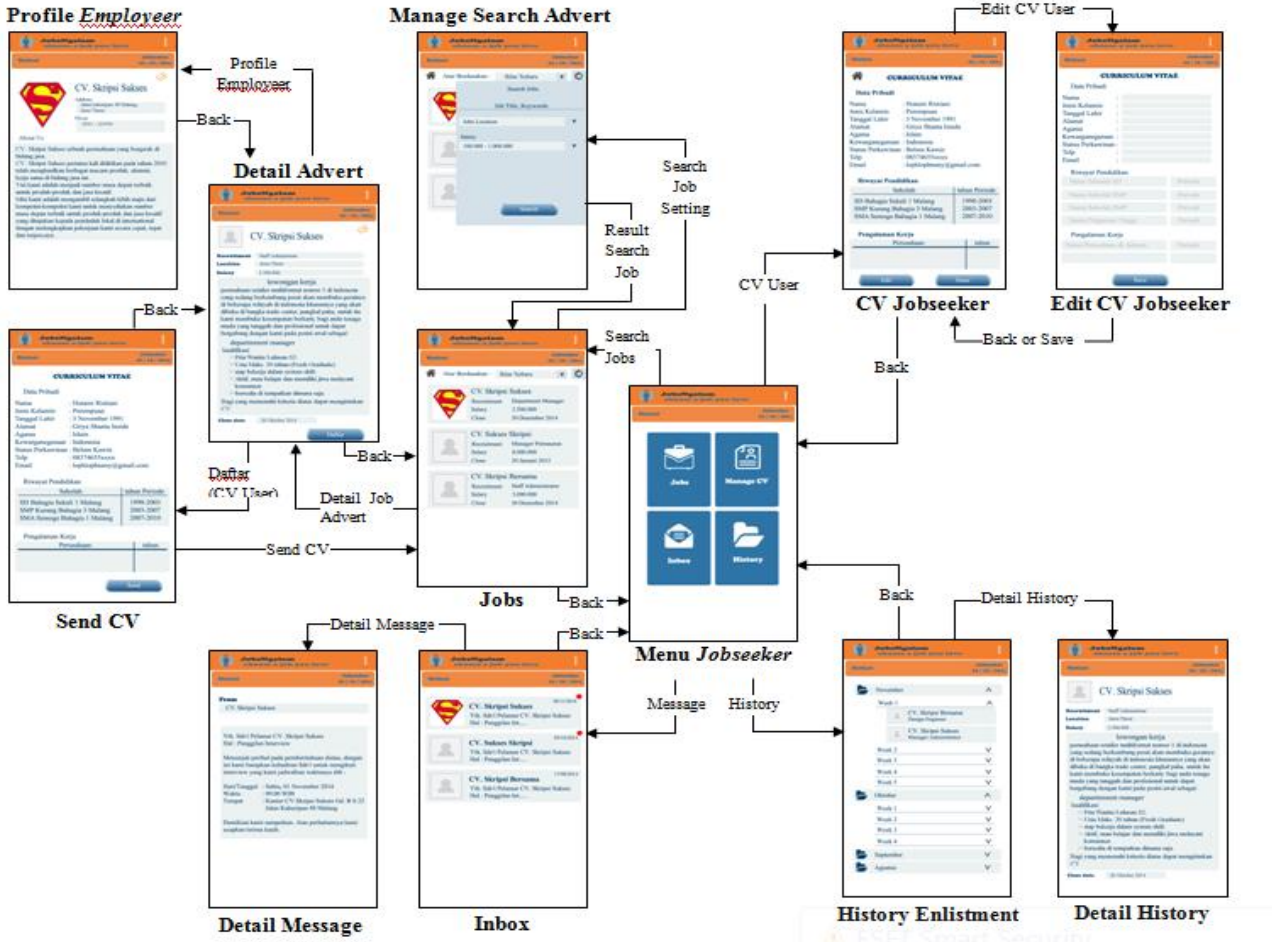
Gambar 5.7 Tampilan Antarmuka *View CV*

5.1.7 *Storyboard*

Storyboard adalah rancangan umum suatu aplikasi yang disusun secara berurutan *layer* demi *layer* serta dilengkapi dengan penjelasan dan spesifikasi dari setiap gambar, *layer*, dan teks. *Storyboard* digunakan untuk merancang antarmuka yang merupakan bagian dari program yang berhubungan atau berinteraksi langsung dengan *user*.



Gambar 5.1 *Storyboard Employeer*



Gambar 5.2 *Storyboard Jobseeker*

5.2 Pengujian

Pada tahap pengujian aplikasi sistem penerimaan lowongan kerja menggunakan *platform* android ini dilakukan dengan pengujian *black-box testing* yaitu pengujian validasi, tahapan dari pengujian tersebut akan dijelaskan pada sub bab berikut.

5.2.1 Pengujian Validasi

Pengujian validasi menggunakan metode *Black Box* dimana pengujian yang dilakukan adalah pengujian fungsionalitas dari sistem, apakah sistem berfungsi dengan hasil yang diinginkan atau tidak. Pengujian validasi ini didasarkan atas *use case* yang telah dibuat sebelumnya. Berikut tabel hasil pengujian validasi sistem.

Tabel 5.6 Pengujian Validasi

No	Kasus Uji	Prosedur dan Input	Kondisi yang diharapkan	Hasil
1	*SignUp*	Daftar dengan nama "hutami", *e-mail* "tami@gmail.com", *password* "123", dan memilih kategori "Jobseeker" atau "Employeer"	• Ditampilkan halaman *SignUp* yang berisi kolom input nama, *e-mail*, *password*, *category*. • Jika *username* dan *password* benar, maka *user* akan masuk ke halaman menu *home Jobseeker* atau *Employeer*.	Valid
2	*Login*	*Login* dengan username "tami@gmail.com" dan password "123"	• Ditampilkan halaman *login* yang berisi kolom input *username* dan *password*. • Jika *username* dan *password* benar, maka *user* akan masuk ke halaman home *jobseeker* atau *employeer*.	Valid
		Login dengan *username* dan *password* salah	• Muncul notifikasi "*username* atau *password* salah". • Tetap berada di halaman *login*.	Valid
3	Membuat iklan	Memilih fungsi "*Create*	• Menampilkan	Valid

	Jobs Employeer	*Job*" dan memasukkan data *recruitment* "Staff Administrasi", *location* "Jawa Timur", *salary* "2.500.000", mengisi *text area* dengan "Kualifikasi pria/wanita lulusan S2, usia maksimal 20 tahun (*Fresh Graduate*)" dan *close date* "Apr/12/2015"	halaman yang berisi kolom *recruitment*, *location*, *salary*, *field*, dan *close date* sesuai data yang dimasukkan. • Jika data iklan yang dimasukkan semuanya telah terisi akan kembali ke menu *Jobs Employeer*.	
		Belum lengkap memasukkan data iklan dan memilih tombol *save*.	• Muncul notifikasi "Data tidak boleh kosong".	Valid
4	Melihat daftar iklan perusahaan *Jobs Jobseeker*	Memilih salah satu iklan yang dipasang oleh perusahaan, kemudian klik tombol daftar	• Menampilkan daftar iklan dari berbagai perusahaan.	Valid
5	Membuat CV di menu *manage CV*	Isikan dengan nama "Hutami Ristiani", Jenis Kelamin "Wanita", Tanggal lahir "03 November 1991", Alamat "Griya Shanta Malang", Agama "Islam", Kewarganegaraan "Indonesia", Status	• Menampilkan halaman yang berisi kolom input nama, jenis kelamin, tanggal lahir, alamat, agama, kewarganegaraan, status perkawinan, telpn, email, dan kolom *input hint*	Valid

