PENGEMBANGAN SISTEM INFORMASI PERPUSTAKAAN MENGGUNAKAN METODE *EXTREME PROGRAMMING* (STUDI PADA: SMK 1 MUHAMMADIYAH MALANG)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Pendidikan

Disusun oleh:
Rizky Edyatna Putra
NIM: 156150601111014

PROGRAM STUDI PENDIDIKAN TEKNOLOGI INFORMASI
JURUSAN SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

PENGEMBANGAN SISTEM INFORMASI PERPUSTAKAAN MENGGUNAKAN METODE *EXTREME PROGRAMMING* (STUDI PADA: SMK 1 MUHAMMADIYAH MALANG)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Pendidikan

Disusun Oleh :
Rizky Edyatna Putra
NIM: 156150601111014

Skripsi ini telah diuji dan dinyatakan lulus pada
17 Juli 2019
Telah diperiksa dan disetujui oleh:

Pembimbing I	Pembimbing II
Satrio Agung W., S.Kom., M.Kom.	Issa Arwani, S.Kom., M.Sc.
NIP: 19860521 201212 1 001	NIP: 19830922 201212 1 003

Mengetahui
Ketua Jurusan Sistem Informasi

Dr. Eng. Herman Tolle, S.T., M.T.
NIP: 19740823 200012 1 001

PERNYATAAN ORISINALITAS

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

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

Malang, 21 Juli 2019

Rizky Edyatna Putra

NIM: 156150601111014

PRAKATA

Puji syukur yang dapat penulis panjatkan atas kehadirat Allah Subhanahu Wa Ta'ala, karena berkat Rahmat dan Karunia-Nya, penulis dapat menyelesaikan laporan skripsi yang berjudul "PENGEMBANGAN SISTEM INFORMASI PERPUSTAKAAN MENGGUNAKAN METODE *EXTREME PROGRAMMING* (STUDI PADA: SMK 1 MUHAMMADIYAH MALANG)". Tidak lupa juga untuk menghaturkan Shalawat serta salam yang penulis panjatkan pada junjungan besar Nabi Muhammad Salallahu Alaihi Wa salam beserta keluarga, sahabat dan pengikutnya.

Penyusunan laporan ini tidak akan berhasil jika tidak ada bantuan dari berbagai pihak. Pada bagian ini, penulis ingin menghaturkan terima kasih dan rasa hormat kepada berbagai pihak yang telah membantu dalam pengerjaan skripsi ini kepada:

1. Bapak Satrio Agung Wicaksono, S.Kom, M.Kom., selaku pembimbing I yang selalu memberikan arahan, pengetahuan baru dan candaan yang membawa suasana gembira dan santai meski tetap serius dan sabar selama proses bimbingan dan pengerjaan skripsi ini, serta penulis dapat belajar banyak hal baru selama menimba ilmu kepada beliau.
2. Bapak Issa Arwani, S.Kom, M.Sc., selaku pembimbing II yang selalu dengan senang hati dan sabar dalam memberikan arahan, masukan, saran dukungan, nasihat, dan motivasi dalam pengerjaan skripsi ini. Serta penulis banyak belajar hal-hal baru selama proses pengerjaan skrispsi kepada beliau.
3. Ibu Esty Hartini selaku pemangku kepentingan dan staff perpustakaan pada SMK 1 Muhammadiyah Malang yang telah memberikan waktu dan kesempatan dalam proses wawancara terkait pengembangan sistem informasi.
4. Bapak, Ibu dan Keluarga yang dengan tulus selalu memberi doa, dukungan baik berupa materil maupun non materil, nasihat untuk bagaimana sabar dan tetap berusaha serta motivasi yang tiada henti untuk memberikan semangat kepada penulis dalam pengerjaan skripsi ini hingga selesai.
5. Teman-teman Pendidikan Teknologi Informasi angkatan 2015 dan angkatan setelahnya yang penulis tidak bisa sebutkan satu-persatu yang telah membantu penulis baik secara langsung ataupun tidak langsung.
6. Karyawan dan staf Fakultas Ilmu Komputer Universitas Brawijaya Malang yang membantu penulis dalam proses skripsi ini.
7. Serta semua pihak yang telah terlibat dalam pengerjaan skripsi ini yang tidak bisa disebutkan penulis satu persatu.

Semoga Allah Subhanahu Wa Ta'ala memberikan balasan berlipat ganda kepada pihak-pihak yang telah membantu penulis yang berpartisipasi dalam menyelesaikan skripsi ini. Penulis menyadari bahwa skripsi ini masih jauh dari kata "sempurna". Yang benar datangnya dari Allah Subhanahu Wa Ta'ala. Yang salah

datangnya dari penulis sendiri sebagaimana manusia yang hakikatnya tidak luput dari kesalahan dan lupa. Akhir kata, semoga laporan skripsi ini dapat bermanfaat bagi lingungan sekitar yang menggunakannya. Amin.

Malang, 21 Juli 2019

Penulis
rizkyedyatnaputra@gmail.com

ABSTRAK

Rizky Edyatna Putra, Pengembangan Sistem Informasi Perpustakaan Menggunakan Metode *Extreme Programming* (Studi Pada: SMK 1 Muhammadiyah Malang)

Pembimbing: Satrio Agung Wicaksono, S.Kom, M.Kom dan Issa Arwani, S.Kom, M.Sc

Pada SMK 1 Muhammadiyah Malang terdapat perpustakaan sekolah yang menyediakan koleksi buku dan berbagai majalah untuk digunakan sebagai media pembelajaran oleh siswa maupun guru yang ada di sekolah. Saat ini, perpustakaan masih menggunakan sistem pembukuan untuk melayani aktivitas pelayanan peminjaman buku dan pengembalian buku. Petugas perpustakaan mencatat semua aktivitas pelayanan yang telah dilakukan pada buku folio bergaris, sehingga petugas menghabiskan banyak waktu untuk melakukan aktivitas pelayanan peminjaman buku maupun pengembalian buku, serta belum adanya pengolahan denda secara otomatis. Petugas perpustakaan juga melakukan pencatatan ulang untuk pembuatan laporan transaksi peminjaman buku dan pengembalian buku. Maka penelitian ini dilakukan dengan mengembangkan sistem informasi perpustakaan yang dapat mengatasi masalah keefisienan waktu pada saat melakukan peminjaman buku, pengembalian buku, pengolahan denda serta pembuatan laporan transaksi. Metode penelitian yang digunakan pada penelitian ini adalah metode *Extreme Programming* (XP) dalam pengembangan sistem informasi perpustakaan yang akan dibangun dan menggunakan bahasa pemodelan *Unified Modeling Language* (UML) untuk mempermudah perancangan sistem yang akan dibangun. Implementasi sistem akan menggunakan bahasa pemrograman *C-Sharp* (C#). Berdasarkan analisis masalah yang dilakukan dengan menggunakan metode wawancara, menghasilkan 8 (delapan) fitur dan 10 (sepuluh) kebutuhan fungsional. Hasil pengujian *white-box testing* dan *black-box testing* menyatakan valid, yang artinya sistem dapat berjalan dengan baik sesuai dengan kebutuhan yang diinginkan. Hasil pengujian *User Acceptance Testing* (UAT) menunjukkan bahwa sistem 100% dapat diterima dan layak digunakan oleh pengguna yaitu petugas perpustakaan.

Kata kunci: *sistem informasi, perpustakaan, XP, UML, C#, UAT*

ABSTRACT

Rizky Edyatna Putra, *Library Information System Development Using Extreme Programming Method* (*Study on*: SMK 1 Muhammadiyah Malang)

***Supervisors*: Satrio Agung Wicaksono, S.Kom, M.Kom *and* Issa Arwani, S.Kom, M.Sc**

SMK 1 Muhammadiyah Malang has a school library that provides a collection of books and various magazines to be used as learning media by students and teachers in schools. At present, libraries still use bookkeeping systems to service book lending and book return services. The library officer records all service activities that have been carried out on a striped folio book, so that the officer spends a lot of time doing book lending and returning books, as well as automatic processing of fines. The library officer also records again for making a book loan transaction report and returning the book. So this research is done by developing a library information system that can overcome the problem of time efficiency when borrowing books, returning books, processing fines and making transaction reports. The research method used in this study is the Extreme Programming (XP) method in developing library information systems to be built and using the Unified Modeling Language (UML) modeling language to facilitate the design of the system to be built. Implementation system will use the C-Sharp programming language (C#). Based on the problem analysis carried out using the interview method, it produces 8 (eight) features and 10 (ten) functional requirements. The white-box testing and black-box testing test results are valid, which means the system can run properly according to the desired needs. The test results of User Acceptance Testing (UAT) indicate that the system is 100% acceptable and is suitable for use by users, namely library officers.

***Keywords**: information system, library, XP, UML, C#, UAT*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Ilmu pengetahuan dan teknologi saat ini mengalami perkembangan yang cukup signifikan dari tahun ke tahun, komputer menjadi salah satu contoh perkembangan teknologi. Sebagai salah satu alat bantu yang membantu pekerjaan manusia, komputer mempunyai kelebihan antara lain kecepatan, keakuratan, dan keefisienan dalam mengolah data. Teknologi informasi juga sudah merambat ke dunia kerja dan pendidikan serta bidang lainnya. Demi mencapai suatu tujuan pendidikan, sumber belajar menjadi salah satu pendukung dalam proses belajar mengajar pada dunia pendidikan. Perpustakaan merupakan lembaga sumber belajar penyedia ilmu pengetahuan dan informasi yang memiliki peranan penting terhadap lembaga pendidikan seperti sekolah.

Perkembangan perpustakaan saat ini telah dipergunakan sebagai sebagai sumber ilmu pengetahuan, pelestarian budaya bangsa, penelitian dan menjadi salah satu pusat informasi. Perpustakaan merupakan sebuah media penyedia informasi bagi siswa baik yang bercetak maupun terekam. Perpustakaan juga disebut sebagai jantung dari sekolah, karena di perpustakaan terdapat banyak sumber pengetahuan seperti majalah, koran, dan buku lainnya. Pendayagunaan sumber informasi pada perpustakaan tergantung pada layanan yang terdapat didalamnya. Artinya layanan yang terdapat pada perpustakaan menjadi tolak ukur keberhasilan suatu perpustakaan.

Prinsip perpustakaan memiliki 3 (tiga) kegiatan utama yakni mengumpulkan (*to collect*) seluruh informasi dari berbagai bidang yang terkait, melakukan pemeliharaan koleksi perpustakaan agar tidak cepat rusak dikarenakan pemakaian ataupun usia (*to preserve*), serta menyajikan informasi agar dapat diberdayakan dan dipergunakan dengan baik dan benar (*to make available*) (Sutarno, 2006). Dalam perkembangan ilmu pengetahuan, informasi memiliki peranan penting. Sistem Informasi merupakan sarana untuk melakukan proses yang ada pada perpustakaan, dimana perpustakaan merupakan faktor penting dalam penunjang transformasi antara sumber pengetahuan dengan para pencari pengetahuan.

Kebutuhan sistem informasi didapatkan dari perilaku pengguna dalam mencari informasi, maka hal yang perlu diperhatikan oleh perpustakaan adalah memberikan layanan informasi yang sesuai dengan kebutuhan pengguna agar mendapatkan umpan balik bagi perpustakaan itu sendiri. Berdasarkan hasil observasi dan wawancara peneliti terhadap petugas perpustakaan sekolah Ibu Esty Hartini, proses pencarian dan peminjaman buku di perpustakaan di SMK Muhammadiyah 1 Malang saat ini dilakukan secara manual dengan cara menggunakan tenaga kerja pustakawan untuk mencari buku yang ingin dipinjam oleh pengunjung perpustakaan ataupun buku yang telah didapatkan oleh pengunjung, yang kemudian proses peminjaman dimulai dengan dilakukannya proses pencatatan. Proses tersebut memerlukan banyak waktu sampai buku dapat dipinjam (LAMPIRAN A.1). Begitu juga sebaliknya, pengembalian buku dilakukan

secara manual dengan sistem pencatatan. Ibu Esty Hartini mengharapkan ada sistem informasi perpustakaan untuk memanajemen transaksi peminjaman dan pengembalian buku yang berada di perpustakaan SMK 1 Muhammadiyah Malang ini tanpa ada proses pencatatan. Kurangnya fasilitas internet dan komputer server yang kurang memadai, maka pengembangan sistem informasi perpustakaan ini dibangun menggunakan basis desktop yang menggunakan bahasa pemrograman *C-Sharp* (C#). Sistem informasi perpustakaan diharapkan mampu menunjang kinerja perpustakaan.

Untuk menyelesaikan masalah tersebut usaha yang dilakukan oleh peniliti adalah melakukan pengembangan sistem informasi perpustakaan yang dikhususkan untuk SMK 1 Muhammadiyah Malang. Pada pembangunan sistem informasi perpustakaan ini, peneliti menggunakan metode *Extreme Programming* yang ada pada Agile Methods. Dalam membangun sistem informasi perpustakaan menggunakan metode *Extreme Programing* menurut Pressman (2012) ada 4 tahap dalam meggunakan *Extreme Programming* yaitu *planning*, *design*, *code*, dan *testing*. Dengan menggunakan metode *Extreme Programming* maka melewati beberapa tahap yakni analisis kebutuhan, melakukan perancangan dilanjutkan dengan implementasi sistem, dan yang terakhir dilakukannya pengujian pada sistem yang telah dibangun. Tahapan *Extreme Programming* inilah yang akan menjadi rumusan masalah pada penelitian ini, dan hasil dari tahapan *Extreme Programming* merupakan tujuan dari penelitian ini. Alasan peneliti menggunakan metode *Extreme Programming* dikarenakan *client* sekaligus *user* dari sistem ini kurang mendalami ilmu teknologi sehingga kemungkinan ada perubahan-perubahan fitur ataupun desain pada saat sistem ini dibangun. Menurut (Pratama, 2017) dalam penelitian skripsinya yang berjudul 'Pendekatan Metodologi *Extreme Programming* pada Aplikasi *e-Commerce* Berbasis *M-Commerce*", *Extreme Programming* mempunyai kelebihan yaitu, tahapan dalam proses pengembangan perangkat lunak lebih sederhana sehingga menjadi lebih adaptif dan fleksibel, serta memberikan keselarasan terhadap perubahan-perubahan dari sisi desain dan fitur dengan penanganan secara fleksibel. Oleh karena itu dalam menyusun skripsi ini penulis mengusulkan judul "**PENGEMBANGAN SISTEM INFORMASI PERPUSTAKAAN MENGGUNAKAN METODE *EXTREME PROGRAMMING***".

1.2 Rumusan Masalah

1. Bagaimana hasil analisis kebutuhan sistem informasi perpustakaan menggunakan metode *Extreme Programming* ?
2. Bagaimana hasil perancangan dan implementasi sistem informasi perpustakaan menggunakan metode *Extreme Programming* ?
3. Bagaimana hasil pengujian sistem informasi perpustakaan metode *Extreme Programming* ?

1.3 Tujuan

1. Mengetahui hasil analisis kebutuhan dalam merancang Sistem informasi Perpustakaan menggunakan metode *Extreme Programming*.
2. Mengetahui hasil perancangan dan implementasi Sistem Informasi Perpustakaan menggunakan metode *Extreme Programming*.
3. Mengetahui hasil pengujian Sistem Informasi Perpustakaan menggunakan metode *Extreme Programming*.

1.4 Manfaat

1. Manfaat bagi perpustakaan :

   Manfaat pengembangan sistem informasi perpustakaan bagi perpustakaan adalah meningkatkan efesiensi dan efektifitas pengelolaan data buku.
2. Manfaat bagi pustakawan :

   Manfaat pengembangan sistem informasi perpustakaan bagi petugas perpustakaan adalah memudahkan proses kerja mereka dengan adanya sistem informasi perpustakaan ini.
3. Manfaat bagi pengunjung perpustakaan :

   Manfaat pengembangan sistem informasi perpustakan bagi pengunjung perpustakaan adalah lebih mudah dan efisien dalam melakukan peminjaman buku, serta pengembalian buku.

1.5 Batasan Masalah

Adapun batasan masalah yang digunakan agar pembahasan tidak melebar terlalu jauh adalah:

1. Analisis kebutuhan hanya sebatas *user stories,* dan perancangan sistem menggunakan OOP (*Object Oriented Programming*).
2. Proses pembuatan rancangan sistem informasi, rancangan *database*, pembuatan *database*, dan proses pembuatan sistem informasi secara keseluruhan.
3. Proses perancangan sistem yang mengimplementasikan bahas pemodelan *Unified Modeling Language*.
4. Proses pembuatan program terdiri dari form input, prosedur eksekusi, dan tampilan data dari proses eksekusi.

1.6 Sistematika Pembahasan

Penelitian skripsi ini, disusun secara sistematis dalam enam babı mulai dari bab pendahuluan hingga bab penutup. Adapun rancangan sistematika pembahasan skripsi disusun sebagai berikut :

1. BAB 1 : PENDAHULUAN

   Merupakan gambaran umum dari penelitian yang dilakukan, seperti : latar belakang, rumusan masalah, tujuan, manfaat penelitian, batasan masalah, dan sistematika pembahasan.
2. BAB II : LANDASAN KEPUSTAKAAN

   Pada bab ini, berisi tentang kumpulan teori yang berkaitan dengan penelitian yang dilakukan.
3. BAB III : METODOLOGI

   Pada bab ini, berisi tentang jenis penelitian, latar penelitian, data dan sumber penelitian, teknik pengumpulan data, teknik analisis data, dan teknik keabsahan data.
4. BAB IV : HASIL DAN PEMBAHASAN

   Pada bab ini, berisi tentang hasil hasil analisis kebutuhan, hasil perancangan sistem, hasil implementasi sistem, serta hasil pengujian sistem agar mendapatkan hasil yang sesuai dengan kebutuhan.
5. BAB V: PENUTUP

   Pada bab ini, berisi tentang kesimpulan, implikasi, dan saran dari pembuatan serta pengujian dari sistem informasi yang telah selesai dibangun.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Tinjauan Pustaka

Penelitian terdahulu merupakan salah satu sumber pustaka peneliti dalam melakukan sebuah penelitian sehingga peneliti dapat mengkaji penelitian terdahulu yang telah dilakukan dengan cara memperkaya teori. Berikut merupakan penelitian terdahulu berupa jurnal dan skripsi yang terkait dengan penilitian yang dilakukan peniliti.

Penelitian terdahulu pertama adalah yang dilakukan oleh (Nama dan Arnoldi, 2016) yang berjudul ”Rancang Bangun Aplikasi Game Edukasi Pembelajaran Aksara Lampung Ajo dan Atu — Belajar Aksara Lampung Berbasis Android Dengan Sistem *Multi-Ending* Menggunakan *Engine Ren'pf”*. Penelitian yang dilakukan menggunakan model SDLC (*System Development Life Cycle*) yaitu *Extreme Programming* (XP) dalam peracangan serta pembangunan aplikasi. Penelitian terdahulu ini menghasilkan sebuah game edukasi yang memiliki sistem multi-ending dan sistem point yang dapat membantu pengguna lebih tertarik untuk terus mencoba mencari ending yang lain sehingga pengguna dapat lebih mudah belajar karena mengulang permainan berkali-kali.

Penelitian terdahulu yang kedua yaitu mecangkup pemanfaatan Sistem Informasi dalam menyelesaikan masalah manajemen perpustakaan yang ada di ruang baca Fakultas Ilmu Komputer Universitas Brawijaya. Penilitian yang dilakukan oleh (Shodiqin, 2016), dalam skripsi yang berjudul “Rancang Bangun Sistem Informasi Manajemen Perpustakaan pada Ruang Baca Fakultas Ilmu Komputer Universitas Brawijaya”, yang melakukan pengembangan sistem informasi perpustakaan ruang baca Fakultas Ilmu Komputer yang sudah ada. Dalam menggali kebutuhan pengumpulan data yang diperlukan peneliti guna mendukung penelitian pengembangan sistem informasi perpustakaan pada SMK 1 Muhammadiyah Malang, peneliti menggunakan metode wawancara dan observasi.

2.2 Perpustakaan

Perpustakaan merupakan bangunan fisik sebagai tempat pengumpulan buku yang disusun menurut sistem tertentu sesuai keperluan pemakai (Lasa, 2007). Perpustakaan harus mengumpulkan koleksi berupa karya tulis, karya cetak maupun karya rekam dengan menggunakan sistem yang baku untuk memenuhi kebutuhan informasi, pendidikan, penelitian, pelestarian dan rekreasi terhadap pemustaka maka. Dengan majunya teknologi, sudah banyak perpustakaan di zaman sekarang yang dimanajemenkan oleh sebuah sistem computer yaitu sistem informasi perpustakaan.

2.3 Sistem Informasi

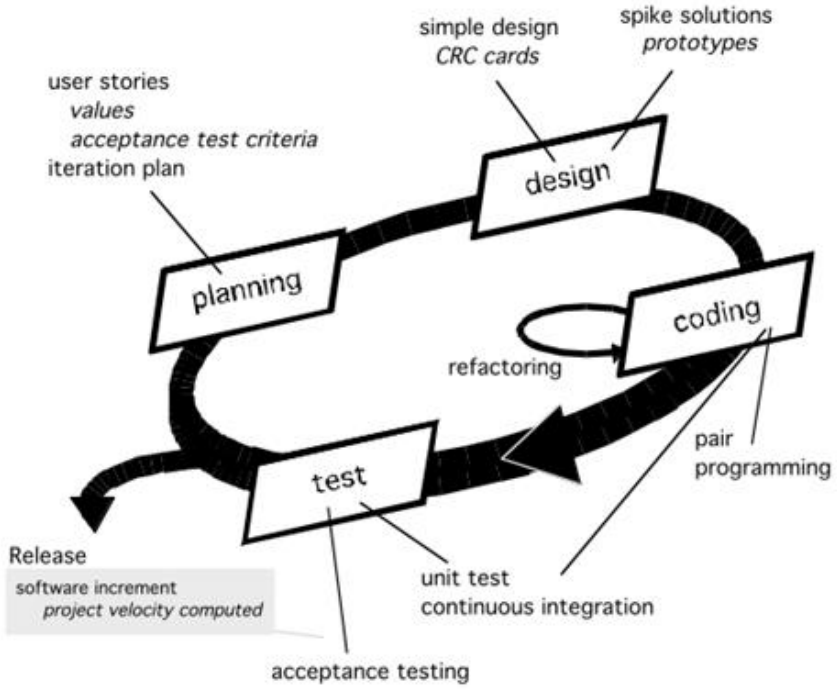
Sistem informasi merupakan suatu sistem yang digunakan oleh organisasi untuk dapat membantu mengelola kebutuhan transaksi harian, mendukung kegiatan suatu organisasi, , menyediakan laporan bagi pihak luar tertentu, dan membantu kegiatan strategis organisasi yang juga bersifat manajerial (Jogiyanto, 2002).