		Perkawinan "Belum Kawin", Telpn "081234567", Email "tami@gmail.com", Sekolah "SD Bahagia Sekali 1 Malang", Tahun Periode "2000-2006", Perusahaan "CV. Skripsi Sukses" Tahun Masuk "2020".	nama sekolah sd dan tahun periode, nama sekolah SMP dan tahun periode, nama sekolah SMA dan tahun periode, nama universitas dan tahun periode, nama perusahaan dan tahun masuk kerja.	
		Belum lengkap memasukkan data CV dan memilih tombol *save*.	• Muncul notifikasi "Data tidak boleh kosong".	Valid

Sumber : Analisis

5.2.2 Pengujian *Usability*

Pengujian *usability* adalah pengujian terhadap kualitas yang menilai seberapa mudah antarmuka yang dijalankan oleh pengguna. Dengan menggunakan teknik kuesioner untuk mempermudah pengujian *usability* ini. Sampel yang nantinya dipakai berjumlah tiga belas sampel secara acak, diantaranya sebelas kuesioner diberikan kepada pencari kerja dan dua kuesioner diberikan kepada penyedia lowongan pekerjaan. Pertanyaan yang dibuat dalam kuesioner ini meliputi tingkat kemudahan dalam penggunaan aplikasi, antarmuka aplikasi, respon aplikasi terhadap fungsi-fungsi yang dikerjakan dan kinerja aplikasi. Pertanyaan-pertanyaan tersebut akan disertakan sebagai lampiran. Hasil dari pengujian *usability* ditunjukkan pada Tabel 5.7.

Tabel 5.7 Hasil pengujian *usability*

No.	PERNYATAAN	SS	S	TS	STS	Total
1.	Fitur-fitur pada aplikasi Sistem	8	3	2	0	13

	Penerimaan Lowongan Kerja ini mudah dipahami					
2.	Pengoperasian aplikasi Sistem Penerimaan Lowongan Kerja ini mudah digunakan	7	6	0	0	13
3.	Tampilan di aplikasi Sistem Penerimaan Lowongan Kerja sudah terlihat menarik	10	3	0	0	13
4.	Aplikasi Sistem Penerimaan Lowongan Kerja ini sudah cukup membantu memudahkan dalam pencarian pekerjaan di masa mendatang	12	1	0	0	13
5.	Apabila aplikasi Sistem Penerimaan Lowongan Kerja *release* di Google Play Store, Anda berminat untuk mengunduh versi berbayar	10	1	2	0	13

Keterangan :

SS = Sangat Setuju
S = Setuju
TS = Tidak Setuju
STS = Sangat Tidak Setuju

5.2.3 Analisis Hasil Pengujian

Proses analisis terhadap hasil pengujian dilakukan untuk mendapatkan kesimpulan dari hasil pengujian aplikasi sistem penerimaan lowongan kerja menggunakan *platform* android yang telah selesai dilakukan. Proses analisis mengacu pada hasil pengujian yang didapatkan. Analisis dilakukan terhadap hasil pengujian di setiap tahap pengujian. Proses analisis yang dilakukan meliputi analisis hasil pengujian validasi, dan analisis hasil pengujian *usability*.

5.2.3.1 Analisis Hasil Pengujian Validasi

Proses analisis terhadap hasil pengujian validasi dilakukan dengan melihat kerjasama antara hasil kinerja sistem dengan sistem dengan daftar kebutuhan. Berdasarkan hasil pengujian validasi dapat disimpulkan bahwa implementasi dan

fungsionalitas sistem aplikasi *mobile* sistem penerimaan lowongan kerja menggunakan *platform* android telah memenuhi kebutuhan yang dijabarkan pada tahap analisis kebutuhan.

5.2.3.2 *Analisis Hasil Pengujian Usability*

Proses analisis terhadap hasil pengujian *usability* dilakukan dengan menggunakan skala Likert. Pengujian *usability* atau kemudahan penggunaan aplikasi layak diterima atau memasukkan jika *index* persentase yang dihasilkan dari setiap pernyataan lebih dari 50%. Interpretasi skor Likert atau persentase dari setiap skor Likert ditunjukkan pada Tabel 5.8. Hasil perhitungan *index* persentase dari setiap pernyataan ditunjukkan pada Tabel 5.9 dan hasil status pengujian *usability* ditunjukkan pada Tabel 5.10.

Tabel 5.8 Implementasi Skor Likert

Skor Likert	Interpretasi skor dengan interval = 25	Pilihan
1	**0% - 24,99%**	Sangat Tidak Memuaskan
2	**25% - 49,99%**	Tidak Memuaskan
3	**50% - 74,99%**	Memuaskan
4	**75% - 100%**	Sangat Memuaskan

Keterangan :

Interval = 25 didapatkan dari pembagian nilai 100 dengan jumlah skor Likert.

Tabel 5.9 *Index* Persentase

No.	Pernyataan	SS	S	TS	STS	Total Skor	Index (%)
1.	Fitur-fitur pada aplikasi Sistem Penerimaan Lowongan Kerja ini mudah dipahami	8	3	2	0	45	86%
2.	Pengoperasian aplikasi Sistem Penerimaan Lowongan Kerja ini mudah digunakan	7	6	0	0	46	88%
3.	Tampilan di aplikasi Sistem Penerimaan Lowongan Kerja sudah terlihat menarik	10	3	0	0	49	94%
4.	Aplikasi Sistem Penerimaan Lowongan Kerja ini sudah cukup membantu memudahkan dalam pencarian pekerjaan di masa mendatang	12	1	0	0	51	98%
5.	Apabila aplikasi Sistem Penerimaan Lowongan Kerja *release* di Google Play Store, Anda berminat untuk mengunduh versi berbayar	10	1	2	0	47	90%

Keterangan :

SS = Sangat Setuju
S = Setuju
TS = Tidak Setuju
STS = Sangat Tidak Setuju

Total Skor = Ssts x 1 + Sts x 2 + Ss x 3 + Sss x 4 ……(1)

Index (%) = Total Skor / Y x 100 ……..…(2)

Y : Skor Likert tertinggi x Jumlah audien

Tabel 5.10 Status pengujian *Usability*

Aspek Penilaian	Persentase (%)	Status
Fitur-fitur pada aplikasi Sistem Penerimaan Lowongan Kerja ini mudah dipahami	86%	Sangat Memuaskan
Pengoperasian aplikasi Sistem Penerimaan Lowongan Kerja ini mudah digunakan	88%	Sangat Memuaskan
Tampilan di aplikasi Sistem Penerimaan Lowongan Kerja sudah terlihat menarik	94%	Sangat Memuaskan
Aplikasi Sistem Penerimaan Lowongan Kerja ini sudah cukup membantu memudahkan dalam pencarian pekerjaan di masa mendatang	98%	Sangat Memuaskan
Apabila aplikasi Sistem Penerimaan Lowongan Kerja *release* di Google Play Store, Anda berminat untuk mengunduh versi berbayar	90%	Sangat Memuaskan

Kesimpulan :

Dari hasil pengujian *usability* yang telah dilakukan, bahwa index persentase (%) yang didapatkan pada setiap pernyataan menunjukkan bahwa hasil rata-rata persentase yang didapat adalah dengan status sangat memuaskan. Dengan adanya hasil tersebut menunjukkan bahwa aplikasi layak diterima dan memudahkan pengguna dengan hasil sangat memuaskan.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil analisis, perancangan, implementasi dan pengujian yang dilakukan, maka diambil kesimpulan sebagai berikut:

1. Perancangan aplikasi penerimaan lowongan kerja pada perangkat bergerak berbasis android dibuat berdasarkan UML (*Unified Modeling Language*), dan didesain sesuai dengan spesifikasi kebutuhan yang telah dianalisa sebelumnya yaitu: *login, sign up*, *list job advert*, *view detail job advert, search job advert*, *create job advert, update job advert*, *history*, *detail history view*, *view message*, *create CV*, *view CV*, *edit CV*, *profile*, *edit profile*, pengaduan, *list history job advert*, *list registrant*, *view CV registrant*, *send message*, *view* pengaduan, *view profile member*, dan manipulasi data.
2. Aplikasi penerimaan lowongan kerja diimplementasikan sesuai dengan perancangan dengan menggunakan bahasa pemrograman java untuk pengolahan sistem pada *device mobile* dan Json untuk pengolahan data antara *device mobile* dengan *database*.
3. Pada pengujian *Black Box testing* yang dilakukan pada sistem memberikan hasil nilai dengan prosentase 100% sistem telah memenuhi spesifikasi kebutuhan yang telah dianalisa. Dan pada pengujian *usability* yang dilakukan menunjukan bahwa index presentase yang didapat seperti Tabel 5.10 menunjukan presentase status pengujian dengan nilai rata-rata 91% menunjukan bahw*a* sistem penerimaan lowongan kerja *mobile* berbasis android ini memiliki fitur-fitur yang mudah dipahami, pengoperasian (navigasi) yang mudah digunakan dan tampilan yang terlihat menarik.

6.2 Saran

Saran yang dapat diberikan untuk sistem aplikasi penerimaan lowongan kerja ini menggunakan platform android selanjutnya antara lain adalah :

1. Melakukan pengembangan untuk beberapa operasi sistem *mobile* lain yaitu Windows Phone, Blackberry, iOS dan symbian.

2. Dalam pengembangan selanjutnya aplikasi ini perlu adanya *team* yang dapat membantu admin dalam melakukan verifikasi keaslian ikan, sehingga dalam melakukan manipulasi data member, admin dapat memastikan keaslian dari iklan lowongan kerja.

DAFTAR PUSTAKA

[ABV-14] Arshdeep Bahga, Vijay Madisetti. 2014. "Cloud Computing: A Hands-On Approach". Georgia Institute of Technology, Atlanta, USA.

[ASR-11] A.S, Rosa. 2011. "Modul Pembelajaran Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Objek)". Modula. Bandung.

[DKB-12] Douglas K. Barry. 2012. "Web Services, Service-Oriented Architectures, and Cloud Computing: The Savvy Manager's", Elsevier, Inc USA.

[GOD-13] Godfrey Nolan, David Truxall, Onur Cinar. 2013. "Android Best Practice". Addison-Wesley. 167-168.

[HVH-10] Helko Van Der Hoom. 2010. "A Survey of Mobile Platforms for Pervasive Computing". Groningen.

[NSH-11] Nazaruddin Safaat H. 2011. "*Android Pemrograman Aplikasi Mobile Smartphone dan Tablet PC Berbasis Android"*. Informatika. Bandung.

[PRE-12] Pressman, Roger. 2012. "Rekayasa Perangkat lunak : Pendekatan Praktisi (Edisi 7)". Andi dan McGrow - Hil Book co. Yogyakarta.

[RSP-13] Ronan Schwarz, Phil Dutson, James Steele, Nelson To. 2013." The Android Developer's Cookbook: Building Applications with the Android SDK". Addison-Wesley.

[WKR-10] Wahana Komputer. 2010. "*The Best 40 Java Application"*. PT Elex Media Komputindo, Jakarta.

LAMPIRAN

Lampiran 1: Skenario *Use Case*

Skenario *Use Case Sign Up*

Nama	*Sign Up*
Tujuan	Untuk melakukan pendaftaran pada sistem
Deskripsi	*Use case* ini menjelaskan *user* yang belum memiliki akun pada sistem dapat melakukan pendaftaran pada sistem
Aktor	*User*
Kondisi Awal	Aplikasi telah di-*download* dan ter-*install*, pengguna membuka aplikasi
Alur Usecase	• Pengguna memilih fungsi *Sign Up* • Pengguna mengisi *form* pendaftaran yang tersedia
Kondisi Akhir	Aplikasi menampilkan *form* pendaftaran

Skenario *Use case Login*

Nama	*Login*
Tujuan	Untuk melakukan *Login* pada sistem
Deskripsi	*Use case* ini menjelaskan *user* yang telah memiiki akun pada sistem dan belum masuk dalam sistem
Aktor	*User*
Kondisi Awal	Aplikasi telah di-*download* dan ter-*install*, pengguna membuka aplikasi
Alur Usecase	• Pengguna memilih fungsi *Login* • Pengguna mengisi *username* & *password* akun miliknya
Kondisi Akhir	Aplikasi menampilkan *form Login*

Skenario *Use Case Profile Employeer*

Nama	*Profile Employeer*
Tujuan	Untuk melihat profil

Deskripsi	*Use case* ini menjelaskan *user* dengan kategori *employeer* dapat melihat profil akunnya
Aktor	*Employeer*
Kondisi Awal	Aplikasi telah di-*download* dan ter-*install*, pengguna membuka aplikasi dan telah *login*
Alur Usecase	• Masuk kehalaman *home* pada aplikasi • Pengguna memilih menu Profil • Pengguna melihat profil akun aplikasinya
Kondisi Akhir	Aplikasi menampilkan profil diri dari akun pengguna

Skenario *Use Case Edit Profile Employeer*

Nama	*Edit Profile Emp*
Tujuan	Untuk mengedit profil
Deskripsi	*Use case* ini menjelaskan *user* dengan kategori *employeer* dapat mengedit profil akunnya
Aktor	*Employeer*
Kondisi Awal	Aplikasi telah di-*download* dan ter-*install*, pengguna membuka aplikasi dan telah *login*
Alur Usecase	• Masuk kehalaman *home* pada aplikasi • Pengguna memilih menu Profil • Pengguna melihat profil akun aplikasinya • Pengguna mengedit profil yang sudah ada
Kondisi Akhir	Aplikasi menampilkan profil diri dari akun pengguna