Metode pendekatan dapat digunakan sebagai acuan untuk mengembangkan sebuah sistem informasi. Metode pendekatan *Extreme Programming* (XP) dari salah satu model SDLC (*System Development Live Cycle*) digunakan peneliti untuk melakukan pengembangan sistem informasi pada penelitian ini.

2.4 *Extreme Programming*

Kent Black merupakan seorang pakar perangkat lunak yang menemukan metode *Extreme Programming* (XP) pertama kali. Kent black membuat proyek *Chrysler Comprerhensive Compensation* (C3) dengan Chrysler yang terancam akan gagal sebelum ia di kontrak. Pada saat Kent Black dan Ron Jeffries menggunakan berbagai metode untuk menyelesaikan proyek sesuai target. Kumpulan metode yang digunakan dalam proses pengembangan perangkat lunak disebut dengan metode *Extreme Programming* (XP). Dengan menggunakan metode *Extreme Programming* (XP), Kent Black lebih fleksibel, adaptif, dan efisien dalam melakukan perubahan pada pembangunan proyek (Widhiartha, 2008).

Menurut Rahmi, Sari, dan Suhatman (2016) dengan arti lain, *Extreme Programming* (XP) adalah metode pengembangan perangkat lunak yang dapat merubah atau menambah kebutuhan sistem maupun alur kerja sistem selama pembangunan perangkat lunak berlangsung. Menurut Kumar, Singh, dan Dwivedi (2015) dari hasil penilitian yang dilakukan, mereka menarik kesimpulan bahwa hasil penelitian yang telah dilakukan menunjukkan bahwa pendekatan *Extreme Programming* untuk pengembangan perangkat lunak jauh lebih baik, produktif dan efektif dibandingkan dengan metode pengembangan perangkat lunak tradisional yang biasanya memberikan *prototype* aplikasi di akhir pengembangan. Adapun tahapan yang dilakukan dalam menggunakan metode pengembangan sistem informasi *Extreme Programming* menurut Pressman R.S. (2012) yaitu, perencanaan (*Planning*), perancangan (*Design*), pengkodean (*Coding*), dan pengujian (*Testing*) yang ada pada Gambar 2.1.



Gambar 2.1 *Extreme Programming Model*
Sumber: Pressman (2012)

2.4.1 Perencanaan (*Planning*)

Pada tahap perencaan ini, peneliti menganalisa kebutuhan-kebutuhan paling dasar yang perlukan sistem. Menurut (Rahmi, Sari, Suhatman, 2016) Pada tahap perencanaan ini, akan dihasilkan kebutuhan bisnis dan kebutuhan sistem dari *User Stories* yang telah didapatkan. *User Stories* yang dimaksud adalah semua kebutuhan-kebutuhan user yang menggunakan sistem ini. *User Stories* biasa dilakukan dengan cara mewawancarai *user* dari sistem yang ingin *user* gunakan. Pada tahap ini juga peneliti juga akan memperkirakan kebutuhan sistem serta kebutuhan bisnis dari sistem. Kebutuhan sistem diperoleh dari analisis yang dilakukan terhadap fungsionalitas sistem. *User Stories* bisa didapatkan dengan cara melakukan wawancara kepada yang bersangkutan. Setelah melakukan wawancara, peneliti melakukan analisis kebutuhan sistem berdasarkan hasil wawancara. Pada tahap perencanaan ini juga dilakukan pemodelan menggunakan bahasa pemodelan *Unified Modeling Language* (UML) hanya sebatas *use case diagram* saja.

User	Kebutuhan Sistem
Admin	Admin memiliki hak akses dalam melakukan pengelolaan data member dan buku. Seperti, mengelola data member, mengelola data buku, menambahkan / menghapus kategori buku, mengecek pesanan buku, melakukan pengecekan data konfirmasi pembelian buku oleh member
Member	Member memiliki hak ases dalam melakukan verifikasi data untuk menjadi member dengan cara membuat akun untuk login, melakukan login jika member ingin melakukan transaksi pembelian buku, menambahkan pembelian ke keranjang belanja, mengubah data pembelian, melakukan konfirmasi pembelian buku.
Calon Member	Calon member hanya dapat melihat deskripsi buku dan infromasi yang tertera pada aplikasi. Jika calon member ingin melakukan transaksi pemesanan produk, harus melakukan login terlebih dahulu dengan menginputkan *email* dan *password*. Dan melakukan registrasi sebagai member sebelum melakukan *login*.

Gambar 2.2 *User Stories*

Sumber: Pratama (2017)

Menurut Kubde dan Sabde (2014) melakukan aktivitas verifikasi dilakukan di beberapa tahap SDLC, salah satu tahap yang wajib dilakukannya verifikasi yaitu analisis kebutuhan yang berada pada tahap perencanaan (*Planning*) pada *Extreme Programming.* Tidak hanya pada tahap perencanaan, pada tahap perancangan (*Design*) juga perlu dilakukannya verifikasi agar memastikan sistem yang sedang dibangun sesuai dengan spesifikasi design dan persyaratan yang diberikan oleh *stakeholder*.

2.4.1.1 Wawancara

Wawancara adalah situasi dimana pewawancara dan responden berhadap-hadapan untuk mendapatkan informasi tentang responden maksimum efisiensi dan minimum bias (Nul, 2013). Menurut Nul (2013), mendapatkan kepercayaan dari responden adalah hal yang penting dalam melakukan proses wawancara. Pewawancara harus berupaya mendapatkan kepercayaan dari responden agar data yang didapatkan berkualitas tinggi. Sikap transparansi adalah salah satu cara mendapatkan kepercayaan responden serta melakukan penyediaan informasi diantaranya, pengenalan diri, tujuan melakukan wawancara, berapa lama waktu yang dibutuhkan, untuk apa data tersebut

digunakan, apakah informasi tersebut bersifat anonim atau tidak, dan dimana hasil penelitian tersebut akan disebarkan.

2.4.2 Perancangan (*Design*)

Tahap perancangan sebenarnya dilakukan secara terus-menerus selama proses pengembangan sistem informasi ini berlangsung. Tahap perancangan pada model proses *Extreme Programming* merupakan tuntunan untuk mengembangkan sistem yang didasari *User Story* yang telah didapat pada tahap sebelumnya yaitu tahap perencanaan. Pada tahap peracangan ini juga dilakukan pemodelan untuk membangun sistem menggunakan bahasa pemodelan *Unified Modeling Language* (UML) dan pemodelan data menggunakan MySql serta perancangn antar muka sistem.

2.4.2.1 *Unified Modeling Language* (UML)

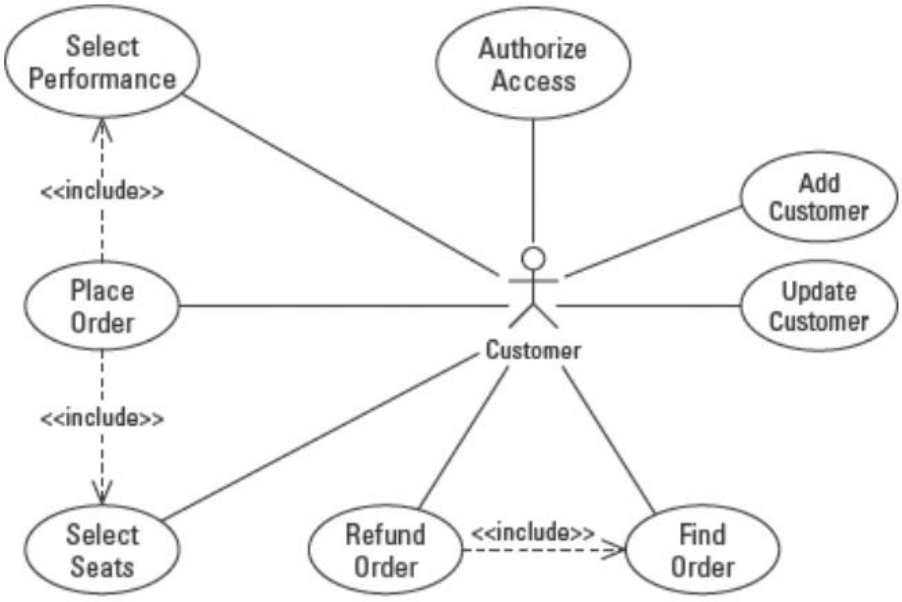
Secara umum, *Unified Modeling Language* (UML) merupakan upaya untuk memberikan pendekatan tunggal untuk *Object Oriented Programming* (OOP) yang berlaku di semua domain perangkat lunak dengan tujuan utama untuk mewakili arsitektur perangkat lunak. Pemodelan analisis *Unified Modeling Language* (UML) berfokus pada model pengguna dan tampilan model struktural sistem. Spesifikasi detail struktur data atribut dan desain prosedural dari seluruh operasi dibuat pada saat pendesainan objek. Desain sistem dan objek dalam *Unified Modeling Language* (UML) diperluas untuk mempertimbangkan desain antarmuka pengguna, manajemen data , dan manajemen tugas untuk subsistem yang telah ditentukan. Artifak dari sistem perangkat lunak divisualisasikan, dikonstuksikan, dispesifikasikan, dan didokumentasikan menggunakan *Unified Modeling Language* (UML) (Pressman, 2012). *Unified Modeling Language* (UML) dapat digambarkan untuk dijadikan gambaran dari sebuah sistem yang memiliki standar diagram dengan berbagai sudut pandang. Diagram tersebut diantaranya *use case diagram*, *use case scenario*, *sequence diagram,* dan *class diagram*.

(a) *Use Case Diagram*

Use case diagram melibatkan beberapa pengguna untuk mengidentifikasi interaksi. Penambahan informasi terhadap sistem, menggunakan pendeskripsian hasil identifikasi interaksi. Penambahan informasi yang dimaksud, dapat berupa model grafik seperti squence diagram ataupun dalam bentuk pendeskripsian text (Sommerville, 2011). Contoh *use case diagram* ada pada Gambar 2.2 dan simbol - simbol *use case diagram* tersedia pada Tabel 2.1.

Tabel 2.1 Simbol-simbol *Use Case Diagram*

GAMBAR	KETERANGAN
	*Use case* menggambarkan tindakan yang dilakukan oleh *actor*. *Use case* ditulis menggunakan kata kerja di awal.
	*Actor* adalah pengguna yang berinteraksi dengan sistem, dan dijadikan peranan yang dapat berinteraksi dengan sistem.
	Relathionship menggambarkan hubungan antara *actor* dengan *use case* atau *use case* dengan *actor* yang berada di dalam sistem.
<<extend>>	*Extend Relationship* adalah hubungan antara *use case* satu dengan *use case* tambahan lainnya yang bersifat optional.
<<include>>	*Include Relationship* merupakan hubungan terarah antara dua use case yang digunakan untuk menunjukkan bahwa perilaku use case yang dimasukkan (tambahan) dimasukkan ke dalam perilaku *use case* yang termasuk (basis).



Gambar 2.3 Contoh *Use Case Diagram*
Sumber: Pender (2003)

Pada Gambar 2.2 yaitu contoh gambar *use case diagram* yang menunjukkan ekspektasi pelanggan teater untuk menggunakan sistem penjualan. Seseorang pelanggan yang menuju bagian penjualan di teater,

pelanggan mempunyai beberapa fitur yang memungkinkan untuk melakukan pemesanan dengan memilih pertunjukan, memilih tempat duduk di pertunjukan, dan membayar pesanan.

(b) *Use Case Scenario*

Use case scenario merupakan sekumpulan skenario interaksi yang dijelaskan secara tekstual, pendeskripsian skenario diurutkan berdasarkan setiap langkah atau aksi yang dilakukan pengguna (actor) pada saat melakukan interaksi pada sistem, baik yang berhasil maupun gagal. U*se case scenario* dijelaskan secara tekstual dalam beberapa format tergantung kebutuhannya yakni singkat (*brief*), informal (*casual*), atau lengkap (*fully dressed*) (Kurniawan, 2018). Contoh isi table *use case senario* tersedia pada Gambar 2.3.

Use Case Name
<Brief description. Usually a paragraph or less.>

Actors
<A list of the Actors who communicate with this Use Case>

Priority
<How important is this Use Case to the project?>

Status
<What point are we in developing this Use Case?>

Pre-Conditions
<A list of conditions that must be true before the Use Case starts>

Post-Conditions
<A list of conditions that must be true when the Use Case ends, no matter which Scenario is executed.>

Extension Points
<If the Use Case has extension points, list them here.>

"Used" Use Cases
<If the Use Case uses other Use Cases, list them here.>

Gambar 2.4 Contoh *Use Case Scenario*
Sumber: Schneider dan Winters (2001)

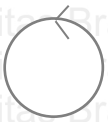
Pada Gambar 2.3 yaitu contoh *use case scenario,* yang mana isi dari tabel *use case scenario* adalah nama *use case,* aktor yang menggunakan *use case,* prioritas *use case* dari *project*, status *use case*, *pre-conditions* yang berisi kondisi-kondisi benar sebelum u*se case* dijalankan, *post-conditions* yang berisi kondisi-kondisi benar ketika *use case* telah dijalankan, e*xtension point* yang berisi list dari point tambahan *use case*, *used use cases* yang berisi list *use case* lain yang digunakan dalam *use case* ini.

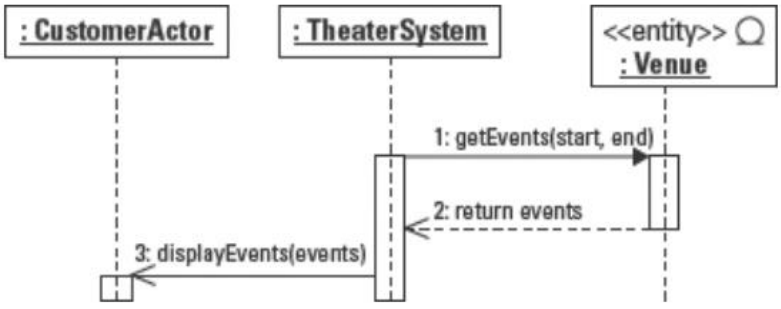
(c) *Sequence Diagram*

Sequence diagram digunakan sebagai pemodelan interaksi antar sistem dan komponen-komponennya. Komponen eksternal dari sistem juga dapat dimasukkan kedalam *sequence diagram*. Menurut Sommerville (2011) untuk memodelkan interaksi diantara objek dan objek itu sendiri

serta objek dengan aktor pada sebuah sistem, digunakan *squence diagram* yang merupakan diagram utama dari Unified *Modeling Language* (UML). Contoh *sequence diagram* ada pada Gambar 2.3 dan simbol-simbol *sequence* d*iagram* tersedia pada Tabel 2.2.

Tabel 2.2 Simbol-simbol *Sequence Diagram*

GAMBAR	KETERANGAN
	*Actor*, menggambarkan orang yang berintaksi dengan sistem.
	*Boundary*, digunakan untuk menggambarkan sebuah form.
	*Control*, menangani tugas utama dari alur sistem.
	*Entity* merupakan gambaran informasi yang harus disimpulkan oleh sistem.
	*Lifeline* merupakan pengindikasian sebuah keberadaan objek pada waktu tertentu.
	Message, hubungan antar objek satu dengan objek lainnya yang menunjukan suatu kejadian.



Gambar 2.5 Contoh *Sequence Diagram*
Sumber: Pender (2003)

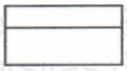
Pada Gambar 2.4 memodelkan tiga objek yaitu actor pelanggan, sistem teater, dan venue teater. Sistem teater meminta tempat untuk serangkaian acara yang terjadi dalam rentang tanggal yang ditentukan. Venue mengembalikan (*return*) data set acara. Sistem teater kemudian

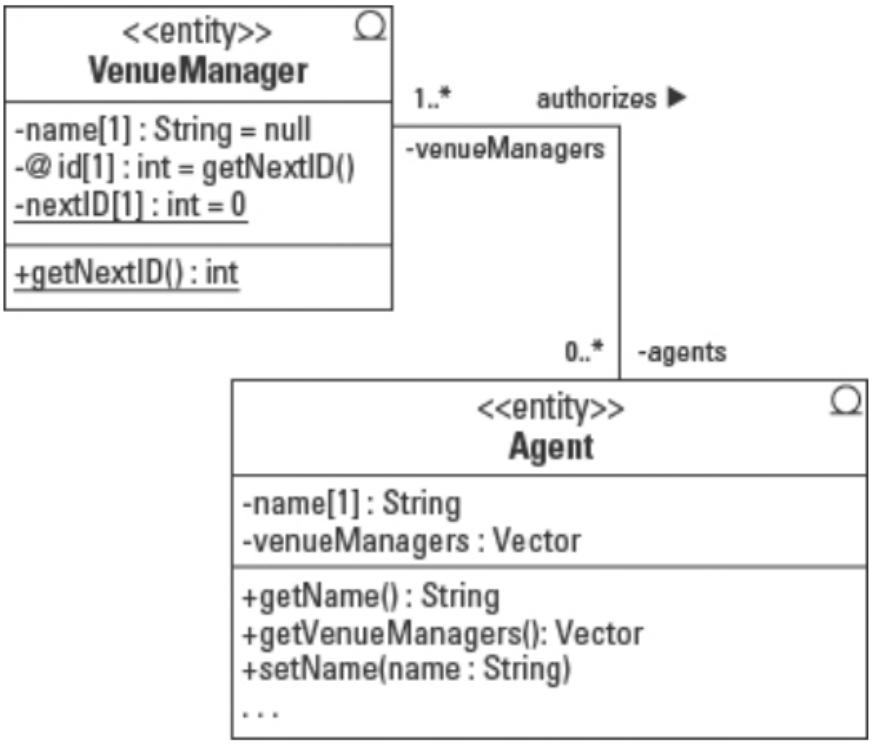
mengirimkan acara ke aktor pelanggan (antarmuka pengguna) untuk ditampilkan.

(d) *Class Diagram*

Menurut Munawar (2005), *class diagram* merupakan sekumpulan dari objek yang sejenis. Dalam sebuah objek memiliki keadaan (*state*) dan perilaku (*behavior*). *State* adalah kondisi dimana objek tersebut dinyatakan dalam *attribute* atau *properties*. Sedangkan perilaku (*behavior*) digunakan untuk mendefinisikan bagaimana sebuah objek dapat memberikan reaksi dan juga melakukan aksi. Contoh *class diagram* ada pada Gambar 2.4 dan simbol - simbol *class diagram* tersedia pada Tabel 2.3.

Tabel 2.3 Simbol-simbol *Class Diagram*

GAMBAR	NAMA	KETERANGAN
	Generalization	Hubungan antara *class* anak dengan *class* induknya, dimana *class* anak memiliki atribut dan proses yang sama dengan *class* induknya.
	*Nary Association*	Assosiasi yang dapat berhubungan dengan 3 (tiga) kelas atau lebih.
	*Class*	*Class* merupakan suatu objek yang memiliki ataupun tidak memiliki atribut dan operasi.
	Realization	Merupakan hubungan antar kelas yang dapat menambah atau menghapus objek dari kelas itu sendiri.
	Dependency	Hubungan di antara dua kelas jika perubahan definisi dari satu kelas dapat menyebabkan perubahan ke kelas yang lain (tetapi tidak sebaliknya).
	Association	*Association* merupakan hubungan atar kelas yang ada pada *class diagram.*



Gambar 2.6 Contoh *Class Diagram*
Sumber: Pender (2003)

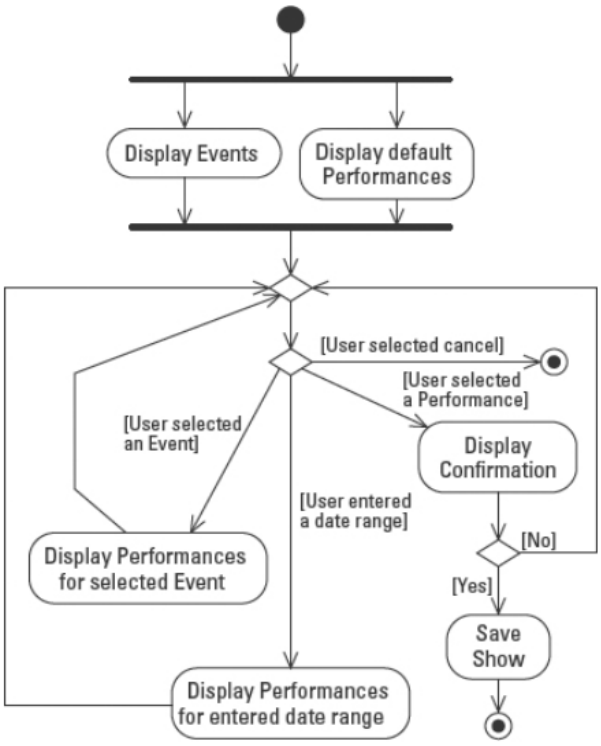
Pada Gambar 2.5 memodelkan dua *class*, yaitu *class* VenueManager dan *class* Agent dengan menggunakan relasi otoritas. Dalam sistem teater, manajer venue memberi wewenang kepada agen untuk menjual tiket. Seiring waktu agen dapat diotorisasi oleh sejumlah pengelola venue yang berbeda. Model mendefinisikan berapa jumlah objek yang dapat menetapkan aturan, mendefinisikan, serta mampu berpartisipasi dalam hubungan objek.