Skenario *Use Case View Profile Employeer*

Nama	*View Profile Emp*
Tujuan	Untuk melihat profil *employeer*
Deskripsi	*Use case* ini menjelaskan *user* dengan kategori *jobseeker* dapat melihat profil dari *employeer*
Aktor	*jobseeker*
Kondisi Awal	Aplikasi telah di-*download* dan ter-*install*,

	pengguna membuka aplikasi dan telah *login*
Alur Usecase	• Masuk kehalaman *home* pada aplikasi • Pengguna memilih menu *Jobs* • Pengguna melihat-lihat iklan yang tersedia • Pengguna memilih salah satu iklan pada *list job advert* • Pengguna melihat detail dari iklan lowongan pekerjaan • Pengguna menekan nama perusahaan pada iklan
Kondisi Akhir	Aplikasi menampilkan profil *employeer* dari akun pembuat iklan lowongan kerja

Skenario *Use Case View List Job Advert*

Nama	*View List Job Advert*
Tujuan	Untuk Melihat daftar iklan lowongan pekerjaan
Deskripsi	*Use case* ini menjelaskan *user* dengan kategori *jobseeker* dapat melihat berbagai iklan lowongan pekerjaan yang terdapat pada *database* aplikasi
Aktor	*Jobseeker*
Kondisi Awal	Aplikasi telah di-*download* dan ter-*install*, pengguna membuka aplikasi dan telah *login*
Alur Usecase	• Masuk kehalaman *home* pada aplikasi • Pengguna memilih menu *Jobs* • Pengguna melihat-lihat iklan yang tersedia
Kondisi Akhir	Aplikasi menampilkan daftar berbagai lowongan pekerjaan pada *database*

Skenario *Use Case View Detail Job Advert*

Nama	*View Detail Job Advert*
Tujuan	Untuk Melihat Detail iklan lowongan pekerjaan
Deskripsi	*Use case* ini menjelaskan *user* dengan kategori *jobseeker* dapat melihat detail dari iklan lowongan pekerjaan

Aktor	*Jobseeker*
Kondisi Awal	Aplikasi telah di-*download* dan ter-*install*, pengguna membuka aplikasi dan telah *login*
Alur Usecase	• Masuk kehalaman *home* pada aplikasi • Pengguna memilih menu *Jobs* • Pengguna melihat-lihat iklan yang tersedia • Pengguna memilih salah satu iklan pada *list job advert* • Pengguna melihat detail dari iklan lowongan pekerjaan
Kondisi Akhir	Aplikasi menampilkan detail lowongan pekerjaan pada *database*

Skenario *Use Case Search Job Advert*

Nama	*Search Job Advert*
Tujuan	Untuk Mencari iklan lowongan pekerjaan
Deskripsi	*Use case* ini menjelaskan *user* dengan kategori *jobseeker* dapat mencari iklan lowongan pekerjaan yang diinginkan berdasarkan kategori tertentu
Aktor	*Jobseeker*
Kondisi Awal	Aplikasi telah di-*download* dan ter-*install*, pengguna membuka aplikasi dan telah *login*
Alur Usecase	• Masuk kehalaman *home* pada aplikasi • Pengguna memilih menu *Jobs* • Pengguna melihat-lihat iklan yang tersedia • Pengguna memilih menu *setting* • Pengguna menentukan batasan / kategori yang diingikan • Pengguna melakukan pencarian
Kondisi Akhir	Aplikasi menampilkan daftar lowongan pekerjaan berdasarkan kategori tertentu sesuai keinginan *jobseeker*

Skenario *Use Case Edit Job Advert*

Nama	*Edit Job Advert*

Tujuan	Untuk mengedit iklan lowongan pekerjaan
Deskripsi	*Use case* ini menjelaskan *user* dengan kategori *employeer* dapat mengedit iklan lowongan pekerjaan yang telah dibuat sebelumnya apabila terdapat kesalahan
Aktor	*Employeer*
Kondisi Awal	Aplikasi telah di-*download* dan ter-*install*, pengguna membuka aplikasi dan telah *login*
Alur Usecase	• Masuk kehalaman *home* pada aplikasi • Pengguna memilih menu *Jobs* • Pengguna memilih *icon create job* • Pengguna membuat iklan lowongan kerja baru • Pengguna melihat hasil iklan lowongan pekerjaan yang telah dibuat • Pengguna memilih menu edit (jika terjadi kesalahan) • Pengguna melakukan pengeditan iklan
Kondisi Akhir	Aplikasi menampilkan *form* untuk mengedit iklan yang telah dibuat sebelumnya

Skenario *Use Case History*

Nama	*History*
Tujuan	Untuk melihat *History*
Deskripsi	*Use case* ini menjelaskan *user* dengan kategori *jobseeker* dapat melihat kembali iklan lowongan pekerjaan apa saja yang pernah diikuti (mendaftar)
Aktor	*Jobseeker*
Kondisi Awal	Aplikasi telah di-*download* dan ter-*install*, pengguna membuka aplikasi dan telah *login*
Alur Usecase	• Masuk kehalaman *home* pada aplikasi • Pengguna memilih menu *History* • Pengguna melihat *list History*
Kondisi Akhir	Aplikasi menampilkan *list history* lowongan pekerjaan yang pernah di ikuti

Skenario *Use Case* Detail *History View*

Nama	Detail *History View*
Tujuan	Untuk melihat Detail *History View*
Deskripsi	*Use case* ini menjelaskan *user* dengan kategori *jobseeker* dapat melihat secara detail iklan lowongan pekerjaan apa saja yang pernah diikuti (mendaftar)
Aktor	*Jobseeker*
Kondisi Awal	Aplikasi telah di-*download* dan ter-*install*, pengguna membuka aplikasi dan telah *login*
Alur Usecase	• Masuk kehalaman *home* pada aplikasi • Pengguna memilih menu *History* • Pengguna melihat list *History* • Pengguna memilih salah satu *history* pada list • Pengguna melihat detail iklan lowongan pekerjaan yang pernah diikuti
Kondisi Akhir	Aplikasi menampilkan secara detail lowongan pekerjaan yang pernah di ikuti

Skenario *Use Case View Message*

Nama	*View Message*
Tujuan	Untuk melihat pesan masuk
Deskripsi	*Use case* ini menjelaskan *user* dengan kategori *jobseeker* dapat menerima dan menampilkan pesan dari *employeer* sebagai balasan dari lowongan pekerjaan yang diikuti *jobseeker*
Aktor	*Jobseeker*
Kondisi Awal	Aplikasi telah di-*download* dan ter-*install*, pengguna membuka aplikasi dan telah *login*
Alur Usecase	• Masuk kehalaman *home* pada aplikasi • Pengguna memilih menu *Inbox* • Pengguna melihat list pesan masuk • Pengguna melihat isi pesan
Kondisi Akhir	Aplikasi menampilkan isi dari pesan

Skenario *Use Case Create CV*

Nama	*Create CV*
Tujuan	Untuk membuat *Curriculum Vitae*
Deskripsi	*Use case* ini menjelaskan *user* dengan kategori *jobseeker* dapat membuat *curriculum vitae*
Aktor	*Jobseeker*
Kondisi Awal	Aplikasi telah di-*download* dan ter-*install*, pengguna membuka aplikasi dan telah *login*
Alur Usecase	• Masuk kehalaman *home* pada aplikasi • Pengguna memilih menu *Manage* CV • Penggguna mengisi *form* untuk membuat CV jika masih belum pernah membuat CV sebelumnya
Kondisi Akhir	Aplikasi menampilkan *form* untuk membuat CV

Skenario *Use Case View CV*

Nama	*View CV*
Tujuan	Untuk Melihat *Curriculum Vitae*
Deskripsi	*Use case* ini menjelaskan *user* dengan kategori *jobseeker* dapat melihat *curriculum vitae* yang telah dibuat sebelumnya
Aktor	*Jobseeker*
Kondisi Awal	Aplikasi telah di-*download* dan ter-*install*, pengguna membuka aplikasi dan telah *login*
Alur Usecase	• Masuk kehalaman *home* pada aplikasi • Pengguna memilih menu *Manage* CV • Penggguna melihat CV yang telah dibuat sebelumnya
Kondisi Akhir	Aplikasi menampilkan CV yang telah dibuat

Skenario *Use Case Edit CV*

Nama	*Edit CV*
Tujuan	Untuk mengedit *Curriculum Vitae*
Deskripsi	*Use case* ini menjelaskan *user* dengan kategori

	jobseeker dapat mengedit *curriculum vitae* yang telah dibuat sebelumnya apabila ada kesalahan atau perubahan
Aktor	*Jobseeker*
Kondisi Awal	Aplikasi telah di-*download* dan ter-*install*, pengguna membuka aplikasi dan telah *login*
Alur Usecase	• Pengguna membuka aplikasi • Masuk kehalaman *home* pada aplikasi • Pengguna memilih menu *Manage* CV • Penggguna melihat CV yang telah dibuat sebelumnya • Pengguna memilih menu edit • Pengguna mengedit CV
Kondisi Akhir	Aplikasi menampilkan *form* edit pada CV yang telah dibuat

Skenario *Use Case List History Job Advert*

Nama	*List History Job Advert*
Tujuan	Untuk melihat *list history* lowongan pekerjaan
Deskripsi	*Use case* ini menjelaskan *user* dengan kategori *Employeer* dapat melihat *list history* lowongan pekerjaan yang pernah dibuat sebelumnya.
Aktor	*Employeer*
Kondisi Awal	Aplikasi telah di-*download* dan ter-*install*, pengguna membuka aplikasi dan telah *login*
Alur Usecase	• Pengguna membuka aplikasi • Masuk kehalaman *home* pada aplikasi • Pengguna memilih menu *List* • Pengguna melihat-lihat daftar iklan yang pernah dibuat
Kondisi Akhir	Aplikasi menampilkan *list history* lowongan pekerjaan

Skenario *Use Case List Registrant*

Nama	*List Registrant*

Tujuan	Untuk melihat daftar pendaftar
Deskripsi	*Use case* ini menjelaskan *user* dengan kategori *Employeer* dapat melihat daftar pendaftar yang mendaftar pada iklan lowongan pekerjaan yang telah dibuat *employeer* tersebut
Aktor	*Employeer*
Kondisi Awal	Aplikasi telah di-*download* dan ter-*install*, pengguna membuka aplikasi dan telah *login*
Alur Usecase	• Masuk kehalaman *home* pada aplikasi • Pengguna memilih menu *List* • Pengguna melihat-lihat daftar iklan yang pernah dibuat • Pengguna memilih salah satu iklan yang pernah dibuat • Pengguna melihat daftar pendaftar pada iklan tersebut
Kondisi Akhir	Aplikasi menampilkan *list history* lowongan pekerjaan

Skenario *Use Case View CV Registrant*

Nama	*View CV Registrant*
Tujuan	Untuk melihat CV pendaftar
Deskripsi	*Use case* ini menjelaskan *user* dengan kategori *Employeer* dapat melihat CV pendaftar pada iklan lowongan pekerjaan yang pernah dibuat
Aktor	*Employeer*
Kondisi Awal	Aplikasi telah di-*download* dan ter-*install*, pengguna membuka aplikasi dan telah *login*
Alur Usecase	• Masuk kehalaman *home* pada aplikasi • Pengguna memilih menu *List* • Pengguna melihat-lihat daftar iklan yang pernah dibuat • Pengguna memilih salah satu iklan yang pernah dibuat • Pengguna melihat daftar pendaftar pada iklan

	tersebut • Pengguna memilih salah satu pendaftar • Pengguna melihat CV pendaftar
Kondisi Akhir	Aplikasi menampilkan CV pendaftar

Skenario *Use Case Send Message*

Nama	*Send Message*
Tujuan	Untuk mengirim pesan
Deskripsi	*Use case* ini menjelaskan *user* dengan kategori *Employeer* dapat membuat dan mengirim pesan sebagai balasan pendaftar yang mendaftar pada iklan yang dibuat *employeer*
Aktor	*Employeer*
Kondisi Awal	Aplikasi telah di-*download* dan ter-*install*, pengguna membuka aplikasi dan telah *login*
Alur Usecase	• Masuk kehalaman *home* pada aplikasi • Pengguna memilih menu *Send Message* • Pengguna membuat pesan • Pengguna mengirim pesan
Kondisi Akhir	Aplikasi menampilkan *form* untuk membuat pesan