(e) *Avtivity Diagram*

Menurut Munawar (2005), *activity diagram* merupakan sebuah teknik pendeskripsian proses bisnis, logika prosedural, dan aliran kerja dalam beberapa kasus. *Activity diagram* memiliki peranan seperti *flowchart*. Perbedaan *flowchart* dengan *activity diagram* terletak pada perilaku pararel yang mendukung, sedangkan flowchart tidak mendukung perilaku pararel. Contoh *Activity diagram* ada pada Gambar 2.5 dan simbol-simbol *activity diagram* tersedia pada Tabel 2.4.

Tabel 2.4 Simbol-simbol *Activity Diagram*

GAMBAR	KETERANGAN
	Mulai, mempresentasikan kegiatan saat awal proses dimulai.
	Arus kegiatan, mempresentasikan jalurnya proses
	Proses, mempresentasikan suatu kegiatan, aktivitas ataupun tindakan.
	*Decision*, mempresentasikan tentang suatu tindakan yang memiliki kondisi.
	*Fork*, mempresentasikan perlakuan atau aktivitas yang dilakukan secara pararel.
	*Final activity*, mempresentasikan bahwa kegiatan proses telah selesai dilakukan.



Gambar 2.7 Contoh *Activity Diagram*
Sumber: Pender (2003)

Pada Gambar 2.6 memodelkan Select Performance dari Gambar 2.2. Panah alur menelusuri dari proses awal hingga akhir melalui decisions dan perulangan, sambilan mengidentifikasi setiap langkah logika dalam proses.

2.4.3 Pengkodean (*Coding*)

Tahapan pengkodean adalah tahapan pengimplementasian sistem yang akan dilakukan setalah peneliti melakukan tahap perancangan, sehingga akan mendapatkan sistem yang sesuai dengan kebutuhan-kebutuhan yang ada tanpa melewati batas penelitian. Pada tahap ini, peneliti akan melakukan proses implementasi menggunakan bahasa pemrograman *C-Sharp* (C#) pada aplikasi Microsoft Visual Studio dan MySql sebagai *Database Management System* (DBMS) dalam mengimplementasikan basis data.

2.4.3.1 Microsoft Visual Studio

Microsoft Visual Studio adalah aplikasi pemrograman yang menggunakan teknologi *Net framework*. Teknologi *Net framework* adalah salah satu komponen Windows yang terintegrasi, serta mendukung pembuatan, penggunaan aplikasi, dan halaman web. Microsoft Visual Studio dapat digunakan untuk pengembangan aplikasi dalam *managed code* (berbentuk Microsoft Intermediate language di atas *NET Framework*) ataupun *native code* (dalam bentuk bahasa mesin yang berjalan di atas Windows). Selain itu, Microsoft Visual Studio juga dapat digunakan untuk mengembangkan aplikasi *Windows Mobile* (yang berjalan di atas *NET Compact Framework*) dan aplikasi *Silverlight* (Komputer, 2008).

2.4.3.2 *C-Sharp* (C#)

C-Sharp (C#) merupakan bahasa pemrograman sederhana yang digunakan untuk tujuan umum. Bahasa pemrograman ini dapat digunakan untuk berbagai fungsi misalnya, membangun aplikasi *desktop* ataupun *mobile*, pemrograman *game*, dan pemrograman *server-side* pada website. Selain itu, *C-Sharp* (C#) juga dapat menjadi bahasa yang berorientasi objek, sehingga *C-Sharp* (C#) mengusung konsep objek seperti, *class*, *inheritance*, *encapsulation*, dan *polymorphism* (Whitaker, 2015).

2.4.3.3 MysQl

Database merupakan sekumpulan data yang saling berhubungan secara login dan memberi penjalasan terhadap data tersebut. Data yang telah berhubungan secara logis akan didesign agar dapat memenuhi kebutuhan permintaan data dari sebuah organisasi yang mana seluruh data tersebut diintegrasikan tanpa adanya duplikasi data. *Database* juga merupakan sekumpulan elemen data terintegrasi yang secara logika saling berhubungan (Indrajani, 2015).

2.4.4 Pengujian (*Testing*)

Pengujian software digunakan terhadap perangkat lunak yang telah dibangun atau masih dalam proses pembangunan untuk memastikan perangkat

lunak tersebut dapat berjalan sesuai dengan fungsionalitas yang diharapkan atau tidak (Mustaqbal, Firdaus, Rahmadi, 2015). Sebelum user menggunakan sistem, harus melakukan proses pengujian oleh programmer dan sistem analis untuk menguji fitur-fitur dan fungsionalitas sistem yang akan dikembangkan. Terdapat beberapa metode yang dapat digunakan dalam melakukan pengujian pada suatu sistem yang telah dikembangkan. Pada penelitian ini, menggunakan pengujian *white-box testing* dan *black-box testing* serta *User Acceptance Testing* (UAT).

2.4.4.1 Pengujian *White-Box*

White-box testing merupakan pengujian terhadap suatu perangkat lunak dengan mengikuti panduan agar dapat meneliti dan melakukan anilis kode program untuk mengetahui kesalahan dari perangkat lunak tersebut. Apabila operasi yang telah diuji menghasilkan output yang tidak sesuai, maka akan dilakukan pengecekan kode program tersebut hingga sesuai dengan yang diharapkan. Pengujian *white-box testing* merupakan pengujian yang meninjau berdasarkan pengecekan terhadap detail perancangan dengan menggunakan struktur kontrol dari desain program secara prosedural untuk membagi pengujian ke dalam beberapa kasus pengujian (Mustaqbal, Firdaus, Rahmadi, 2015).

Menurut Mustaqbal, Firdaus, dan Rahmadi (2015) pengujian *white-box testing* akan diujikan menggunakan beberapa tahap untuk menyelesaikan kasus, diantaranya:

1. Pengujian terhadap alur program yang menggunakan logika.
2. Pengujian terhadap seluruh perulangan yang ada, sesuai batasan-batasannya.
3. Pengujian terhadap struktur data yang bersifat internal dan tervaliditasi.

Kelebihan *white-box testing* diantaranya (Mustaqbal, Firdaus, Rahmadi, 2015) :

1. Kesalahan Logika

   Menggunakan sintax ‘if’ dan sintax pengulangan. Langkah selanjutnya, metode *white-box testing* akan mencari serta mendeteksi semua kondisi yang tidak sesuai, dan mencari akhir dari perulangan tersebut.

2. Ketidaksesuaian Asumsi

   Menampilkan serta memonitor beberapa asumsi yang diyakini tidak sesuai atau yang akan diwujudkan. Selanjutnya, dilakukan penganalisisan kembali kemudian diperbaiki.

3. Kesalahan Pengetikan

   Mendeteksi serta mencari bahasa pemrograman yang bersifat case sensitif.

Perhitungan *cyclometic complexity* dapat dilakukan menggunakan rumus berikut:

V(G) = Jumlah edge – jumlah node + 2

V(G) = Jumlah predicated node + 1

V(G) = Jumlah region

2.4.4.2 Pengujian *Black-Box*

Pengujian *black-box testing* ialah sebuah pengujian terhadap kebutuhan fungsional dari sebuah aplikasi yang melakukan persyaratan fungsional sebuah program untuk mendapatkan kondisi input yang sepenuhnya menjalankan persyaratan tersebut (Pressman, 2012).

Black-box testing merupakan pengujian yang mengerucut pada spesifikasi fungsional dari sebuah aplikasi. Penguji dapat melakukan pengujian pada spesifikasi fungsional program, serta melakukan pendefinisian pada kumpulan kondisi input (Mustaqbal, Firdaus, Rahmadi, 2015).

Black-box testing pada umumnya mencari kesalahan-kesalahan yang biasa ada didalam sistem, kesalahan tersebut antara lain:

1. Fungsi yang salah maupun yang tidak teridentifikasi.
2. Kesalahan antarmuka (*interface errors*).
3. Kesalahan pada struktur data dan akses basis data.
4. Kesalahan performansi (*performance errors*).
5. Kesalahan inisialisasi dan terminasi.

Menurut Mustaqbal, Firdaus dan Rahmadi (2015) pengujian *black-box testing* ini di desain untuk menjawab pertanyaan seputar sistem yang dibangun. Pertanyaan tersebut antara lain:

1. Agar dapat dinyatakan valid, bagaimana cara fungsi-fungsi tersebut diuji?
2. Untuk menemukan *test case* yang baik, input seperti apa yang dibutuhkan?
3. Bagaimana cara agar sistem dapat sensitif terhadap input-input tertentu?
4. Bagaimana jika beberapa data dapat diisolasi?
5. Seberapa banyak rata-rata data dan jumlah data yang mampu diatasi sistem?
6. Efek apa yang dapat membuat kombinasi data ditangani spesifik pada operasi sistem?

2.4.4.3 Pengujian *User Acceptance Testing* (UAT)

Menurut Sualim, Yassin, dan Mohamad (2016), *User Acceptance Testing* (UAT) merupakan proses verifikasi solusi dari permasalahan yang berfungsi untuk pengguna sistem. *User Acceptance Testing* (UAT) berfungsi untuk memastikan bahwa solusi yang diberikan oleh peneliti memecahkan permasalahan pengguna. Tujuan *User Acceptance Testing* (UAT) ialah untuk mengumpulkan masukan dari

pengguna sistem setelah memiliki pengalaman menggunakan sistem menyelesaikan permasalahan yang diberikan pengguna. Dalam jurnalnya kriteria yang digunakan dalam *User Acceptance Testing* (UAT) yaitu *Usability Criteria* yang terdiri dari, efisiensi, kepuasan, efektifitas dan *error*. Menurut Sualim, Yassin, dan Mohamad (2016) menjelaskan bahwa hasil dari pengujian ini adalah mendeskripsikan hasil penilaian pengguna dalam bentuk paragraf.

Tabel 2.5 Kriteria *User Acceptance Testing* (UAT)

Kriteria	Deskripsi
Efisiensi	• Pengguna mencapai tujuan tertentu dan sumber daya yang dikeluarkan untuk mencapainya. • Waktu penyelesaian tugas dan waktu untuk mempelajari sistem. • Pengguna dapat menemukan informasi lebih cepat.
Efektivitas	• Pengguna dapat mencapai tujuan tertentu. • Termasuk kualitas solusi dan tingkat kesalahan. • Tercapainya tujuan untuk keseluruhan sistem.
Kepuasan	• Kenyamanan pengguna dan sikap positif terhadap penggunaan aplikasi. • Kepuasan pengguna dapat diukur dengan skala penilaian sikap. • Tercapainya tujuan untuk keseluruhan sistem. • Pengguna menganggap aplikasi keseluruhan dapat diterima.
Error	• Tingkat kesalahan yang didapat selama waktu menjalakan sistem.

Pada Tabel 2.5 menjelaskan tentang kriteria yang ada pada *User Acceptance Testing* (UAT) berserta deskripsi dari setiap kriterianya. Untuk mengetahui penilaian dari pengguna, pengembang harus mempunyai pertanyaan-pertanyaan yang dibuat untuk mewakili setiap kriteria yang ada dan *test case* atau kasus uji yang diujikan untuk mendapatkan penilaian dari pengguna.

Tabel 2.6 Contoh *Test Case User Acceptance Testing* (UAT)

Step No. 1 Deskripsi : Membuka URL elearning.utm.my?14152. Hasil yang diharapkan : Halaman *login* ditampilkan. a) "Username" text field b) "Password" text field c) "Login" button Step No. 2 Deskripsi : Setelah menampilkan halaman login. Mengisi form login dengan valid a) Mengisi kolom "Username" dengan "wendy" b) Mengisi kolom "Password" dengan "1234wendy" Hasil yang diharapkan : a) Memverifikasi Username b) Memverifikasi Password

Pada Tabel 2.6 menjelaskan contoh *test case* atau kasus uji yang diujikan ke pengguna untuk mendapatkan penilaian dari pengguna. Pada tabel 2.6 mencontohkan *test case* untuk melakukan fungsi *login* pada sistem.

Tabel 2.7 Contoh Pertanyaan *User Acceptance Testing* (UAT)

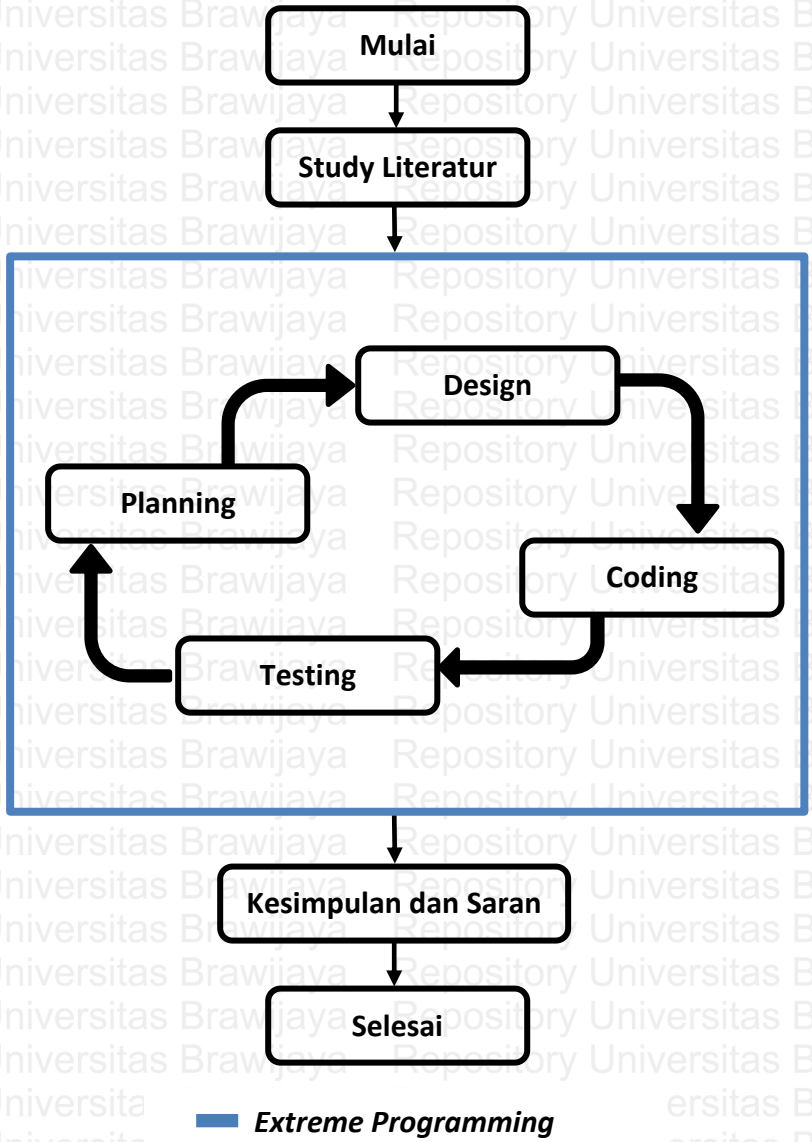
Kriteria	ID	Pertanyaan	Kemungkinan Jawaban
Efisiensi	Q8	Seberapa baik tingkat akurasi dan kelengkapan aplikasi berbasis web dapat memberikan informasi dan fungsionalitas sesuai dengan: 1. Kualitas dari solusi untuk kesalahan-kesalahan 2. Penyeleseaian kesalahan	• Efisien • Sedang • Tidak semuanya
Efektifitas	Q9	Seberapa efektif aplikasi web itu berdasarkan: 1. Partisi informasi yang logis 2. Teks tautan yang tepat 3. Pemuatan style sheet yang tepat 4. Pesan dan manajemen data yang dinamis 5. Konsistensi dan tata letak	• Efektif • Sedang • Tidak semuanya

Pada Tabel 2.7 mencontohkan pertanyaan yang akan diberikan ke pengguna setelah pengguna melakukan *test case* yang diberikan pengembang. Pertanyaan yang ada mewakili setiap kriteria dari yang ada pada *User Acceptance Testing* (UAT), beserta kemungkinan jawaban yang akan pengguna berikan.

BAB 3 METODOLOGI

3.1 Langkah-Langkah Penelitian

Tipe penelitian ini merupakan implementatif-pengembangan (*development*), peneliti melakukan pengembangan sistem informasi perpustakaan pada SMK 1 Muhammadiyah Malang. Tahapan penetitian dalam pengembangan Sistem Informasi Perpustakaan pada SMK 1 Muhammdiyah Malang yang ada pada Gambar 3.1.



Gambar 3.1 Langkah-langkah Penelitian

3.1.1 Study Literatur

Pada tahap study literatur peneliti mengumpulkan teori-teori yang berhubungan dengan penelitian sebagai landasan dalam melakukan pengembangan sistem informasi perpustakaan pada SMK 1 Muhammadiyah

Matang. Data tersebut diperoleh dengan cara mengkaji penelitian yang sudah ada , membaca referensi, serta menarik kesimpulan sehingga diperoleh teori ilmiah yang melandasi penelitian. Dalam penelitian ini peneliti menggunakan beberapa buku dan jurnal internasional maupun nasional. Salah satu buku dan jurnal internasional yang digunakan oleh peniliti yaitu, buku dari Tom Pender (2003) yang berjudul "*UML Bible*" dan jurnal internasional yang dibuat oleh Manish Kumar, Santosh Kumar Singh, dan Dr. R. K. Dwivedi yang berjudul "*A Detail Study of Agile Software Developnment with Extreme Programming*" diterbitkan oleh *International Journal of Advanced Research in Computer Science and Software Engineering* (IJARCSSE) Volume 5, Nomor 10, Oktober 2015.

3.1.2 Perencanaan (*Planning*)

Pada tahap perencanaan ini, bertujuan untuk mengetahui kebutuhan pengembangan sistem informasi perpustakaan. Hasil dari analisis kebutuhan diperoleh kebutuhan sistem yang berfokus pada tujuan pembuatan sistem informasi perpustakaan dan melakukan pemodelan sistem yang dibutuhkan. Dalam tahap ini dilakukannya *user stories* dan hasil dari *user stories* tersebut dapat ditarik benang merah terhadap kebutuhan sistem yang diinginkan oleh *stackholder*. Peneliti melakukan *user stories* ini menggunakan metode wawancara. Pada tahap perencanaan ini juga, peneliti melakukan pemodelan untuk membangun kebutuhan sistem menggunakan bahasa pemodelan *Unified Modeling Language* (UML), yang menggunakan beberapa diagram dari UML yaitu *use case diagram* dan *use case scenario.*

3.1.2.1 *Wawancara*

Pada tahap wawancara ini, peneliti melakukan kegiatan diskusi serta tanya jawab dengan narasumber yang dianggap memiliki pengetahuan terhadap permasalahan pada penelitian. Tujuan dari wawancara yaitu untuk mengetahui apa saja permasalahan yang terjadi pada Perpustakaan di SMK 1 Muhammadiyah Malang. Pada tahap wawancara peniliti mewawancarai pustakawan yang bernama Ibu Esty Hartini yang di Perpustakaan SMK 1 Muhammadiyah Malang guna mengumpulkan data dan menganalisis kebutuhan sistem yang akan dirancang dengan menggunakan pertanyaan seperti pada (Lampiran A).

3.1.3 Perancangan (*Design*)

Setelah melakukan tahap analisis kebutuhan sistem, tahap selanjutnya adalah melakukan perancangan sistem yang akan dibangun. Pada tahap analisis kebutuhan yang telah dilakukan, diperoleh hasil berupa fitur-fitur yang harus ada pada sistem. Pada tahap ini akan dilakukan perancangan arsitektur sistem dan alur kerja sistem sesuai dengan kebutuhan fungsional. Pemodelan akan dilakukan untuk membangun sistem dengan menggunakan bahasa pemodelan *Unified Modeling Language* (UML), yang menggunakan beberapa diagram dari UML yaitu *sequence diagram* dan *class diagram*. Langkah terakhir yang akan dilakukan ialah merancangan struktur penyimpanan data di dalam *database* dan perancangan antar muka pada sistem.

3.1.4 Implementasi (*Coding*)

Pada tahap implementasi ini dilakukan pengkodean menggunakan bahasa pemrograman *C-Sharp* (C#) untuk membuat sistem berbasis desktop menggunakan aplikasi pemrograman Miscrosoft Visual Studio. Hasil dari tahap perancangan akan dilakukan pengimplementasian sistem menggunakan metode pemrograman *Object Orientation Programming* (OOP). Tahap implementasi akan menghasilkan sebuah sistem informasi perpustakaan yang berbasis desktop dan pembuatan tabel-tabel di dalam basis data dengan mengimplementasikan MySql sebagai *Database Management System* (DBMS).

3.1.5 Pengujian dan Analisis (*Testing*)

Sebelum diterapkannya sistem informasi perpustakaan ini, sistem harus melewati tahap pengujian terlebih dahulu. Pengujian dan analisis dilakukan guna mengidentifikasi kesalahan-kesalahan yang akan terjadi pada sistem. Setelah memperoleh hasil pengujian, dapat dilakukan analisis dari sistem yang diimplementasikan. Pada penilitian ini peneliti akan menggunakan pengujian *white-box testing*, *black-box testing* dengan metode validasi serta *User Acceptance Testing* (UAT) dengan menggunakan kriteria *usability testing*.