Lampiran 2: Perancangan *Activity* Diagram

Diagram *Activity Login, Sign Up*

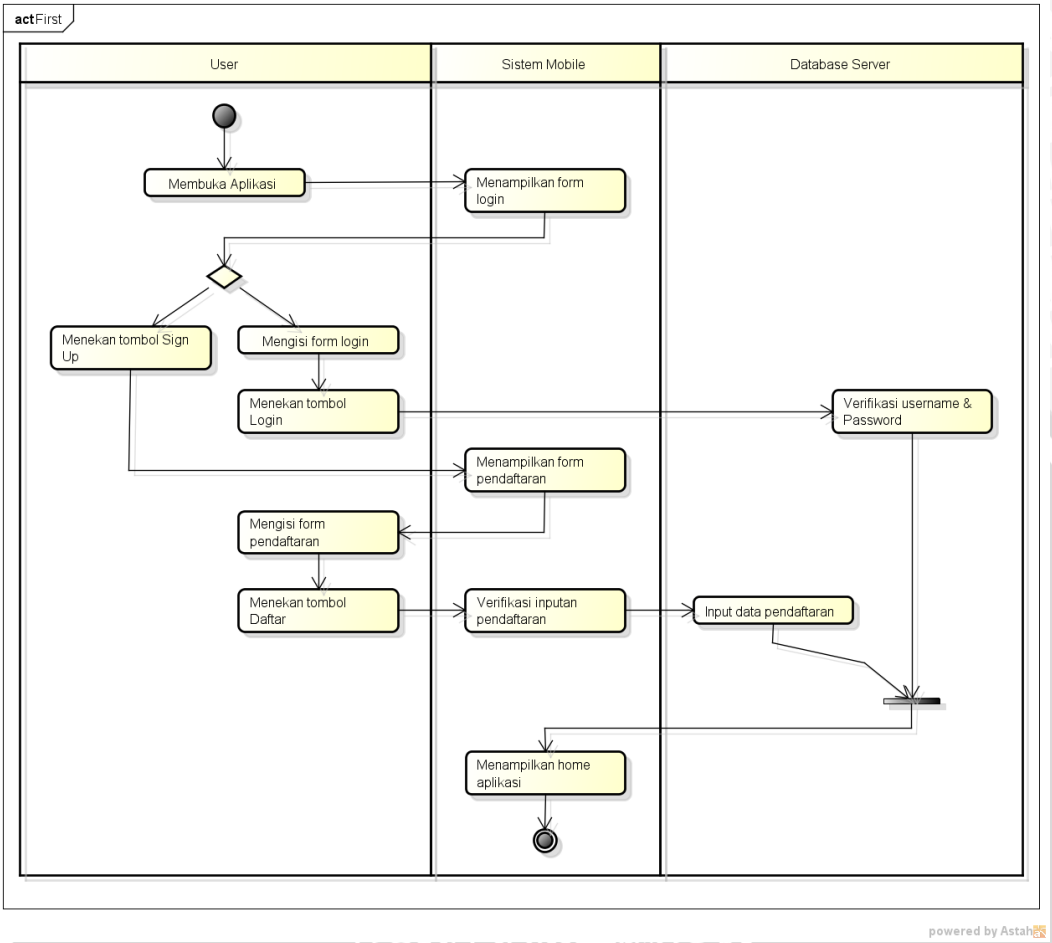


Diagram *Activity Profile Employeer*

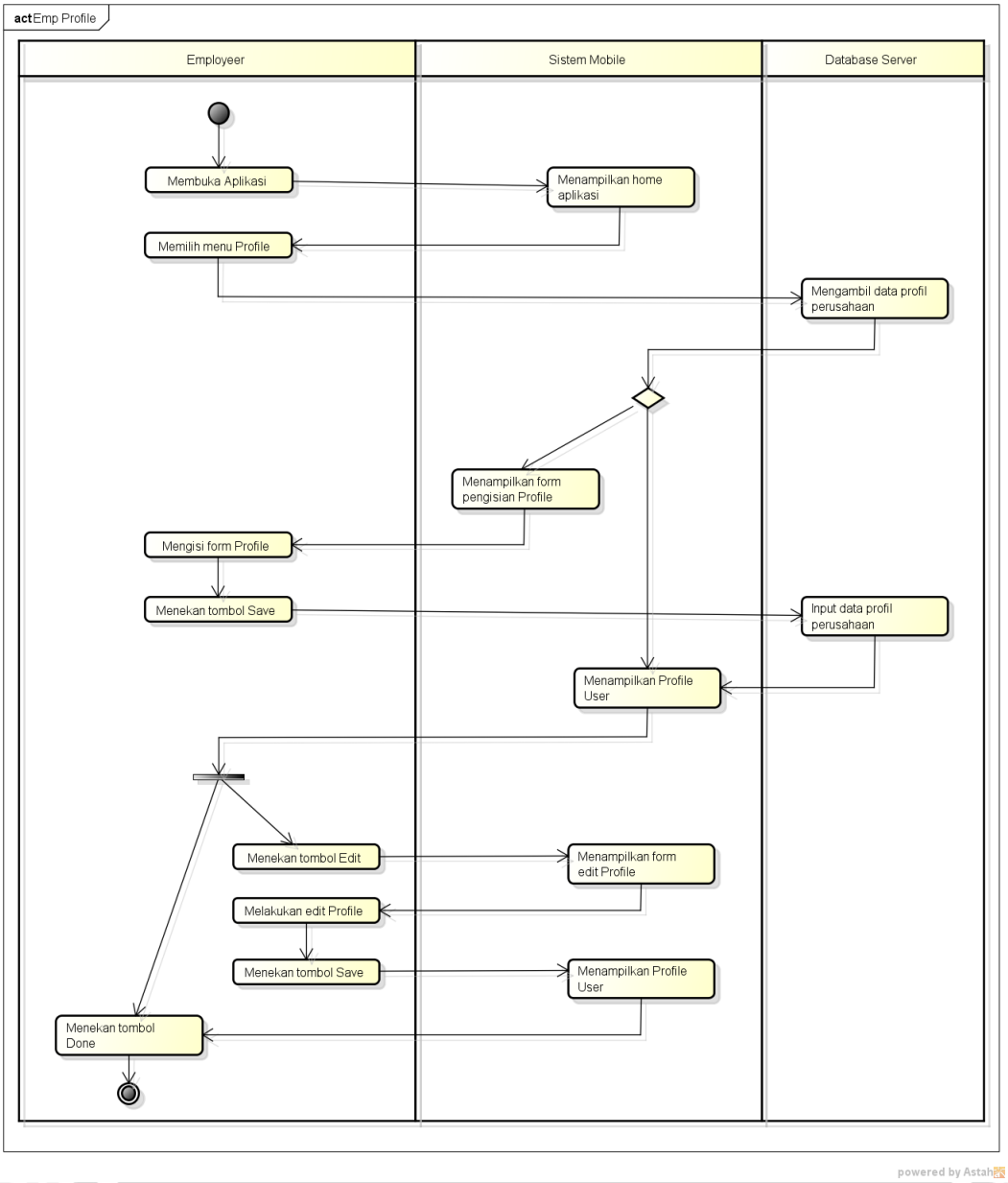