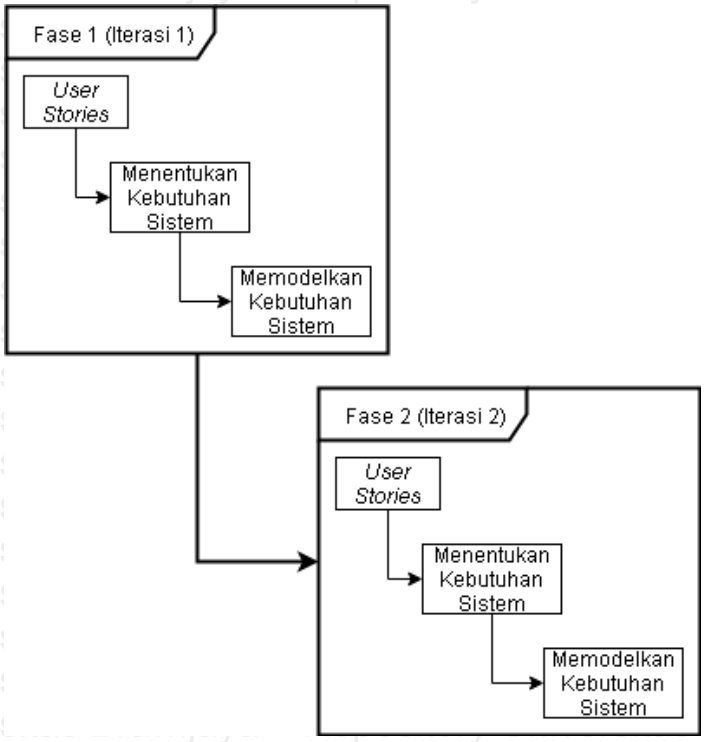
3.1.6 Kesimpulan dan Saran

Kesimpulan dan saran merupakan tahapan terakhir dalam sebuah penelitian. Setelah melakukan pengujian serta pengimplementasian, maka penarikan dan pembuatan kesimpulan dapat ditakukan. Penarikan kesimpulan ini bertujuan untuk mengetahui letak kelebihan dan kekurangan dari sistem yang telah dikembangkan oleh peneliti, dan akan memberikan saran apabila akan melakukan penelitian lebih lanjut.

BAB 4 HASIL DAN PEMBAHASAN

4.1 Perencanaan (*Planning*)

Pada tahap perencanaan ini, peneliti menganalisis kebutuhan yang dibutuhkan dan diprioritaskan oleh sistem. Peneliti melakukan analisis terhadap kebutuhan fungsional sistem dari sistem yang akan dibangun karena pada hasil wawancara peniliti tidak menemukan kebutuhan Non-Fungsional (Lampiran A.1). Pada kebutuhan fungsional diperlukan identifikasi kebutuhan user yang diperoleh dari *user story*. Pada tahap ini terjadi iterasi terhadap tahapan perencanaan yang dilakukan peneliti sampai *stackholder* telah memverifikasi hasil dari analisis peneliti pada tahap perencanaan ini. Tahapan iterasi akan digambarkan pada Gambar 4.1.



Gambar 4.1 Diagram Iterasi pada Tahap Perencanaan

4.1.1 Iterasi 1

4.1.1.1 *User Stories*

Pada bagian *user stories*, mendeskripsikan aktor — aktor yang berperan pada penggunaan sistem yang diperoleh dari *user stories* dan akan dicantumkan pada *Use Case Diagram* nantinya. Berikut adalah aktor dan karakteristik atau hal yang dapat dilakukan terhadap sistem yang didapatkan peneliti dari hasil *user stories* menggunakan metode wawancara (Lampiran A.1) yang mana Pengunjung sebagai (siswa, guru, staff) dan *Admin* sebagai (pustakawan) akan ditabelkan pada Tabel 4.1.

Tabel 4.1 *User Stories*

No	Aktor	User Story
1	Pengunjung	• Sebagai seorang pengunjung, saya ingin mencari buku yang ingin dibaca atau dipinjamkan, sehingga saya tidak perlu melakukan pencarian buku secara manual yang menghabiskan banyak waktu.
2	*Admin*	• Sebagai seorang *admin*, saya ingin melakukan tugas saya sebagai pustakawan tanpa ada campur tangan pengguna lain, sehingga pengunjung tidak dapat mengakses sistem sebagai *admin*. • Sebagai seorang *admin*, saya ingin melakukan pengelolaan buku antara lain penambahan, perubahan, dan penghapusan data buku, sehingga saya tidak memerlukan buku folio sebagai rekap data buku. • Sebagai seorang *admin*, saya ingin melakukan pelayanan perpustakaan yaitu transaksi peminjaman dan transaksi pengembalian, sehingga saya tidak memerlukan buku folio sebagai rekap dari pelayanan perpustakaan. • Sebagai seorang *admin*, saya ingin melakukan pembuatan laporan secara otomatis, sehingga saya dapat lebih mudah melihat daftar transaksi tanpa harus menulis ulang pada buku folio.

4.1.1.2 Kebutuhan Sistem

Kebutuhan sistem merupakan analisis yang dilakukan terhadap kebutuhan fungsional sistem. Kebutuhan fungsional merupakan fungsionalitas atau proses yang harus disediakan sistem sesuai dengan yang diminta oleh *stackholder* yang diperoleh dari *user story*. Kebutuhan fungsional sistem yang telah dibuat telah melewati tahap verifikasi kepada *stackholder* (Lampiran B). Aturan penomoran kebutuhan sistem yang akan didalam sistem akan ditabelkan pada Tabel 4.2 dan kebutuhan fungsional sistem akan ditabelkan pada Tabel 4.3.

Tabel 4.2 Aturan Penomoran Kebutuhan Sistem

Kode	Keterangan
SIMP-	Sistem Informasi Manajemen Perpustakaan
-F-	Kebutuhan Fungsional
-(x)	Nomor urutan Kode Kebutuhan

Tabel 4.3 Kebutuhan Fungsional Sistem

No	Kode	Nama Fitur	*User*	Deskripsi dan Spesifikasi
1.	SIMP-F-01	Cari Buku	Pengunjung	• Pengunjung dapat mencari buku dengan kriteria pencarian. • Sistem akan menampilkan buku yang telah dicari oleh pengunjung.
2.	SIMP-F-02	*Login*	*Admin*	• *Admin* melakukan *login* pada sistem. • Sistem menampilkan form *login* dengan *password field* dan tombol *Login*.
3.	SIMP-F-03	Cari Buku	*Admin*	• *Admin* dapat mencari buku dengan dengan kriteria pencarian. • Sistem akan menampilkan buku yang telah dicari oleh *Admin*.
4.	SIMP-F-04	Kelola Buku	*Admin*	• *Admin* dapat melakukan penambahan buku, melakukan perubahan data buku dan melakukan penghapusan data buku dengan melakukan login terlebih dahulu. • Sistem akan menampilkan pilihan menu dari fitur kelola buku yang akan dipilih oleh *Admin* untuk melakukan pengelolaan buku.

Tabel 4.3 Kebutuhan Fungsional Sistem (Lanjutan)

No	Kode	Nama Fitur	*User*	Deskripsi dan Spesifikasi
5.	SIMP-F-05	Transaksi Peminjaman	*Admin*	• *Admin* melakukan validasi peminjaman buku yang telah di cari oleh Pengunjung dan akan dipinjam oleh Pengunjung. Sistem akan menampilkan form peminjaman dengan format detail peminjam buku, detail buku, jumlah buku yang dipinjam, tanggal peminjaman, dan tanggal kapan buku harus dikembalikan oleh peminjam. • Sistem akan mencatat semua data dari form peminjaman dengan kode unik yaitu kode peminjaman dan sistem akan melakukan cetak bukti peminjaman untuk disimpan oleh Pengunjung
6.	SIMP-F-06	Transaksi Pengembalian	*Admin*	• *Admin* melakukan validasi pengembalian buku. Sistem akan menampilkan form pengembalian buku dan *Admin* menuliskan kode peminjaman yang akan diberikan Pengunjung. • Sistem akan menampilkan detail peminjam, tanggal peminjaman, tanggal pengembalian, detail buku, jumlah buku, dan detail denda.

Tabel 4.3 Kebutuhan Fungsional Sistem (Lanjutan)

No	Kode	Nama Fitur	*User*	Deskripsi dan Spesifikasi
7.	SIMP-F-07	Melihat Riwayat	*Admin*	• *Admin* dapat melihat daftar riwayat peminjaman buku dan mencari riwayat peminjaman buku berdasarkan tanggal. • Sistem akan menampilkan seluruh riwayat peminjaman buku dan akan menampilkan pencarian riwayat peminjaman buku apabila *Admin* melakukan pencarian. Pada opsi riwayat *Admin* juga sistem dapat melakukan cetak laporan. • Sistem akan melakukan pencetakan laporan sesuai dengan jenjang waktu yang diinginkan *Admin*.
8.	SIMP-F-8	*Logout*	*Admin*	• *Admin* melakukan *logout* dari sistem. • Sistem akan menampilkan pesan timbul (*pop-up*) dengan pesan "apakah anda yakin untuk logout ?" dengan menyediakan tombol "Iya" dan "Tidak".

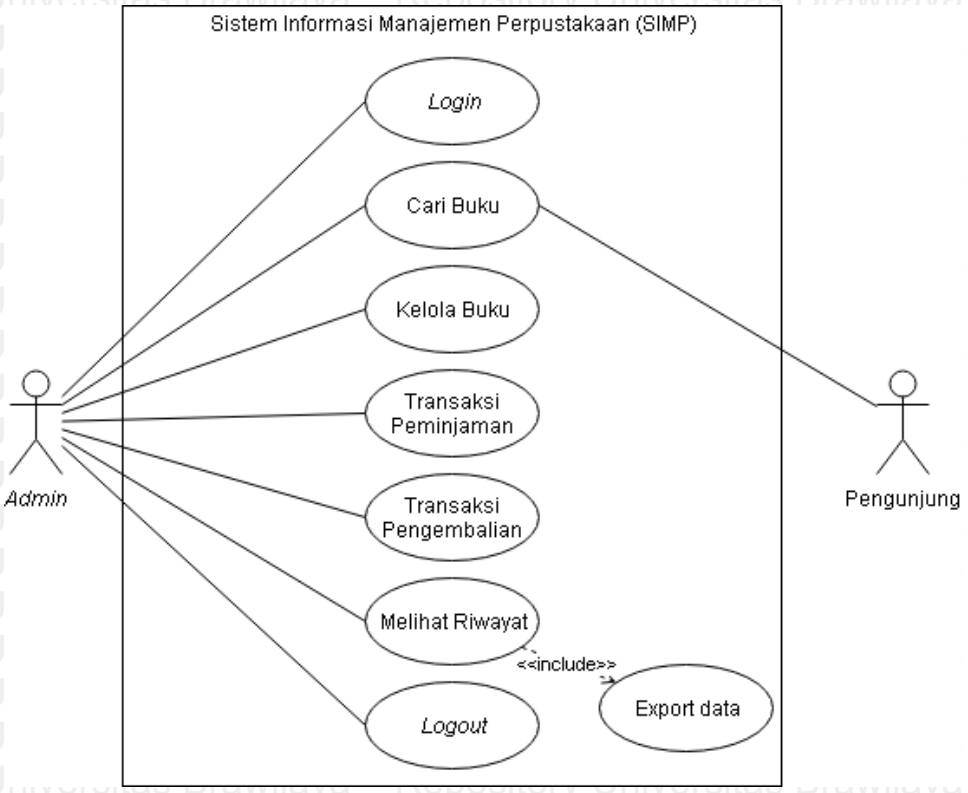
4.1.1.3 Pemodelan Kebutuhan Sistem

Pada tahap pemodelan kebutuhan sistem ini dilakukan agar lebih mudah dalam melanjutkan pada tahap yang selanjutnya yaitu melakukan perancangan sistem. Peneliti melakukan pemodelan menggunakan *Unfied Modeling Language* (UML). Diagram yang akan digunakan diantaranya, *use case diagram* dan *use case scenario*. Pemodelan kebutuhan sistem ini akan dibuat berdasarkan tabel fungsionalitas sistem Tabel 4.3 yang telah diverifikasi oleh *stackholder* (Lampiran B).

a. Use Case Diagram

Use case pada perencanaan sistem informasi manajemen perpustakaan ini terdiri dari 2 (dua) aktor yaitu Pengunjung dan *Admin*. Pengunjung adalah *user* yang hanya bisa melakukan pencarian buku yang ingin dipinjamkan. Sedangkan *Admin* adalah user yang dapat melakukan

semua pengelolaan perpustakaan antara lain, melakukan pencarian buku, melakukan pengelolaan buku, melakukan transaksi peminjaman maupun pengembalian buku, dan export data transaksi. Semua fitur yang dilakukan admin menginclude fungsi *login* dan *logout*. Adapun rancangan *use case* pada sistem ini dapat dilihat pada Gambar 4.1.



Gambar 4.2 *Use Case Diagram* Sistem Informasi Perpustakaan (SIMP)

Pada Gambar 4.1 adalah *use case* diagram SIMP beserta user, actor atau pengguna dari sistem tersebut. Terdapat 5 (lima) fungsionalitas sistem yang diambil dari tabel fungsional Tabel 4.3. Terdapat 5 (dua) *use case* tanpa aktor yaitu, tambah buku, edit buku, hapus buku, export data laporan dan cetak bukti peminjaman. *Use case* tanpa aktor ini berada pada *use case* lain atau menggunakan *use case* lain terlebih dahulu sebagai syarat terprosesnya *use case*.

b. *Use Case Scenario*

Setiap skenario dalam sistem yang akan dibangun, dideskripsikan urutan aksi atau langkah yang dilakukan oleh pengunjung dan *admin* pada saat melakukan interaksi pada sistem, baik yang sukses maupun gagal. Dalam *use case scenario* akan dijelaskan alur proses sistem, *precondition* (kondisi sebelum proses yang harus dipenuhi), *postcondition* (kondisi setelah proses dijalankan), *alternative flow* (skenario alternatif). *Use case scenario* akan dibagi berdasarkan aktor yang telah dimodelkan menggunakan *use case diagram* pada Gambar 4.1 yaitu pengunjung dan *admin*. *Use case scenario* sistem akan dijelaskan pada Tabel 4.4-4.11.

- ***Use Case Scenario* Pengunjung**

Pada bagian ini akan dijelaskan *use case scenario* dari aktor pengunjung. Pengunjung sebagai aktor pada sistem ini hanya mempunyai 1 (satu) fitur yaitu, mencari buku. Adapun *use case scenario* akan di tabelkan pada Tabel 4.4.

Tabel 4.4 *Use Case Scenario* Pengunjung Cari Buku

Use case ID	SIMP-F-01
Sub usecase ID	-
Use case name	Cari Buku
Actor	Pengunjung
Description	Pengunjung dapat melakukan pencarian buku yang diinginkan berdasarkan tahun buku, judul buku,kode buku, pengarang buku ataupun penerbit buku.
Precondition	Memilih masuk sebagai pengunjung pada halaman utama sistem.
Main flow	1.Sistem akan menampilkan langsung halaman pencarian buku. 2. Pengunjung melakukan pilihan pada menu dropdown (seleksi pencarian buku). 3. Pengunjung mengisi kolom pencarian menggunakan kata kunci yang akan dicari. 4. Pengunjung menekan tombol “cari buku”. 5. Sistem akan menampilkan semua daftar buku yang terkait dengan kata kunci yang dituliskan pengunjung.
Alternative flow	1. Sistem akan menampilkan *pop-up* notifikasi bahwa kolom pencarian masih kosong, apabila pengunjung belum mengisi kolom pencarian dan sudah menekan tombol “cari buku”. 2. Sistem akan menampilkan *pop-up* notifikasi bahwa pencarian yang dilakukan tidak menemukan data yang cocok, apabila tidak ada kecocokan kata kunci yang dimasukkan oleh pengunjung dengan seluruh daftar buku yang ada di dalam sistem.
Postcondition	Data buku yang ditemukan kecocokan dengan kata kunci yang dituliskan pengunjung akan ditampilkan oleh sistem.

- *Use Case Scenario Admin*

  Pada bagian ini akan dijelaskan *use case scenario* dari aktor *admin*. Admin sebagai aktor pada sistem ini mempunyai 4 (empat) fitur dan 6 (enam) fungsi yaitu, mencari buku, mengelola buku (tambah buku, edit buku, dan hapus buku), melakukan transaksi peminjaman, melakukan transaksi pengembalian dan melihat riwayat. Adapun *use case scenario* akan di tabelkan pada.

Tabel 4.5 *Use Case Scenario Admin* Cari Buku

Use case ID	SIMP-F-03
Sub usecase ID	-
Use case name	Cari Buku
Actor	*Admin*
Description	*Admin* dapat mencari buku yang diinginkan berdasarkan judul buku, pengarang buku, ataupun penerbit buku.
Precondition	Memilih masuk sebagai *admin* pada halaman utama sistem dan melakukan *login*.
Main flow	1. Sistem akan menampilkan halaman utama *admin*. 2. Halaman dashboard *admin* akan menampilkan form pencarian buku. 3. *Admin* melakukan pilihan pada menu dropdown (seleksi pencarian buku). 4. *Admin* mengisi kolom pencarian menggunakan kata kunci yang akan dicari. 5. *Admin* menekan tombol “cari buku”. 6. Sistem akan menampilkan semua daftar buku yang terkait dengan kata kunci yang dituliskan *admin*.
Alternative flow	1. Sistem akan menampilkan *pop-up* notifikasi bahwa kolom pencarian masih kosong, apabila pengunjung belum mengisi kolom pencarian dan sudah menekan tombol “cari buku”. 2. Sistem akan menampilkan *pop-up* notifikasi bahwa pencarian yang dilakukan tidak menemukan data yang cocok, apabila tidak ada kecocokan kata kunci yang dimasukkan oleh pengunjung dengan seluruh daftar buku yang ada di dalam sistem.
Postcondition	Data buku yang ditemukan kecocokan dengan kata kunci yang dituliskan *admin* akan ditampilkan oleh sistem.

Tabel 4.6 *Use Case Scenario Admin* Tambah Buku

Use case ID	SIMP-F-04
Sub usecase ID	SIMP-F-04-A
Use case name	Tambah Buku
Actor	*Admin*
Description	*Admin* dapat melakukan penambakan daftar buku baru pada sistem.
Precondition	Memilih masuk sebagai *admin* pada halaman utama sistem dan melakukan *login*.
Main flow	1. Sistem akan menampilkan halaman utama *admin*. 2. *Admin* memilih dan menekan "Menu" pada menu bar dan menekan sub menu "Tambah Buku" 3. Sistem akan menampilkan halaman kelola buku. 4. *Admin* memilih opsi tambah buku dengan menekan tombol "tambah buku" pada halaman kelola buku. 5. Sistem akan menampilkan *form* tambah buku. 6. *Admin* mengisi *form* yang disediakan sistem mengenai data buku yang ingin ditambahkan oleh *admin*. 7. *Admin* menekan tombol "tambah buku".
Alternative flow	1. Sistem akan menampilkan *pop-up* notifikasi bahwa ada kolom yang belum terisi oleh *admin* pada *form* tambah buku. 2. Sistem akan menampilkan *pop-up* notifikasi bahwa data buku yang sama telah tersedia pada *database* daftar buku, dan sistem menawarkan apakah tetap akan menambahkan buku dengan hanya memasukkan jumlah buku yang ingin dimasukkan oleh *admin*.
Postcondition	Data buku yang telah ditambahkan *admin* tersimpan pada *database* sistem.

Tabel 4.7 *Use Case Scenario Admin* Edit Buku

Use case ID	SIMP-F-04
Sub usecase ID	SIMP-F-04-B
Use case name	Edit Buku
Actor	*Admin*
Description	*Admin* dapat melakukan pengeditan pada daftar buku yang tersedia di *database* sistem.

Tabel 4.7 *Use Case Scenario Admin* Edit Buku (Lanjutan)

Precondition	Memilih masuk sebagai *admin* pada halaman utama sistem dan melakukan *login*.
Main flow	1. Sistem akan menampilkan halaman utama *admin*. 2. *Admin* memilih dan menekan "Menu" pada menu bar dan menekan sub menu "Hapus Buku" 3. Sistem akan menampilkan seluruh daftar buku yang ada pada *database* sistem. 4. *Admin* melakukan perubahan data buku dengan menekan pilihan buku pada table yang menampilkan seluruh daftar buku. 5. Sistem menampilkan *form* edit buku, dengan kolom data buku telah terisi oleh data buku yang dipilih *admin* untuk dirubah. 6. *Admin* melakukan perubahan data pada kolom yang diinginkan. 7. *Admin* menekan tombol "Simpan" apabila telah melakukan perubahan yang diinginkan oleh *admin*.
Alternative flow	1. *Admin* juga dapat melakukan pencarian buku yang diinginkan untuk dilakukan perubahan data. 2. Sistem akan menampilkan *pop-up* notifikasi bahwa ada kolom yang belum terisi oleh *admin* pada *form* edit buku. 3. *Admin* dapat melakukan perubahan data buku dengan melakukan pencarian buku terlebih dahulu pada kolom pencarian di atas daftar buku.
Postcondition	Data buku yang telah dirubah oleh *admin* tersimpan pada *database* sistem.

Tabel 4.8 *Use Case Scenario Admin* Hapus Buku

Use case ID	SIMP-F-04
Sub usecase ID	SIMP-F-04-C
Use case name	Hapus Buku
Actor	*Admin*
Description	*Admin* dapat melakukan penghapusan pada daftar buku yang tersedia di *database* sistem.

Tabel 4.8 *Use Case Scenario Admin* Hapus Buku (Lanjutan)

Precondition	Memilih masuk sebagai *admin* pada halaman utama sistem dan melakukan *login*.
Main flow	1. Sistem akan menampilkan halaman utama *admin*. 2. *Admin* memilih dan menekan "Menu" pada menu bar dan menekan sub menu "Hapus Buku". 3. Sistem akan menampilkan seluruh daftar buku yang ada pada *database* sistem. 4. *Admin* melakukan penghapusan data buku dengan memilih buku yang akan dihapus. 5. Sistem *pop-up* notifikasi apakah benar admin akan menghapus data buku tersebut.
Alternative flow	1. *Admin* juga dapat melakukan pencarian buku yang diinginkan untuk dilakukan perubahan data.
Postcondition	Data buku yang telah dihapus oleh *admin* akan terhapus juga pada *database* sistem.

Tabel 4.9 *Use Case Scenario Admin* Transaksi Peminjaman

Use case ID	SIMP-F-05
Sub usecase ID	-
Use case name	Transaksi Peminjaman
Actor	*Admin*
Description	*Admin* dapat melakukan transaksi peminjaman buku.
Precondition	Memilih masuk sebagai *admin* pada halaman utama sistem dan melakukan *login*.
Main flow	1. Sistem akan menampilkan halaman utama *admin*. 2. *Admin* memilih opsi peminjaman buku dengan memilih dan menekan tombol "Menu" pada menu bar dan memilih "Transaksi". 3. Sistem akan *form* peminjaman buku. 4. *Admin* melakukan pengisian pada kolom data peminjam yang tersedia pada *form* peminjaman buku yang ditampilkan oleh sistem. 5. *Admin* melakukan penambahan buku yang akan dipinjam pada keranjang buku yang ada dengan menekan buku pada tabel daftar buku. 6. *Admin* menekan tombol "Simpan Peminjaman".

Tabel 4.9 *Use Case Scenario Admin* Transaksi Peminjaman(Lanjutan)

Alternative flow	1. Sistem akan menampilkan *pop-up* notifikasi bahwa ada kolom yang belum terisi oleh *admin* pada *form* peminjaman buku.
Postcondition	1. Data peminjaman buku akan tersimpan pada *database* sistem. 2. Sistem akan mencetak bukti peminjaman.

Tabel 4.10 *Use Case Scenario Admin* Transaksi Pengembalian

Use case ID	SIMP-F-06
Sub usecase ID	-
Use case name	Transaksi Pengembalian
Actor	*Admin*
Description	*Admin* dapat melakukan transaksi pengembalian buku.
Precondition	Memilih masuk sebagai *admin* pada halaman utama sistem dan melakukan *login*.
Main flow	1. Sistem akan menampilkan halaman utama *admin*. 2. *Admin* memilih opsi peminjaman buku dengan memilih dan menekan tombol "Menu" pada mebu bar dan memilih "Transaksi". 3. Sistem akan *form* peminjaman buku. 4. *Admin* melakukan pengisian pada kolom kode peminjaman. 5. *Admin* menekan tombol "cari". 6. Sistem akan menampilkan data peminjaman buku yang telah diinputkan kode peminjaman oleh *admin*. 7. *Admin* menyesuaikan data yang ada, menekan tombol lihat denda dan akan memberikan denda apabila sistem menampilkan denda yang diberikan dikarenakan keterlambatan pengembalian. 8. *Admin* menekan tombol "kembalikan".
Alternative flow	Sistem akan menampilkan *pop-up* notifikasi bahwa buku telah dikembalikan apabila kode peminjaman telah divalidasi oleh *admin*.
Postcondition	Data peminjaman buku akan tervalidasi pengembalian buku dan tersimpan pada *database* sistem.

Tabel 4.11 *Use Case Scenario Admin* Melihat Riwayat

Use case ID	SIMP-F-07
Sub usecase ID	-
Use case name	Melihat Riwayat
Actor	*Admin*
Description	*Admin* dapat melihat seluruh daftar riwayat dari peminjaman buku.
Precondition	Memilih masuk sebagai *admin* pada halaman utama sistem dan melakukan *login*.
Main flow	1. Sistem akan menampilkan halaman utama *admin*. 2. *Admin* memilih opsi peminjaman buku dengan memilih dan menekan tombol "Menu" pada mebu bar dan memilih "Riwayat". 3. Sistem akan menampilkan halaman riwayat peminjaman buku. 4. Sistem akan menampilkan seluruh daftar riwayat peminjaman buku yang telah tervalidasi pengembaliannya.
Alternative flow	*Admin* dapat mencetak laporan peminjaman buku dengan menakan tombol "export data" dan sistem mengexport data.
Postcondition	1. Sistem akan menampilkan seluruh data riwayat peminjaman. 2. Sistem export data laporan.

4.1.2 Iterasi 2

4.1.2.1 *User Stories*

Pada iterasi ke-2 (dua) ini, *user stories* didapatkan dari hasil verifikasi kebutuhan sistem yang dilakukan peniti terhadap *stackholder* (Lampiran B). *User stories* pada iterasi ke-2 (dua) akan dijelaskan pada Tabel 4.12

Tabel 4.12 *User Stories*

No	Aktor	User Story
1	*Admin*	• Sebagai seorang *admin*, saya ingin melakukan pembuatan laporan secara otomatis dan langsung melakukan cetak laporan, sehingga saya tidak perlu merubah dan melakukan kesesuaian template terhadap laporan yang dilakukan yang pada Microsoft Excel.

4.1.2.2 Kebutuhan Sistem

Kebutuhan fungsional sistem yang akan diubah adalah kebutuhan fungsional melihat riwayat sistem yang mana "Export Data" diubah menjadi "Cetak Laporan" (*print out* laporan) yang akan dijelaskan pada Tabel 4.13.

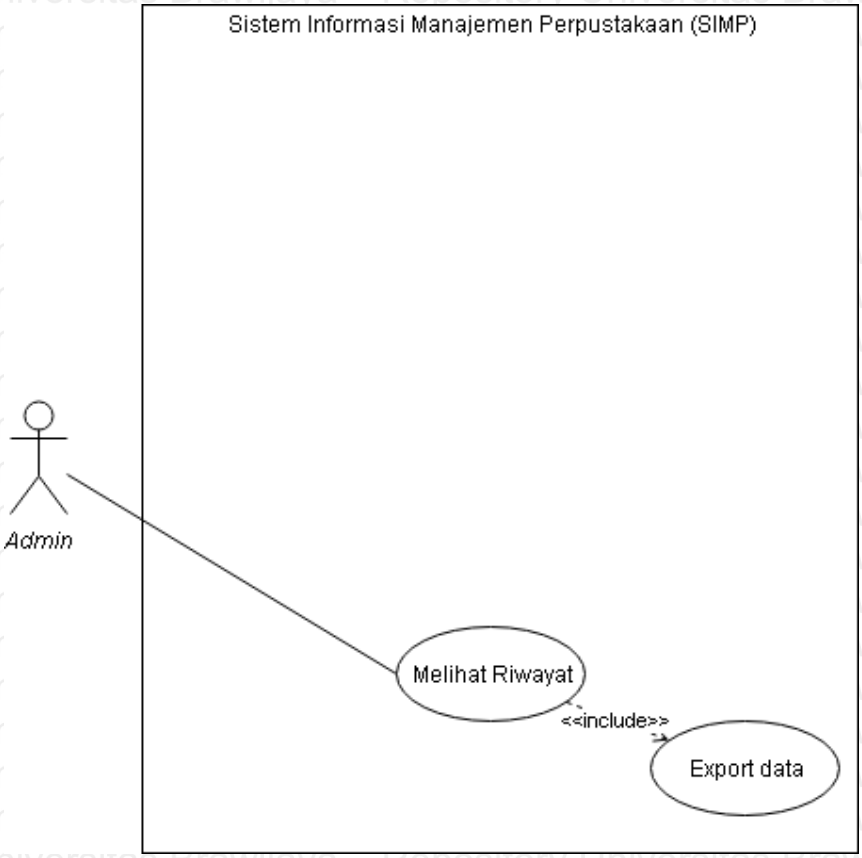
Tabel 4.13 Kebutuhan Fungsional Sistem

No	Kode	Nama Fitur	*User*	Deskripsi dan Spesifikasi
1.	SIMP-F-07	Melihat Riwayat	*Admin*	• *Admin* dapat melihat daftar riwayat peminjaman buku dan mencari riwayat peminjaman buku berdasarkan tanggal. • Sistem akan menampilkan seluruh riwayat peminjaman buku dan akan menampilkan pencarian riwayat peminjaman buku apabila *Admin* melakukan pencarian. Pada opsi riwayat *Admin* juga sistem dapat melakukan cetak laporan. • Sistem akan melakukan pencetakan laporan sesuai dengan jenjang waktu yang diinginkan *Admin*.

4.1.2.3 Pemodelan Kebutuhan Sistem

a. Use Case Diagram

Use Case pada tahap iterasi ke 2 (dua) ini hanya merubah fungsional sistem antara use case *admin* dengan kebutuhan fungsionalitas sistem melihat riwayat buku yang merubah fungsi esport data transaksi menjadi cetak laporan transaksi



Gambar 4.3 *Use Case Diagram* Sistem Informasi Perpustakaan (SIMP)

Use case pada Gambar 4.2 terlihat proses extend yang terjadi pada kebutuhan fungsional melihat riwayat berubah menjadi "Cetak Laporan" yang sebelumnya adalah "Export Data".

b. Use Case Scenario

Use case scenario pada tahap iterasi ke 2 (dua) ini merubah *scenario* sistem export data yang ada pada kebutuhan fungsional *admin* melihat riwayat menjadi mencetak data transaksi yang akan dijelaskan pada Tabel 4.14.

Tabel 4.14 *Use Case Scenario Admin* Melihat Riwayat

Use case ID	SIMP-F-07
Sub usecase ID	-
Use case name	Melihat RIwayat
Actor	*Admin*
Description	*Admin* dapat melihat seluruh daftar riwayat dari peminjaman buku.
Precondition	Memilih masuk sebagai *admin* pada halaman utama sistem dan melakukan *login*.

Tabel 4.14 *Use Case Scenario Admin* Melihat Riwayat (Lanjutan)

Main flow	1. Sistem akan menampilkan halaman utama *admin*. 2. *Admin* memilih opsi peminjaman buku dengan memilih dan menekan tombol "Menu" pada menu bar dan memilih "Riwayat". 3. Sistem akan menampilkan halaman riwayat peminjaman buku. 4. Sistem akan menampilkan seluruh daftar riwayat peminjaman buku yang telah tervalidasi pengembaliannya.
Alternative flow	*Admin* dapat mencetak laporan peminjaman buku dengan menakan tombol "cetak laporan" dan sistem mengexport data.
Postcondition	1. Sistem akan menampilkan seluruh data riwayat peminjaman. 2. Sistem mencetak data laporan.

4.1.3 Hasil Perencanaan (*Planning*)

Hasil dari tahap perencanaan yang telah dilakukan peneliti dari hasil wawancara kepada *stackholder* yang dilakukan pada iterasi ke 1 (satu) (Lampiran A.1) yaitu sistem mempunyai 2 (dua) aktor dengan spesifikasi yang telah dijelaskan pada Tabel 4.1. Dari 2 (dua) aktor tersebut sistem mempunyai 8 (delapan) fitur keseluruhan yang telah diverifikasi oleh *stackholder* (Lampiran B) dijelaskan pada Tabel 4.3. Pada iterasi ke 2 (dua) dilakukan perubahan pada kebutuhan fungsional sistem yang ada pada *user* admin untuk bisa mencetak laporan data transaksi. Setelah melakukan tahap perencanaan (*Planning*) peneliti akan melanjutkan pada tahap perancangan (*Design*).

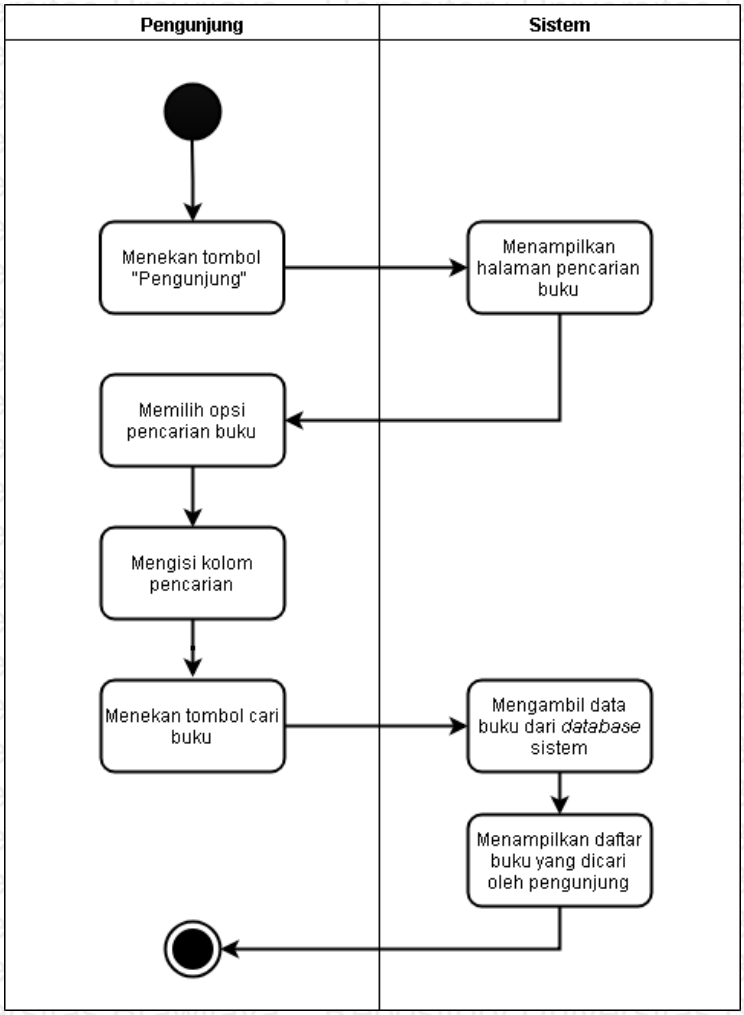
4.2 Perancangan (*Design*)

Pada tahap perancangan ini, setelah melakukan tahap perencanaan yang menghasilkan aktor-aktor berserta fitur yang berada di dalam sistem. Pada tahap ini, dilakukan perancangan arsitektur sistem dan proses jalannya sistem sesuai dengan kebutuhan fungsional sistem. Pada tahap ini, perancangan juga akan dimodelkan *activity diagram, sequence diagram,* dan *class diagram* serta merancang struktur penyimpanan data di dalam *database*.

4.2.1 *Activity Diagram*

Activity diagram pada dasarnya adalah diagram alur untuk mewakili aliran dari satu aktivitas ke aktivitas lain. Aktivitas dapat digambarkan sebagai operasi sistem. *Activity diagram* dibuat untuk mengetahui alur proses dari sebuah fitur didalam sistem, sekaligus mengetahui aktor-aktor yang berperan didalamnya.

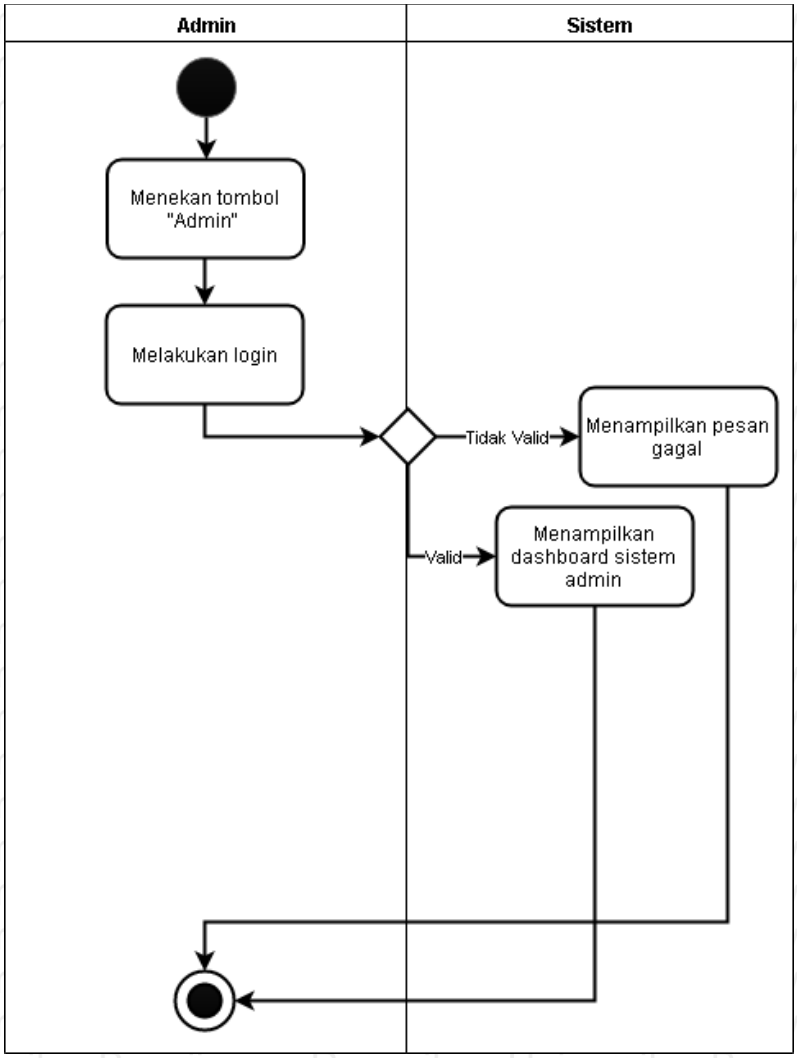
4.2.1.1 *Activity Diagram* Pengunjung Mencari Buku



Gambar 4.4 *Activity Diagram* Pengunjung Mencari Buku

Activity diagram yang tersedia pada Gambar 4.3 menjelaskan alur proses pengunjung untuk melakukan pencarian buku. Pengunjung memulai proses dengan cara memilih opsi masuk ke dalam sebagai pengunjung dengan cara menekan tombol “Pengunjung” pada halaman utama sistem. Sistem akan secara langsung menampilkan halaman pencarian buku dikarenakan pengunjung hanya bias mengakses sistem dengan fungsi pencarian buku saja. Selanjutnya pengunjung memilih opsi pencarian buku yang tersedia di menu *dropdown* yaitu dapat melakukan pencarian buku berdasarkan judul buku, penerbit buku, atau pengarang buku. Pengunjung mengisi kata kunci yang diinginkan dan menekan tombol “Cari Buku”. Sistem secara otomatis akan menampilkan buku yang ingin dicari oleh pengunjung. Apabila buku tidak ditemukan maka akan menampilkan pesan “Buku yang anda cari tidak ditemukan, cek kembali kata kunci anda”.

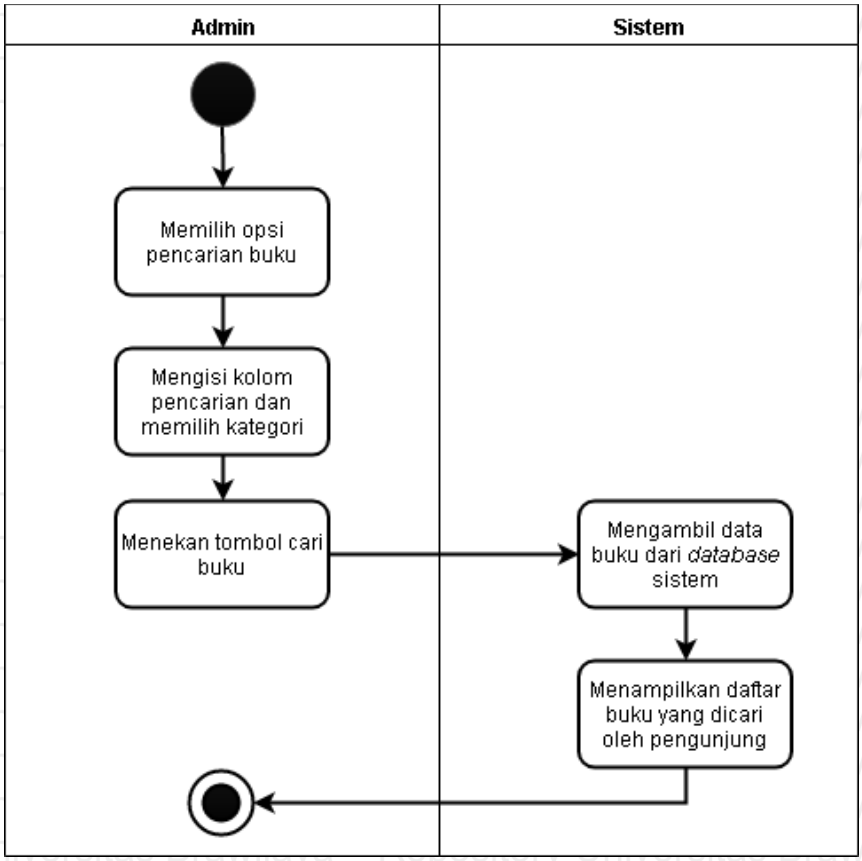
4.2.1.2 *Activity Diagram Admin Login*



Gambar 4.5 *Activity Diagram Admin Login*

Activity diagram yang tersedia pada Gambar 4.4 menjelaskan alur proses *admin* untuk melakukan login kedalam sistem. *Admin* memulai proses dengan cara memilih opsi masuk kedalam sebagai *admin* dengan cara menekan tombol "Admin" pada halaman utama sistem. Sistem akan menampilkan halaman login, dan pengunjung memasukkan password untuk melakukan login kedalam sistem. Jika password yang dimasukkan oleh *admin* salah, maka sistem akan menampilkan pesan gagal login.