Diagram *Activity Manage CV*

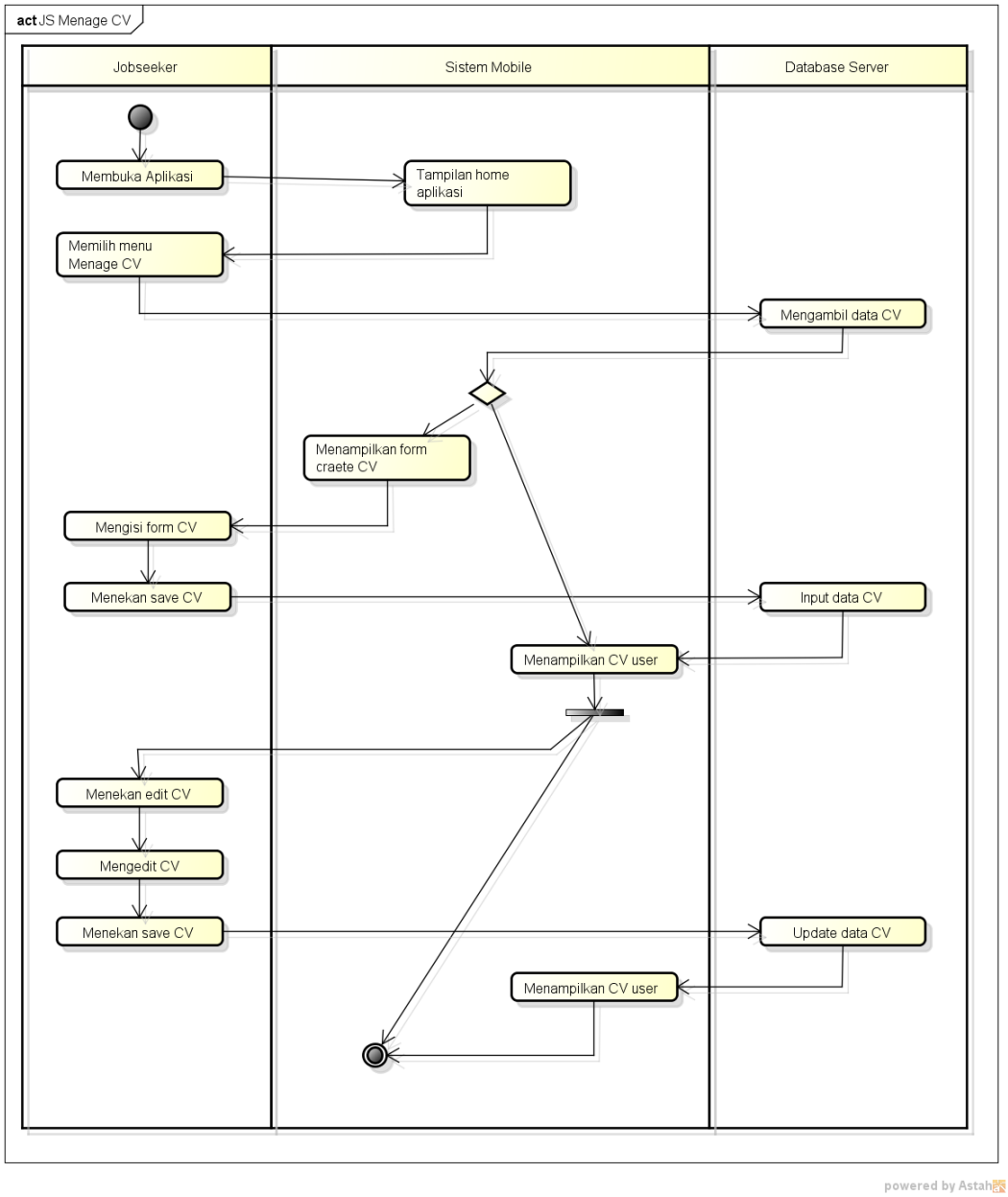


Diagram *Activity Send Message*

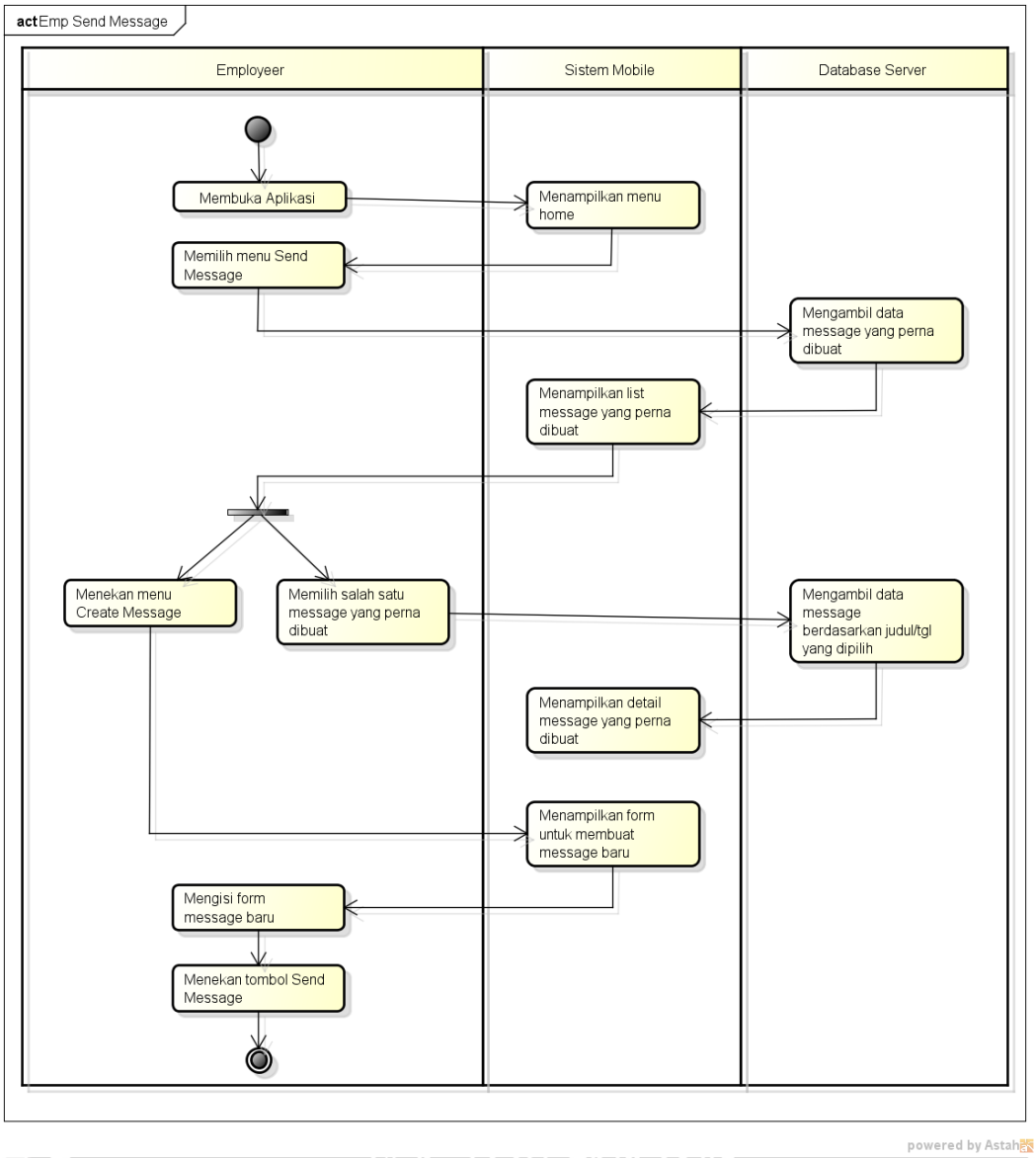
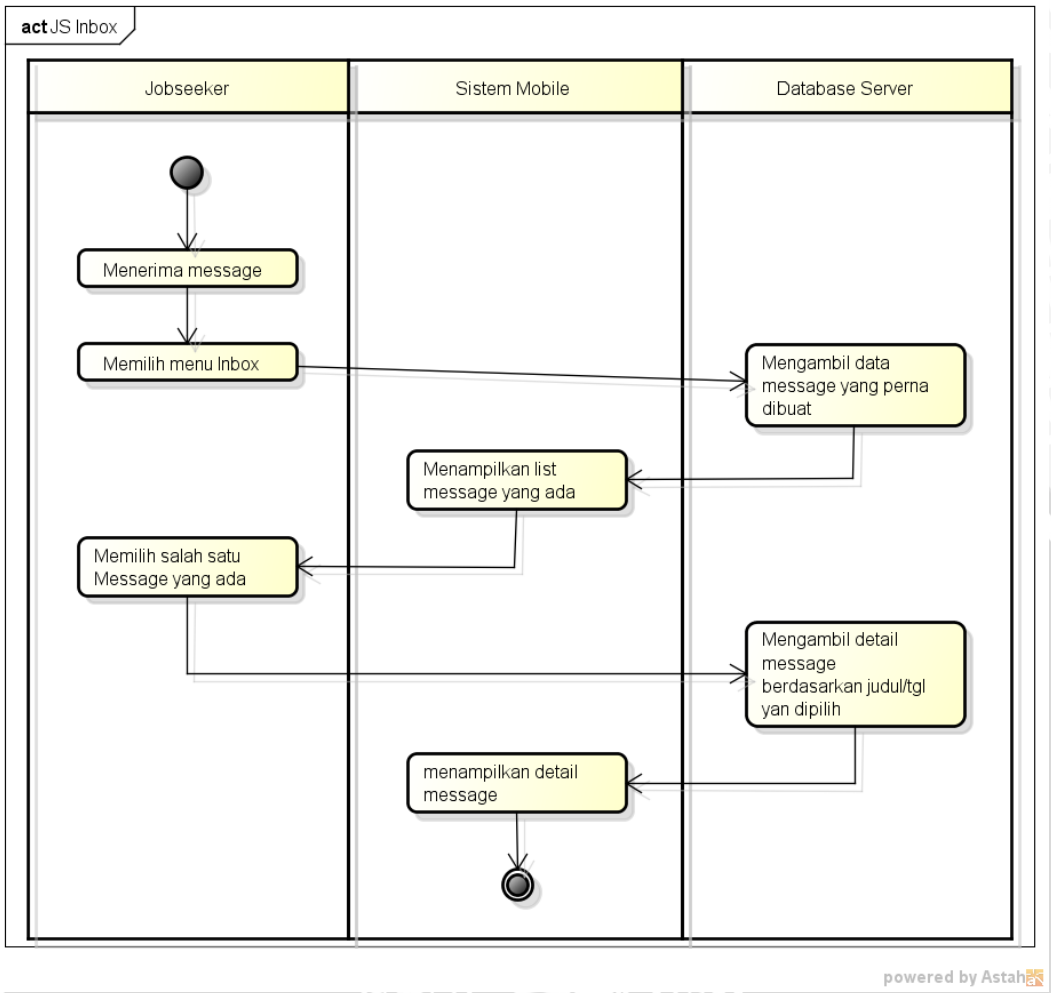