4.2.1.3 *Activity Diagram Admin* Mencari Buku

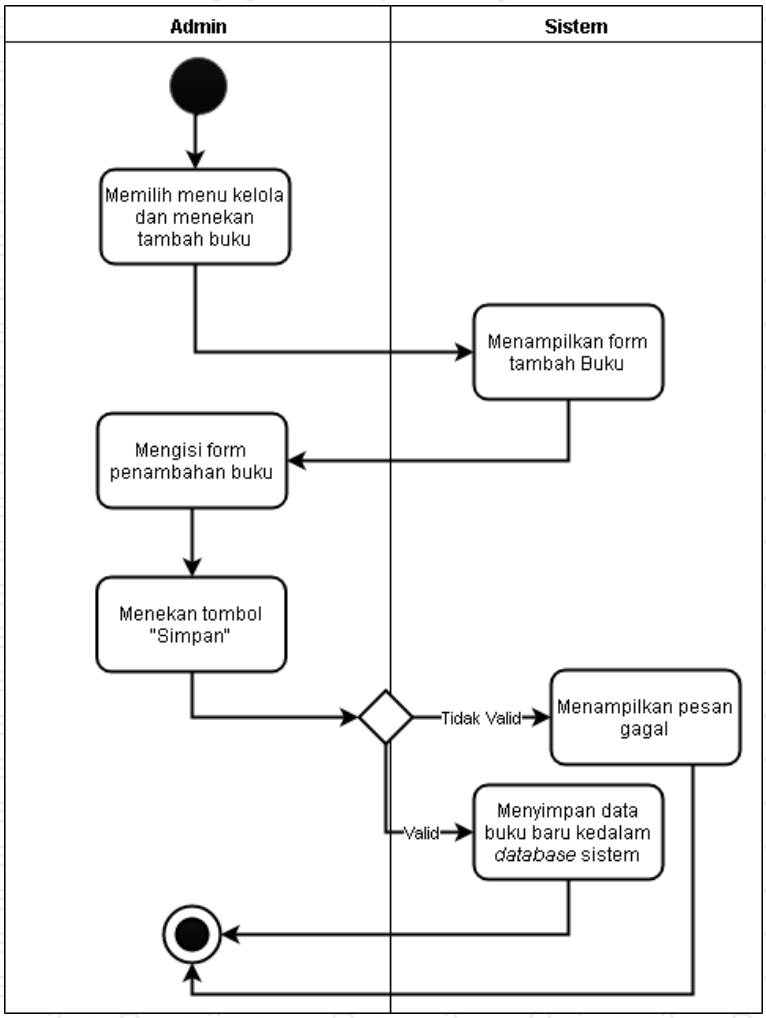


Gambar 4.6 *Activity Diagram Admin* Mencari Buku

Activity diagram yang tersedia pada Gambar 4.5 menjelaskan alur proses *admin* untuk melakukan pencarian buku. Pada Gambar 4.5 *admin* melakukan pencarian sistem setelah *admin* melakukan proses *login*. Sistem akan secara otomatis akan menampilkan halaman Pencarian buku yang sekaligus halaman dashboard *admin* setelah *admin* berhasil *login*. Selanjutnya *admin* memilih opsi pencarian buku yang tersedia di menu *dropdown* yaitu dapat melakukan pencarian buku berdasarkan judul buku, penerbit buku, atau pengarang buku. *Admin* mengisi kata kunci yang diinginkan dan menekan tombol "Cari Buku". Sistem secara otomatis akan menampilkan buku yang ingin dicari oleh pengunjung. Apabila buku tidak ditemukan maka akan menampilkan pesan "Buku yang anda cari tidak ditemukan, cek kembali kata kunci anda".

4.2.1.4 *Activity Diagram Admin* Kelola Buku

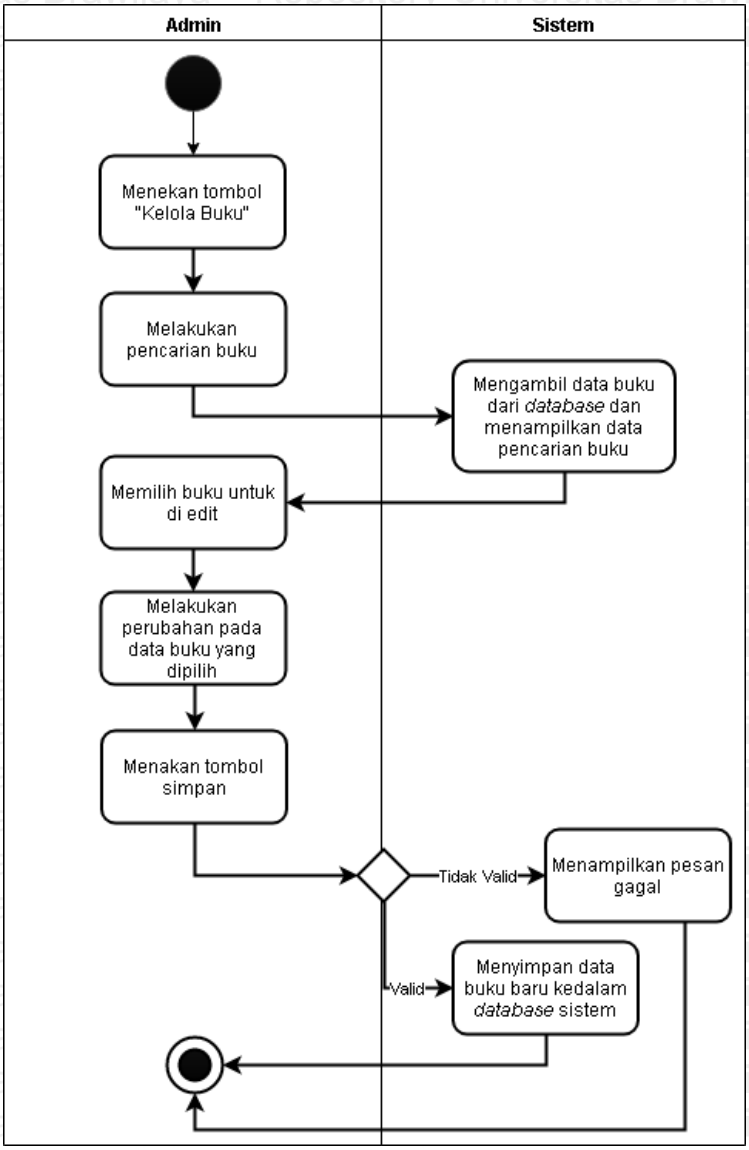
(a) *Activity Diagram Admin* Tambah Buku



Gambar 4.7 *Activity Diagram Admin* Tambah Buku

Activity diagram yang tersedia pada Gambar 4.6 menjelaskan alur proses *admin* untuk melakukan tambah buku pada fitur kelola buku. Alur proses di atas setelah *admin* melakukan alur proses login dan kelola buku. Setelah sistem melakukan menampilkan dashboard *admin. Admin* melakukan penambahan buku dengan cara menekan tombol "Tambah Buku" yang tersedia pada sub menu "Kelola" pada menu bar. Sistem akan menampilkan halaman penambahan buku, dan *admin* mengisi form yang telah ada untuk melengkapi data-data buku yang tersimpan. Setelah data lengkap maka *admin* dapat menekan tombol "Simpan". Apabila data yang diisi oleh *admin* tidak lengkap maka sistem akan menampilkan pesan "Periksa kembali dan lengkapi data buku".

(b) *Activity Diagram Admin* Edit Buku

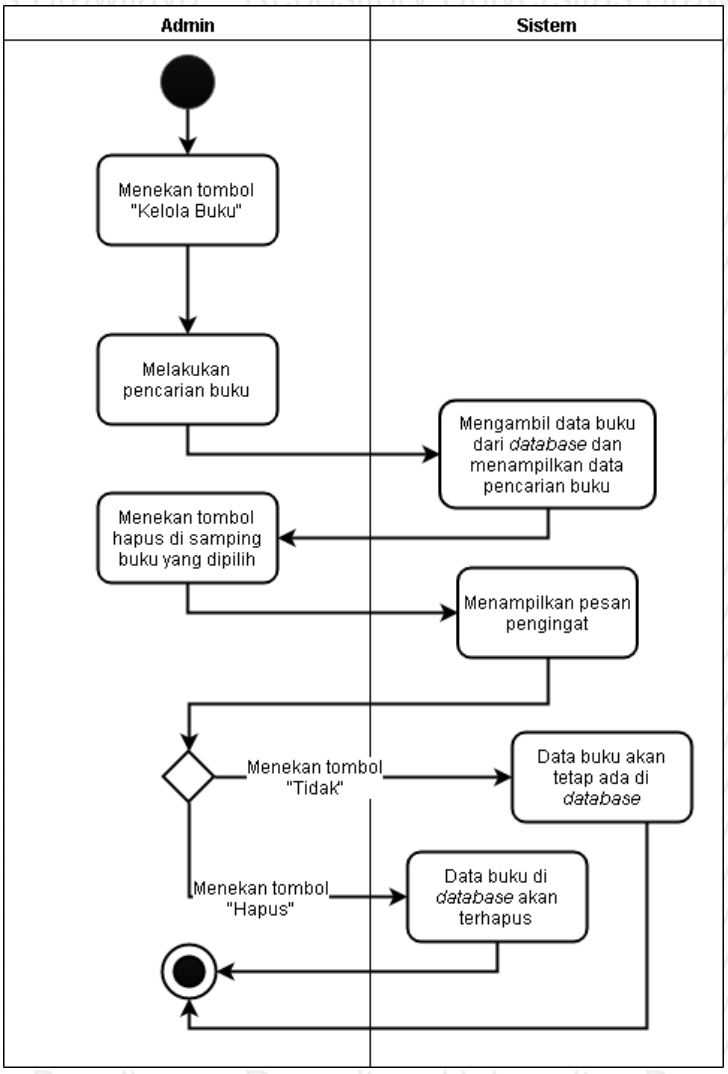


Gambar 4.8 *Activity Diagram Admin* Edit Buku

Activity diagram yang tersedia pada Gambar 4.7 menjelaskan alur proses *admin* untuk melakukan edit buku pada fitur kelola buku. Alur proses di atas setelah *admin* melakukan alur proses login dan kelola buku. Setelah sistem melakukan menampilkan dashboard *admin. Admin* melakukan pengeditan buku dengan cara menekan sub menu "Edit Buku" yang tersedia pada sub menu "Kelola" di menu bar. Sistem akan menampilkan halaman pengeditan buku, *admin* melakukan pencarian seperti yang telah dijelaskan pada *activity diagram admin* mencari buku Gambar 4.4 dan memilih buku untuk dilakukan perubahan data. Setelah data yang telah di edit telah dipenuhi dan lengkap maka *admin* dapat menekan tombol "Simpan". Apabila data yang diisi oleh *admin* tidak

lengkap maka sistem akan menampilkan pesan “Periksa kembali dan lengkapi data buku”.

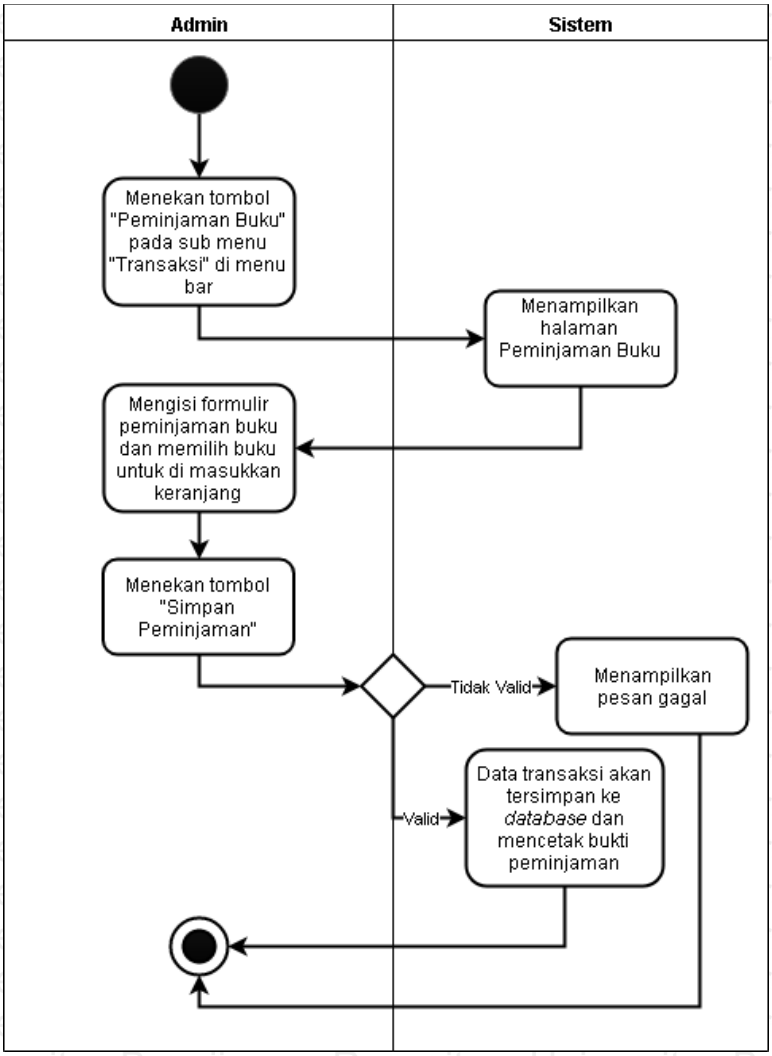
(c) *Activity Diagram Admin* Hapus Buku



Gambar 4.9 *Activity Diagram Admin* Hapus Buku

Activity diagram yang tersedia pada Gambar 4.8 menjelaskan alur proses *admin* untuk melakukan hapus buku pada fitur kelola buku. Alur proses di atas setelah *admin* melakukan alur proses login dan kelola buku. Setelah sistem melakukan menampilkan dashboard *admin. Admin* melakukan penghapusan buku dengan cara menekan sub menu “Hapus Buku” pada menu “Kelola” yang tersedia di menu bar. Sistem akan menampilkan halaman penghapusan buku, *admin* melakukan pencarian buku seperti yang telah dijelaskan pada *activity diagram admin* mencari buku Gambar 4.4. Setelah menemukan buku yang ingin di hapus, maka *admin* dapat menekan tombol hapus. Sistem akan menampilkan pesan pengingat apakah *admin* akan benar-benar ingin menghapus buku.

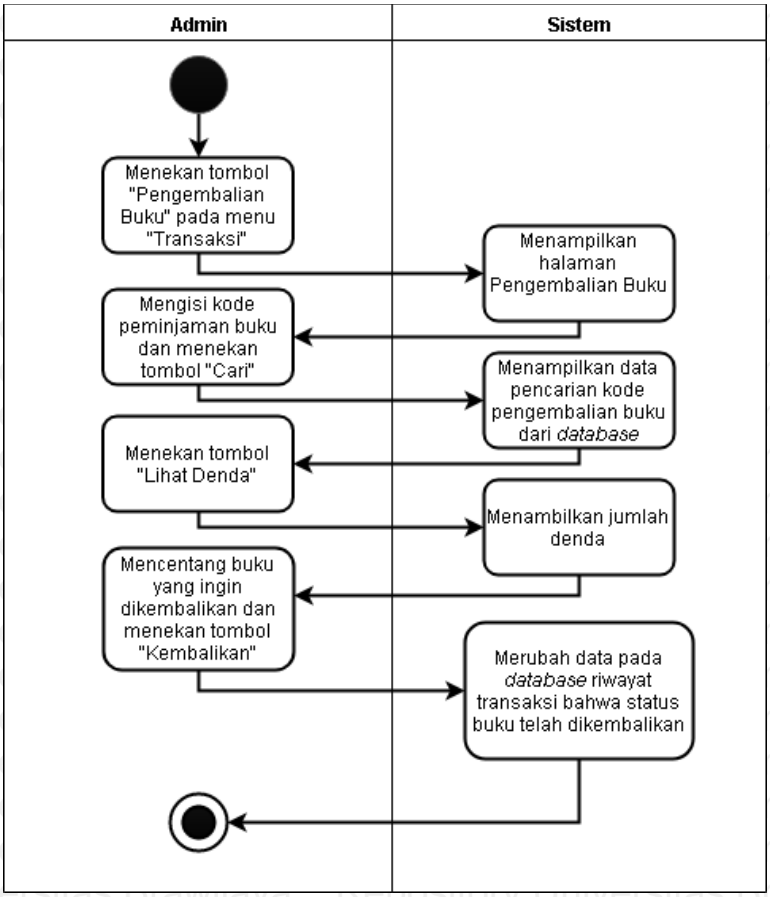
4.2.1.5 *Activity Diagram Admin* Melakukan Transaksi Peminjaman Buku



Gambar 4.10 *Activity Diagram Admin* Melakukan Peminjaman Buku

Activity diagram yang tersedia pada Gambar 4.9 menjelaskan alur proses *admin* untuk melakukan transaksi peminjaman buku. Alur proses diatas setelah *admin* melakukan alur proses login. Setelah sistem melakukan menampilkan dashboard *admin. Admin* melakukan transaksi peminjaman buku dengan cara menekan sub menu "Peminjaman Buku" pada menu "Transaksi" yang tersedia di menu bar. Sistem akan menampilkan halaman peminjaman buku, *admin* melakukan pengisian form peminjaman. Setelah form telah terisi lengkap *admin* dapat menekan tombol "Simpan Peminjaman" dan sistem akan mencetak bukti peminjaman yang akan diserahkan kepada peminjam. Apabila form peminjaman tidak lengkap diisi oleh *admin* maka sistem akan menampilkan pesan "Harap lengkapi data yang akan disimpan".

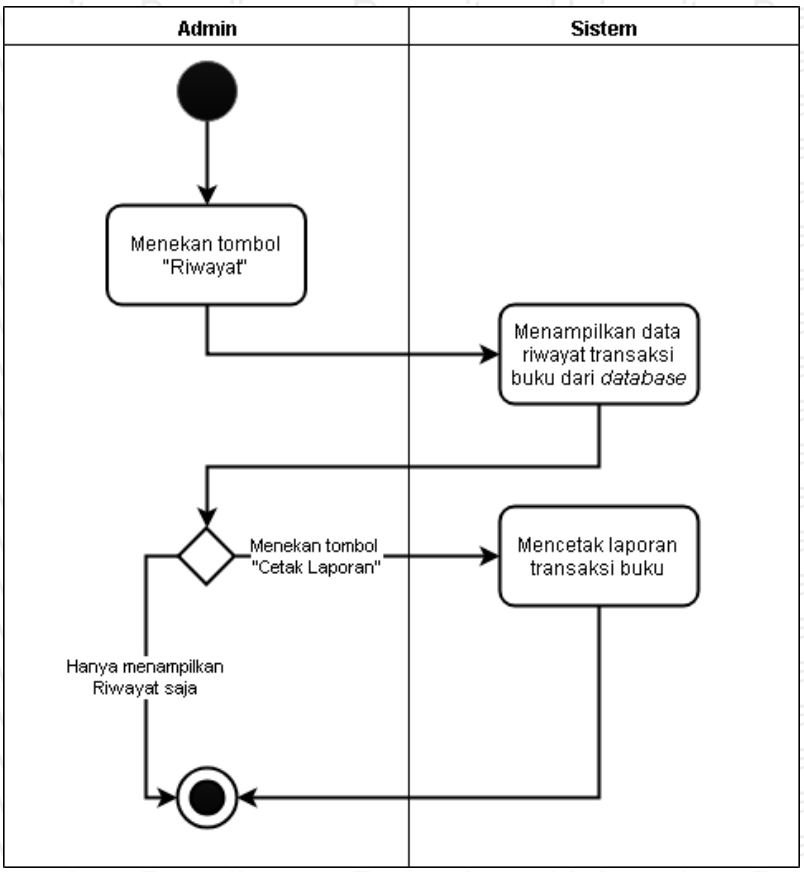
4.2.1.6 *Activity Diagram Admin* Melakukan Pengembalian Buku



Gambar 4.11 *Activity Diagram Admin* Melakukan Pengembalian Buku

Activity diagram yang tersedia pada Gambar 4.10 menjelaskan alur proses *admin* untuk melakukan transaksi pengembalian buku. Alur proses diatas setelah *admin* melakukan alur proses login. Setelah sistem melakukan menampilkan dashboard *admin. Admin* melakukan transaksi pengembalian buku dengan cara menekan tombol "Pengembalian Buku". Sistem akan menampilkan halaman pengembalian buku, *admin* memasukkan kode peminjaman yang ada pada struk peminjaman dan menekan tombol "Cari". Sistem akan menampilkan data peminjaman dengan kode yang sama. *Admin* mencocokkan data buku, apabila telah cocok maka *admin* menekan tombol "Kembalikan" untuk merubah status buku menjadi "Telah dikembalikan".

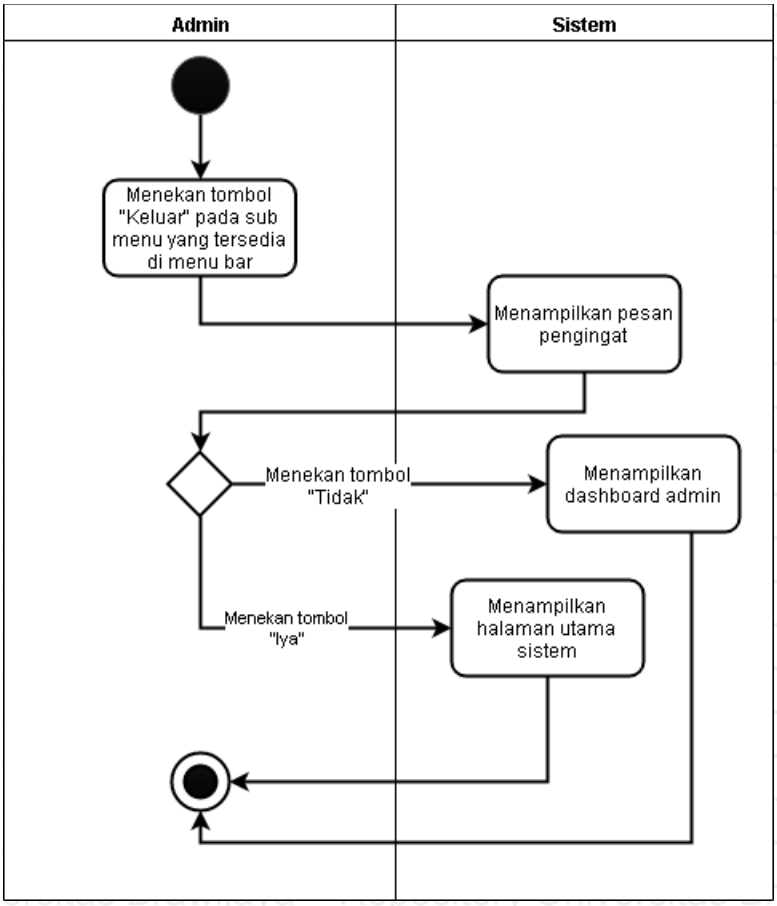
4.2.1.7 *Activity Diagram Admin* Melihat Riwayat



Gambar 4.12 *Activity Diagram Admin* Melihat Riwayat

Activity diagram yang tersedia pada Gambar 4.11 menjelaskan alur proses *admin* untuk melihat riwayat transaksi dan mencetak laporan. Alur proses diatas setelah *admin* melakukan alur proses login. Setelah sistem melakukan menampilkan dashboard *admin. Admin* menekan menu "Riwayat" pada menu bar. Sistem akan menampilkan halaman riwayat dan menampilkan seluruh riwayat transaksi peminjaman yang telah dilakukan. Untuk mencetak laporan maka *admin* diharuskan untuk menekan tombol cetak laporan dan sistem akan mencetak laporan peminjaman buku.

4.2.1.8 *Activity Diagram Admin* Melakukan Logout



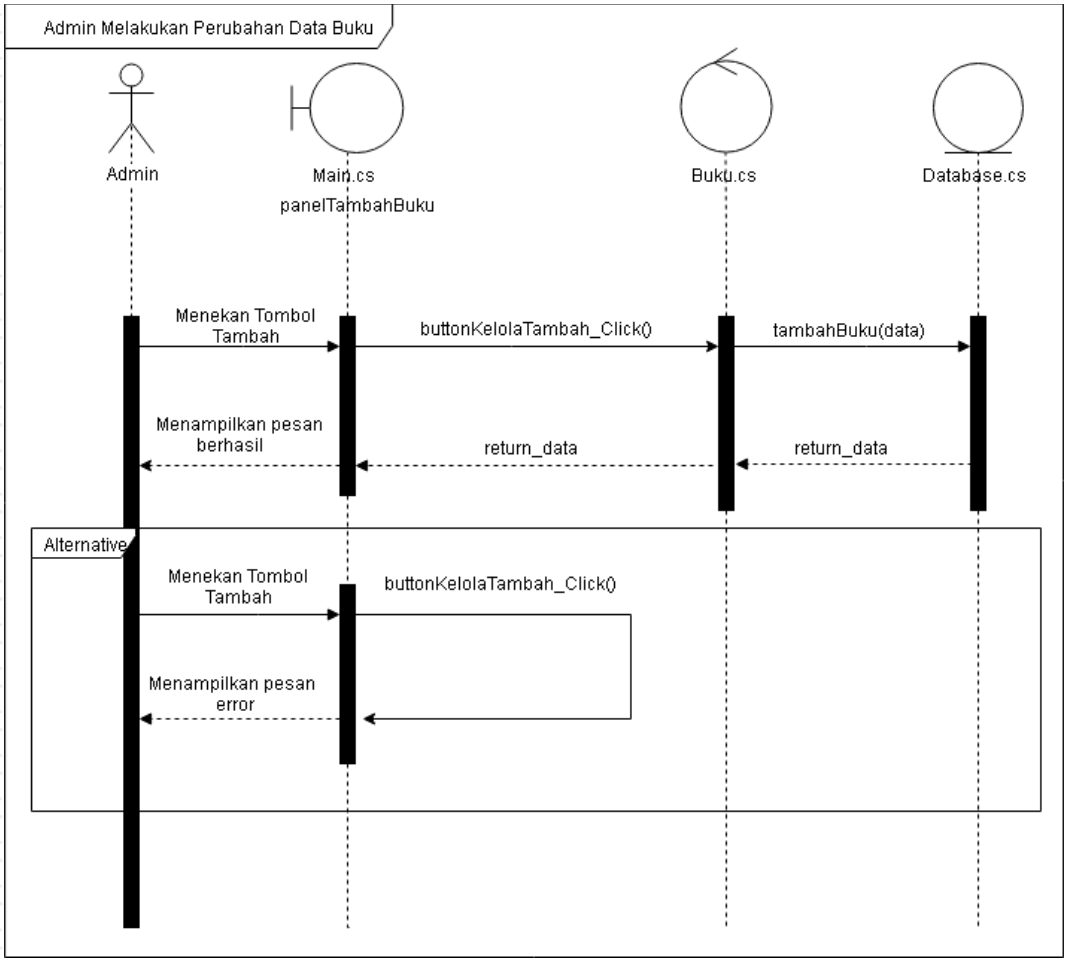
Gambar 4.13 *Activity Diagram* Admin *Melakukan Logout*

Activity diagram yang tersedia pada Gambar 4.12 menjelaskan alur proses *admin* untuk melakukan logout dari akses *admin*. Alur proses diatas setelah *admin* melakukan alur proses login. Setelah sistem melakukan menampilkan dashboard *admin. Admin* menekan tombol "*Logout*" maka sistem akan menampilkan pesan pengingat "apakah *admin* benar-benar ingin *logout*?". Apabila *admin* menekan tombol "iya" maka sistem akan menampilkan halaman utama, jika *admin* menekan tombol tidak maka sistem akan menampilkan halaman dashboard admin.

4.2.2 Sequence Diagram

Sequence Diagram berfokus pada waktu dan penunjukan urutan interaksi secara visual dengan menggunakan sumbu vertikal diagram untuk mewakili waktu serta pesan apa yang akan dikirim. Pada penelitian ini fitur -fitur yang akan digambarkan ke dalam *sequence diagram* hanya fitur-fitur penting saja yang terdiri dari, kelola buku, transaksi peminjaman sekaligus pengembalian buku dan pembuatan laporan.

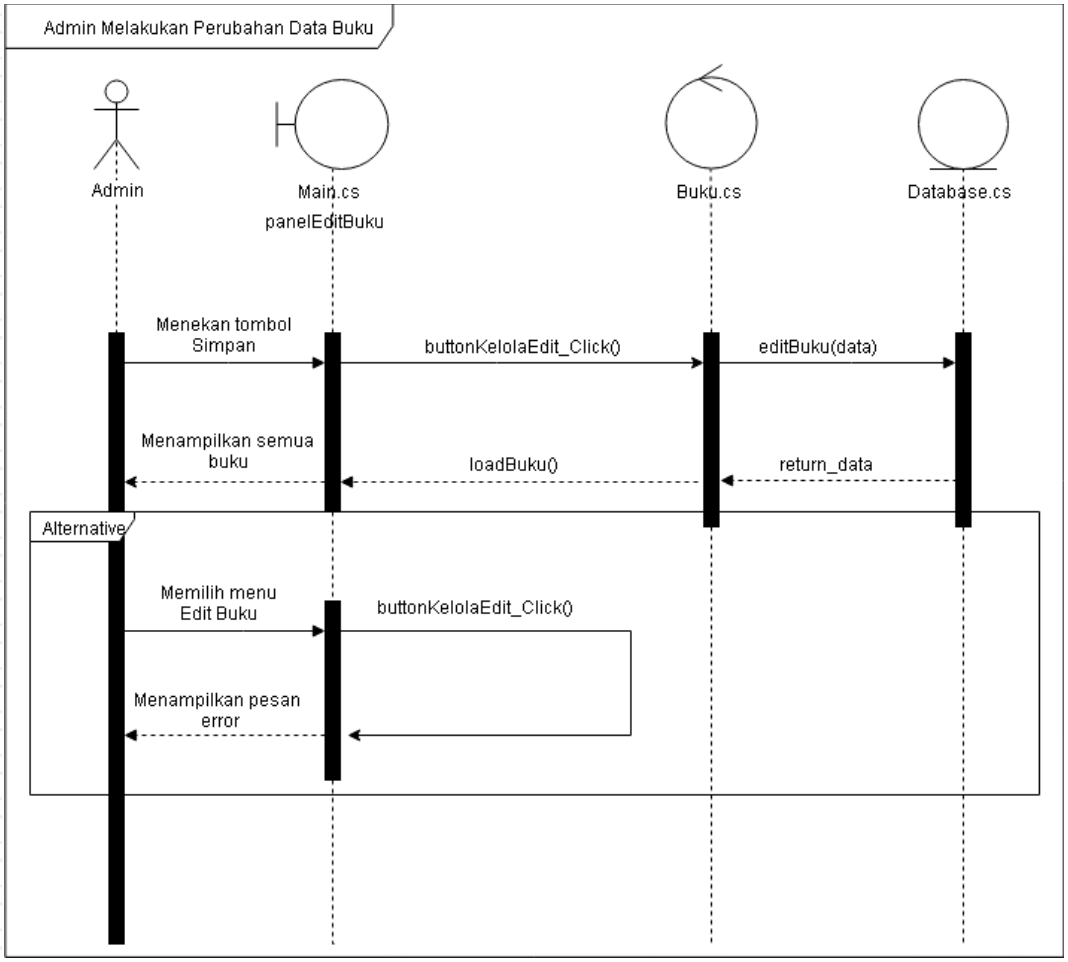
4.2.2.1 *Sequence Diagram* Tambah Buku



Gambar 4.14 *Sequence Diagram* Tambah Buku

Squence diagram pada Gambar 4.13 menjelaskan tentang proses alur *admin* melakukan penambahan buku. Dimulai dari *admin* menekan tambah buku pada menu bar kelola yang tersedia apabila telah login sebagai admin. *Admin* melakukan pengisian form yang telah disediakan panelTambahBuku pada *class* Main. Setalah data yang diisikan oleh *admin* valid, *class* Main akan mengakses *class* Buku menggunakan method buttonKelolaTambah. Dimana method buttonKelolaTambah mengambil data dan menginisialisasi data yang sudah teriisi oleh *admin* pada panelTambahBuku. *Class* Buku akan mengakses *class* Database dengan menggunakan method tambahBuku yang berisi *query* "INSERT" data yang telah diinialisasi oleh *class* Database. *Class* Database akan mereturn data success kepada *class* Buku, dan menampilkan semua daftar buku yang ada menggunakan method loadBuku. Apabila data yang diisikan *admin* pada panelTambahBuku tidak valid maka *class* buku akan mengembalikan pesan eror untuk ditampilkan pada panelTambahBuku.

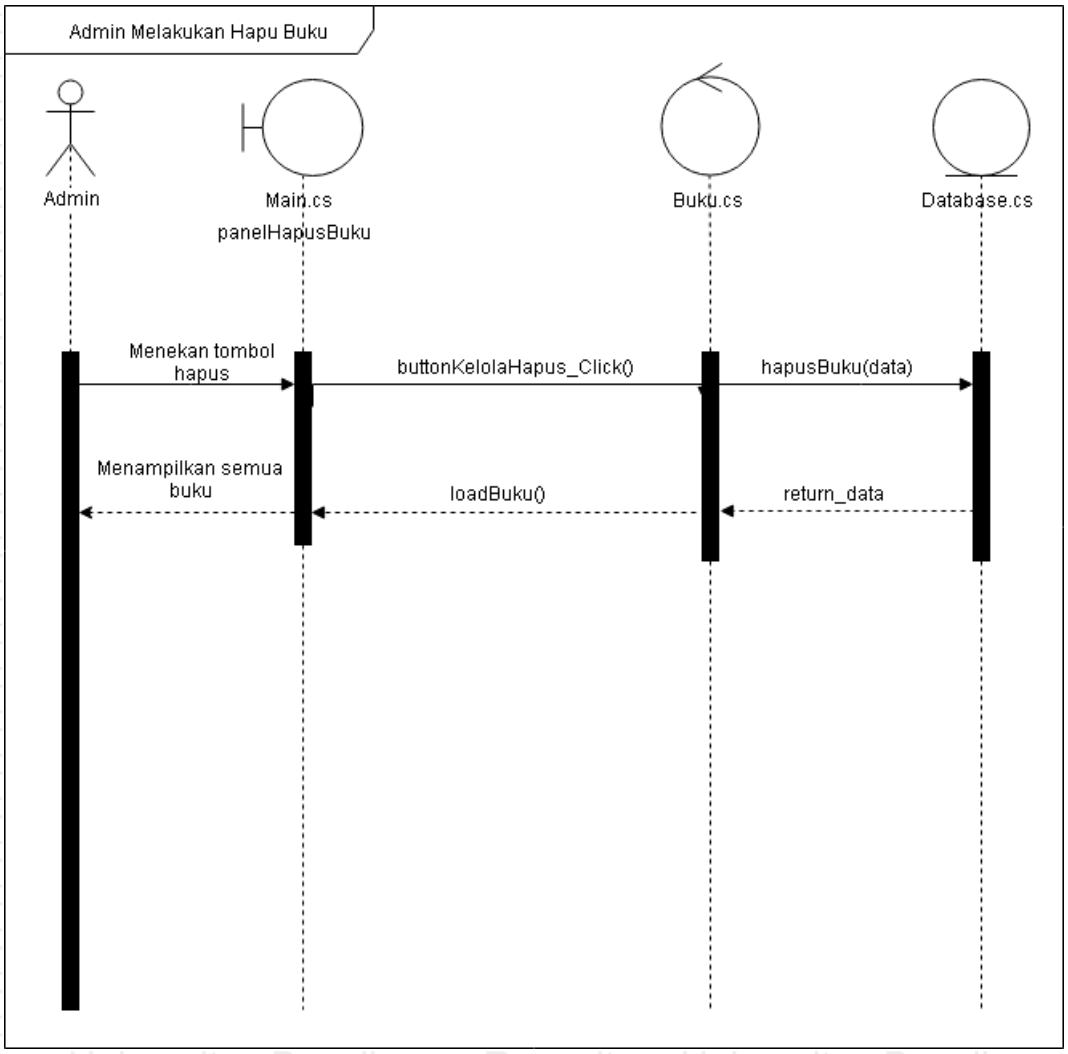
4.2.2.2 *Sequence Diagram* Edit Buku



Gambar 4.15 *Sequence Diagram* Edit Buku

Squence diagram pada Gambar 4.14 menjelaskan tentang proses alur *admin* melakukan pengeditan buku. Dimulai dari *admin* menekan menu edit buku menu bar yang disediakan. *Admin* melakukan pencarian buku untuk dilakukan pengeditan data buku. *Admin* melakukan pengeditan data yang telah dipilih untuk diedit yang ditampilkan panelEditBuku. Setalah data telah selesai diedit oleh *admin*, *class* main akan mengakses *class* buku yang berisi method editBuku. Dimana method edit_buku mengambil data dan menginisialisasi data yang sudah teriisi oleh *admin* pada panelEditBuku. *Class* Buku akan mengakses *class* Database dengan menggunakan method editBuku yang berisi *query* "UPDATE" data yang telah diinialisasi oleh class Database. *Class* database akan mereturn data success kepada *class* buku, dan class buku menggunakan method loadBuku() untuk menampilkan semua daftar buku yang ada pada panelEditBuku. Apabila data yang diisikan *admin* pada panelEditBuku tidak valid maka class buku akan mengembalikan pesan eror untuk ditampilkan pada panelEditBuku.

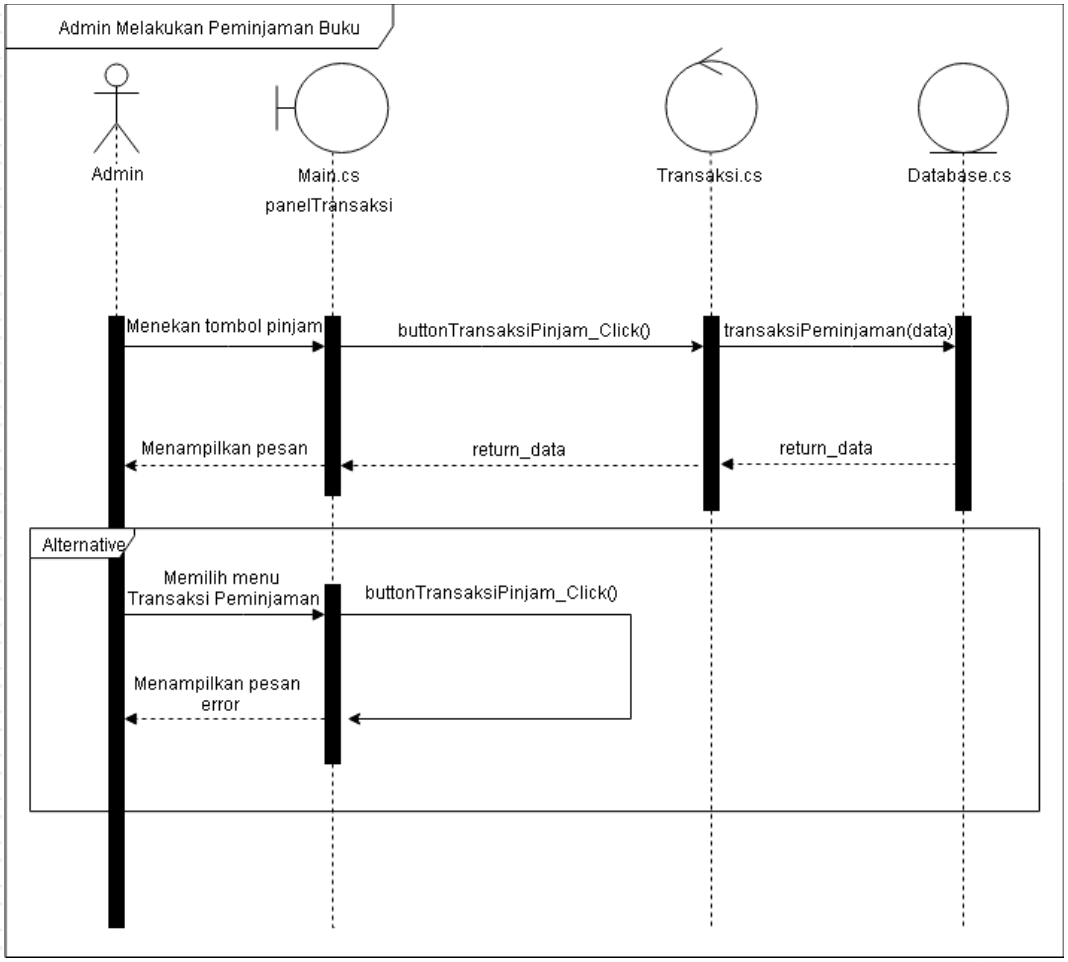
4.2.2.3 *Sequence Diagram* Hapus Buku



Gambar 4.16 *Sequence Diagram* Hapus Buku

Squence diagram pada Gambar 4.15 menjelaskan tentang proses alur *admin* melakukan penghapusan buku. Dimulai dari *admin* menekan menu hapus buku pada menu bar yang tersedia pada menu bar kelola. *Admin* melakukan pencarian buku untuk dilakukan penghapusan data buku. *Admin* melakukan penghapusan data buku yang telah dipilih untuk dihapus yang ditampilkan panelHapusBuku. *Class* main akan mengakses *class* buku menggunakan method hapusBuku. Dimana method hapusBuku mengambil data buku pada halaman *view* v_halaman_edit_buku. *Class* buku akan mengakses *class* database dengan *query* "DELETE". Database akan mereturn data success kepada *class* buku, dan dan *class* buku menampilkan semua daftar buku menggunakan method loadBuku yang akan tampil pada panelHapusBuku.

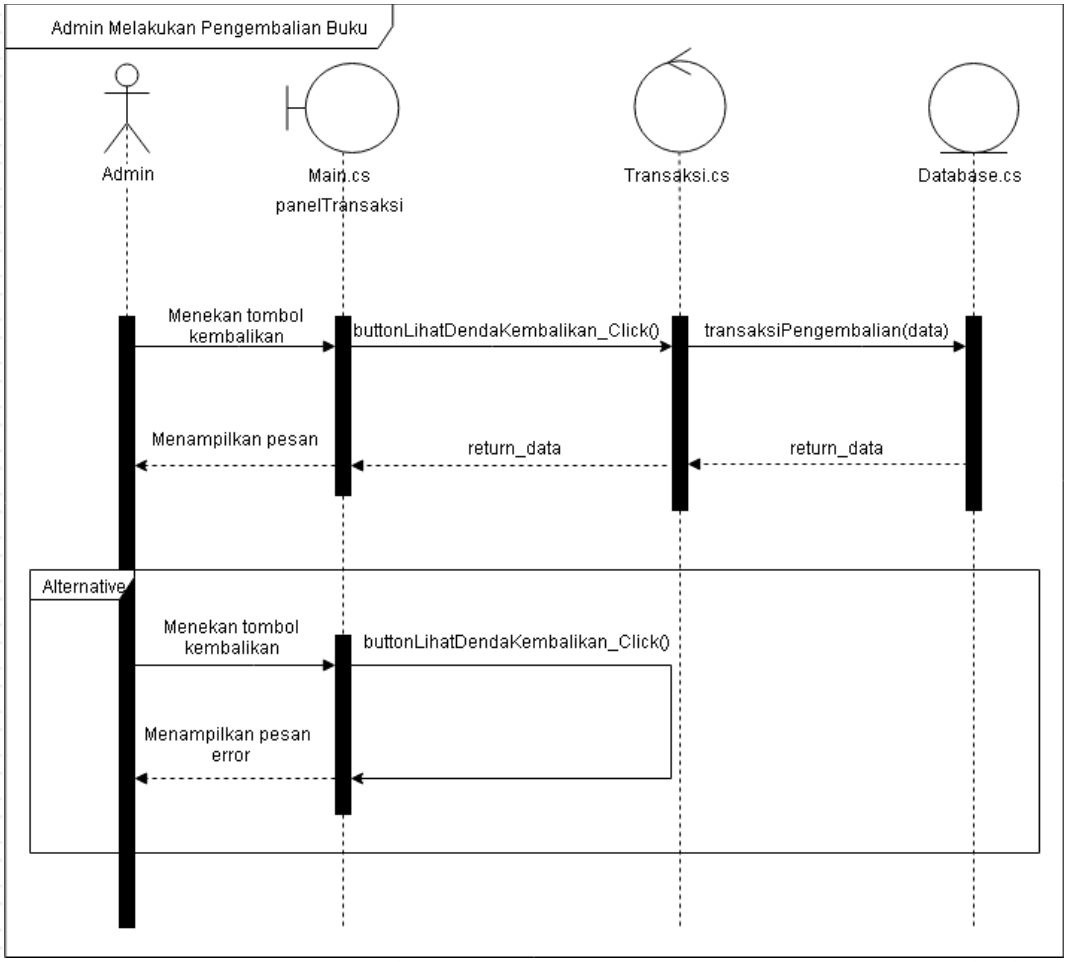
4.2.2.4 *Sequence Diagram* Transaksi Peminjaman Buku



Gambar 4.17 *Sequence Diagram* Transaksi Peminjaman Buku

Squence diagram pada Gambar 4.16 menjelaskan tentang proses alur *admin* melakukan transaksi peminjaman buku. Dimulai dari *admin* menekan menu peminjaman buku pada menu bar transaksi. *Admin* melakukan pengisian form yang telah disediakan panelTransaksi. Setalah data yang diisikan oleh *admin* valid, panelTransaksi akan mengakses *class* transaksi yang berisi method peminjaman. Dimana method peminjaman mengambil data dan menginisialisasi data yang sudah teriisi oleh *admin* pada halaman panelTransaksi. *Class* transaksi akan mengakses *class* database dengan *query* "INSERT" data yang telah diinialisasi oleh *class* transaksi. *Class* database mereturn data success kepada *class* transaksi, dan *class transaksi* akan melakukan cetak pesan melalui method messagebox untuk ditampilkan pada halaman panelTransaksi, dan *class* main akan otomatis melakukan pencetakan bukti peminjaman. Apabila data yang diisikan *admin* pada panelTransaksi tidak valid maka *class* transaksi akan mengembalikan pesan eror untuk ditampilkan pada panelTransaksi.