Diagram *Activity Inbox*



Lampiran 3: Hasil Pengisian Kuesioner

KUESIONER PENGUJIAN APLIKASI SISTEM PENERIMAAN LOWONGAN KERJA *MOBILE* BERBASIS ANDROID

Nama : Ach. faisol Baihaqqi / PT. Mulya Petro Danelindo

Umur : 25 Tahun

Setelah Anda menjalankan Aplikasi *Sistem Lowongan Kerja*, silahkan isi data-data dibawah ini dengan memberikan tanda **(X)** untuk setiap jawaban dari pernyataan di bawah ini yang menurut Anda paling tepat terkait aplikasi tersebut.

Keterangan :

SS = Sangat Setuju
S = Setuju
TS = Tidak Setuju
STS = Sangat Tidak Setuju

No.	PERNYATAAN	SS	S	TS	STS
1.	Fitur-fitur pada aplikasi Sistem Penerimaan Lowongan Kerja ini mudah dipahami		X		
2.	Pengoperasian aplikasi Sistem Penerimaan Lowongan Kerja ini mudah digunakan	X			
3.	Tampilan di aplikasi Sistem Penerimaan Lowongan Kerja sudah terlihat menarik		X		
4.	Aplikasi Sistem Penerimaan Lowongan Kerja ini sudah cukup membantu memudahkan dalam pencarian pekerjaan di masa mendatang	X			
5.	Apabila aplikasi Sistem Penerimaan Lowongan Kerja *release* di Google Play Store, Anda berminat untuk mengunduh versi premium	X			

KUESIONER PENGUJIAN APLIKASI SISTEM PENERIMAAN

LOWONGAN KERJA *MOBILE* BERBASIS ANDROID

Nama : Nurwidyaningtyas

Umur : 23 th

Setelah Anda menjalankan Aplikasi *Sistem Lowongan Kerja*, silahkan isi data-data dibawah ini dengan memberikan tanda **(X)** untuk setiap jawaban dari pernyataan di bawah ini yang menurut Anda paling tepat terkait aplikasi tersebut.

Keterangan :

SS = Sangat Setuju
S = Setuju
TS = Tidak Setuju
STS = Sangat Tidak Setuju

No.	PERNYATAAN	SS	S	TS	STS
1.	Fitur-fitur pada aplikasi Sistem Penerimaan Lowongan Kerja ini mudah dipahami	X			
2.	Pengoperasian aplikasi Sistem Penerimaan Lowongan Kerja ini mudah digunakan		X		
3.	Tampilan di aplikasi Sistem Penerimaan Lowongan Kerja sudah terlihat menarik	X			
4.	Aplikasi Sistem Penerimaan Lowongan Kerja ini sudah cukup membantu memudahkan dalam pencarian pekerjaan di masa mendatang		X		
5.	Apabila aplikasi Sistem Penerimaan Lowongan Kerja *release* di Google Play Store, Anda berminat untuk mengunduh versi premium	X			

KUESIONER PENGUJIAN APLIKASI SISTEM PENERIMAAN

LOWONGAN KERJA *MOBILE* BERBASIS ANDROID

Nama : Arif Dibisono

Umur :

Setelah Anda menjalankan Aplikasi *Sistem Lowongan Kerja*, silahkan isi data-data dibawah ini dengan memberikan tanda **(X)** untuk setiap jawaban dari pernyataan di bawah ini yang menurut Anda paling tepat terkait aplikasi tersebut.

Keterangan :

SS = Sangat Setuju
S = Setuju
TS = Tidak Setuju
STS = Sangat Tidak Setuju

No.	PERNYATAAN	SS	S	TS	STS
1.	Fitur-fitur pada aplikasi Sistem Penerimaan Lowongan Kerja ini mudah dipahami	X			
2.	Pengoperasian aplikasi Sistem Penerimaan Lowongan Kerja ini mudah digunakan		X		
3.	Tampilan di aplikasi Sistem Penerimaan Lowongan Kerja sudah terlihat menarik	X			
4.	Aplikasi Sistem Penerimaan Lowongan Kerja ini sudah cukup membantu memudahkan dalam pencarian pekerjaan di masa mendatang	X			
5.	Apabila aplikasi Sistem Penerimaan Lowongan Kerja *release* di Google Play Store, Anda berminat untuk mengunduh versi premium	X			