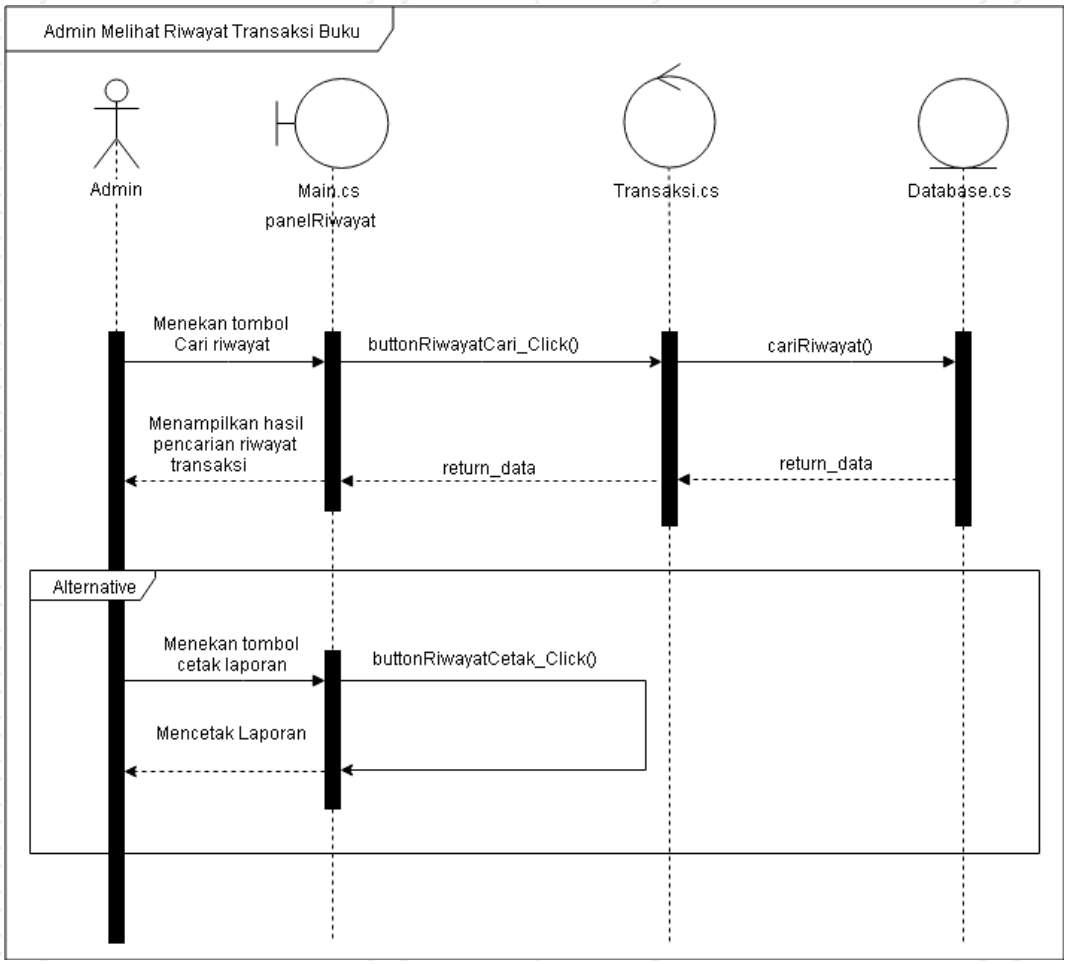
4.2.2.5 *Sequence Diagram* Transaksi Pengembalian Buku



Gambar 4.18 *Sequence Diagram* Transaksi Pengembalian Buku

Squence diagram pada Gambar 4.17 menjelaskan tentang proses alur *admin* melakukan transaksi pengembalian buku. Dimulai dari *admin* menekan menu pengembalian pada menu bar. *Admin* melakukan pengisian kode peminjaman sebagai pembeda untuk setiap transaksi. Setalah data yang diisikan oleh *admin* valid dan *admin* telah mencocokan data peminjam dan data buku, maka *class* main akan mengakses *class* transaksi menggunakn method pengembalian. Dimana method pengembalian mengambil data dan menginisialisasi data yang sudah teriisi oleh *admin* pada halaman panelTransaksi. *Class* transaksi akan mengakses class database dengan *query* "UPDATE" menggunakan method transaksiPengembalian. *Class* database akan mereturn data *success* kepada *class* transaksi, dan *class* transaksi akan melakukan cetak pesan melalui method messagebox yang ditampilkan *pada* panelTransaksi. Apabila data yang diisikan *admin* pada halaman panelTransaksi tidak valid maka *class* transaksi akan mengembalikan pesan eror untuk ditampilkan pada halaman panelTransaksi.

4.2.2.6 *Sequence Diagram* Melihat Riwayat



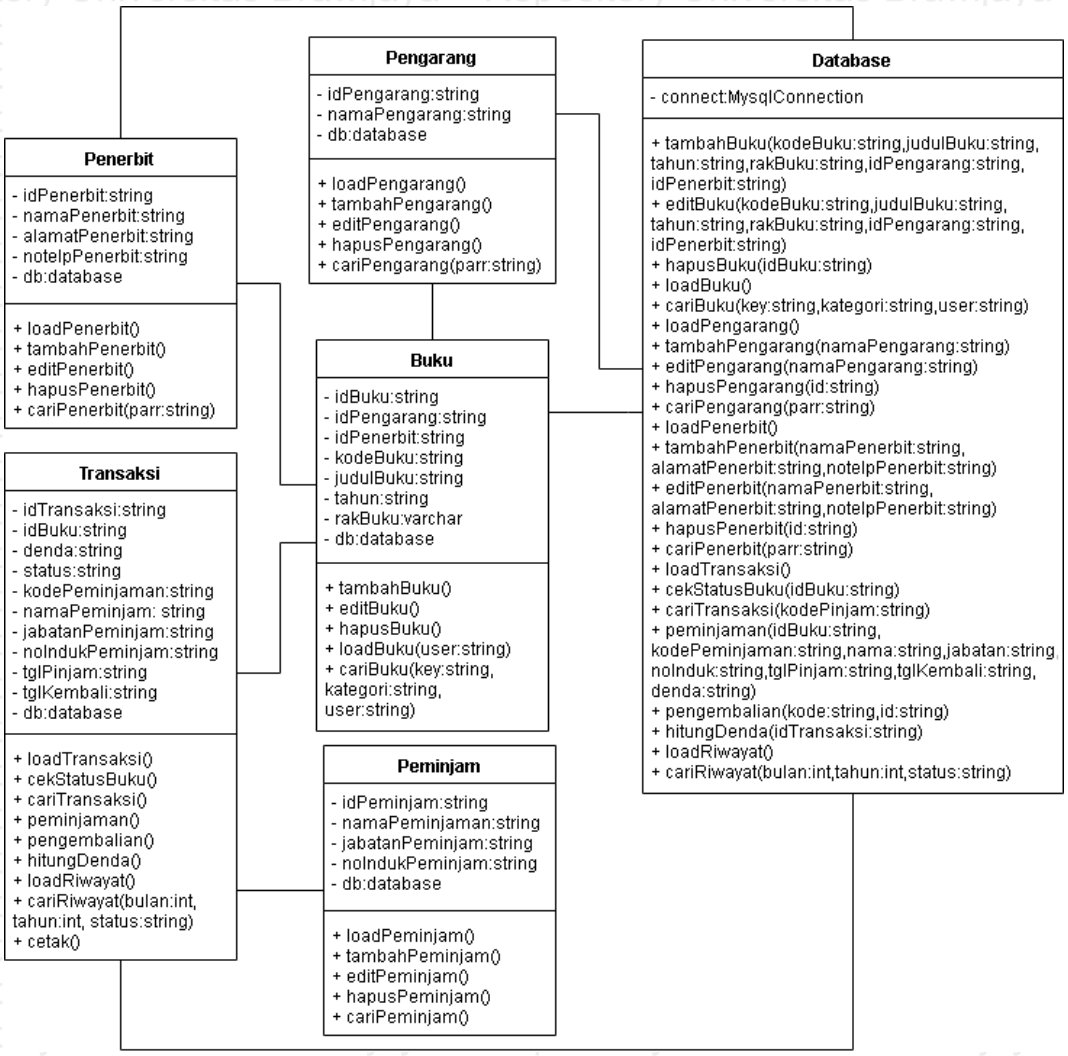
Gambar 4.19 *Sequence Diagram* Melihat Riwayat

Squence diagram pada Gambar 4.18 menjelaskan tentang proses alur *admin* melakukan lihat riwayat buku. Dimulai dari *admin* menekan menu riwayat pa menu bar. panelRiwayat pada *class* main melakukan pemanggilan method loadRiwayat pada *class* riwayat. *Class* riwayat akan melakukan pemanggilan method loadRiwayat untuk melakukan *query* "SELECT" pada *class* database. *Class* riwayat akan menerima return_data dari *class* database dan menampilkan semua data riwayat transaksi buku pada panelRiwayat di kelas main. *Alternative flow* pada *sequence diagram* Gambar 4.15 ini adalah melakukan proses pencetakan laporan. Dimulai dari *admin* menekan tombol cetak pada panelRiwayat. *Class* main akan memanggil method cetakLaporan kepada *class* Riwayat. *Class* Riwayat akan menampilkan pesan cetak dan menampilkan preview cetak laporan.

4.2.3 Class Diagram

Class diagram adalah penggambaran atau pendefinisian suatu objek berjumlah satu atau lebih dan mendefinisikan atribut yang akan digunakan oleh objek serta hubungan antara suatu objek dengan objek lainnya. Pendefinisian atribut yang akan digunakan didapatkan dari hasil wawancara kepada Ibu Esty

Hartini selaku petugas perpustakaan di SMK 1 muhammadiyah Malang ditinjau dari jawaban *stackholder* di pertanyaan ke-5 (lima) pada saat melakukan wawancara (LAMPIRAN A.1).

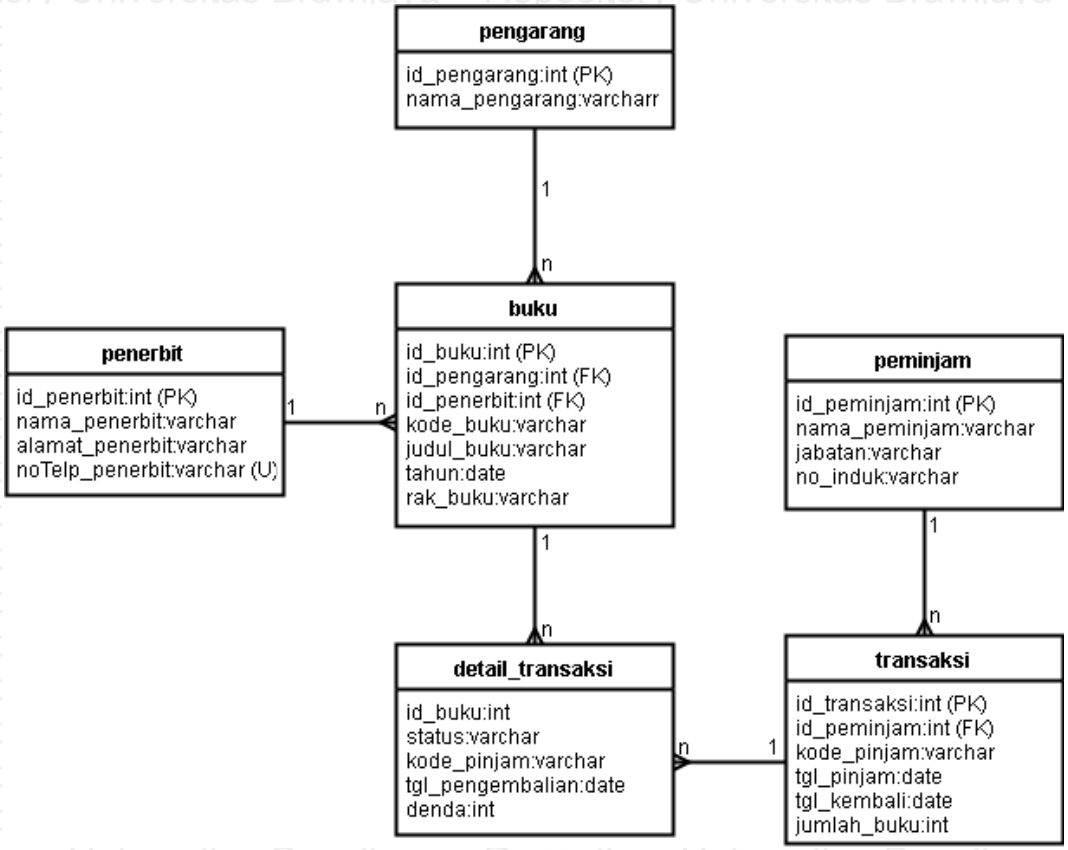


Gambar 4.20 *Class Diagram*

Class diagram yang digambarkan pada Gambar 4.19 yaitu *class diagram* yang menggambarkan relasi atau hubungan antar objek pada sistem dan atribut apa saja yang ada pada setiap objek. Pada Gambar 4.19 terlihat ada 5 (lima) kelas antara lain kelas buku, pengarang, penerbit, transaksi, dan database. Seluruh relasi yang ada pada *class diagram* adalah agregasi.

4.2.4 Perancangan *Database*

Dari *class diagram* yang telah dibuat pada sub bab 4.2.3 maka dapat dilakukan pemetaan dari *class diagram* yang telah dibuat untuk dilakukan pemetaan ke *Relational Data Model.* Pemetaan *class diagram* ke *Relational Data Model* akan digambarkan pada Gambar 4.19.



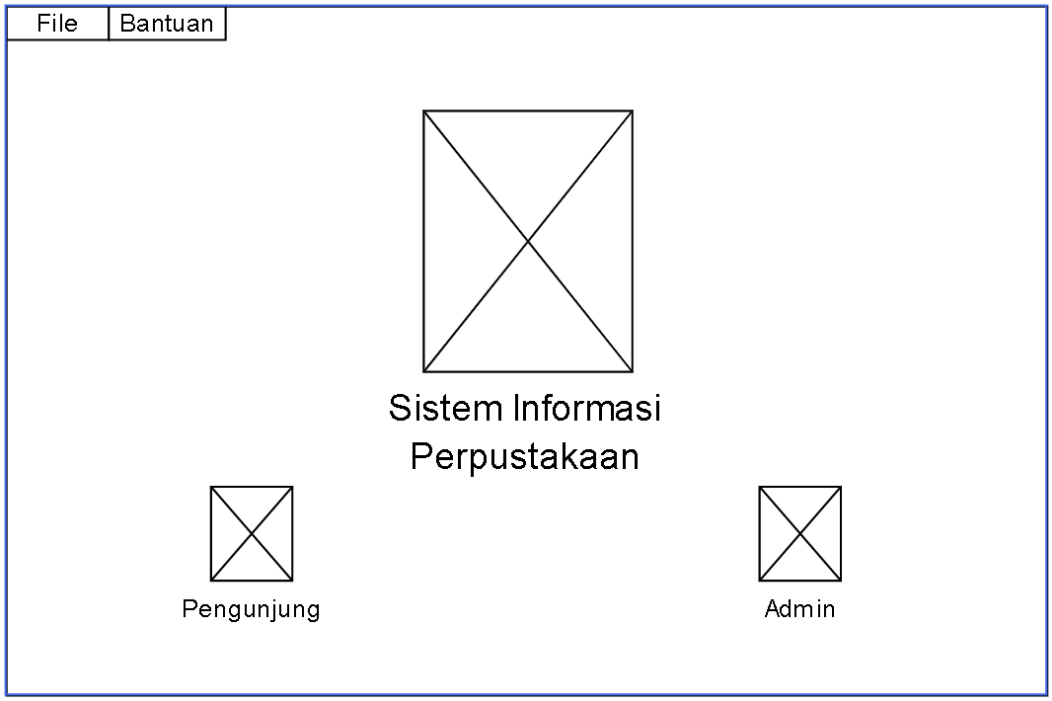
Gambar 4.21 *Relational Model Data*

Pada Gambar 4.20 menjelaskan pemetaan data model yang memiliki 5 (lima) entitas antara lain, entitas penerbit, entitas pengarang, entitas buku, entitas transaksi, dan entitas peminjam. Setiap entitas tersebut memiliki relasi yang berbeda. Relasi antara pengarang dan buku adalah One-to-Many yang berarti 1 (satu) entitas buku dapat hanya memiliki 1 (satu) entitas pengarang, dan sebaliknya entitas pengarang dapat memiliki atribut dari entitas buku. Relasi antara penerbit dan buku adalah One-to-Many yang berarti 1 (satu) entitas penerbit dapat memiliki lebih dari 1 (satu) entitas buku, dan 1 (satu) entitas buku hanya dapat memiliki 1 (satu) entitas penerbit. Relasi antara entitas buku dan entitas transaksi adalah Many-to-Many yang artinya 1 (satu) entitas buku dapat memiliki lebih dari 1 (satu) entitas transaksi dan sebaliknya 1 (satu) entitas transaksi dapat memiliki lebih dari 1 (satu) entitas buku. Relasi antara entitas transaksi dan entitas peminjam adalah One-to-Many yang berarti 1 (satu) entitas peminjam dapat memiliki lebih dari 1 (satu) entitas transaksi, dan 1 (satu) entitas transaksi hanya dapat memiliki 1 (satu) entitas peminjam.

4.2.5 Perancangan Antar Muka Sistem

Perancangan antar muka sistem yang akan dibuat adalah *wireframe. Wireframe* ini akan mempresentasikan tampilan sistem yang akan dibuat.

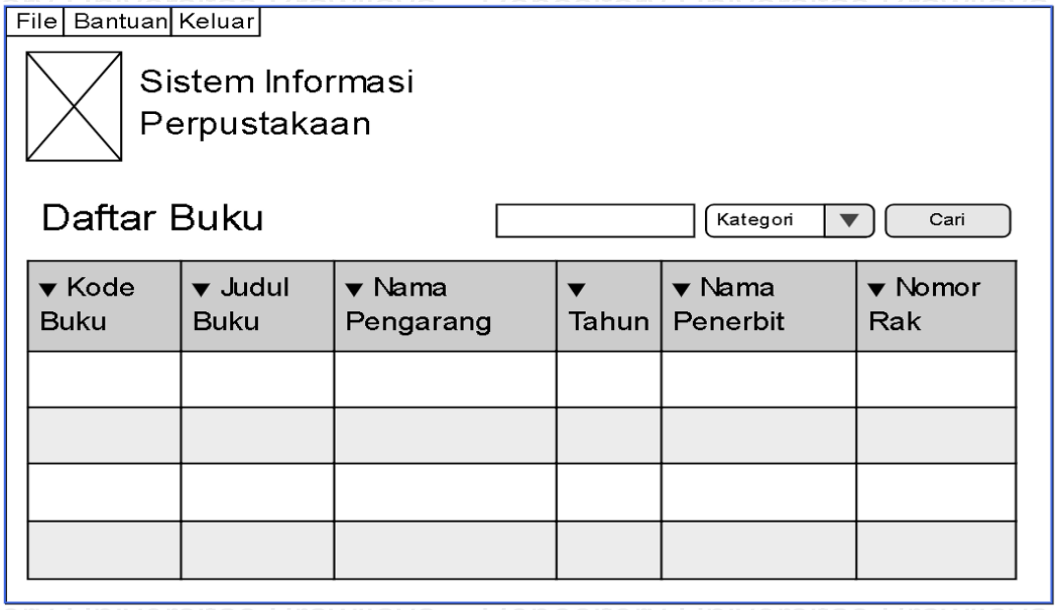
4.2.5.1 *Wireframe* Tampilan Awal Sistem



Gambar 4.22 *Wireframe* Tampilan Awal Sistem

Pada Gambar 4.21 menampilkan gambar *wireframe* tampilan awal dari sistem yang baru di buka. *User* dapat memilih masuk kedalam sistem sebagai pengunjung atau *admin* dengan cara menekan tombol pengunjung atau tombol *admin* pada tampilan awal sistem.

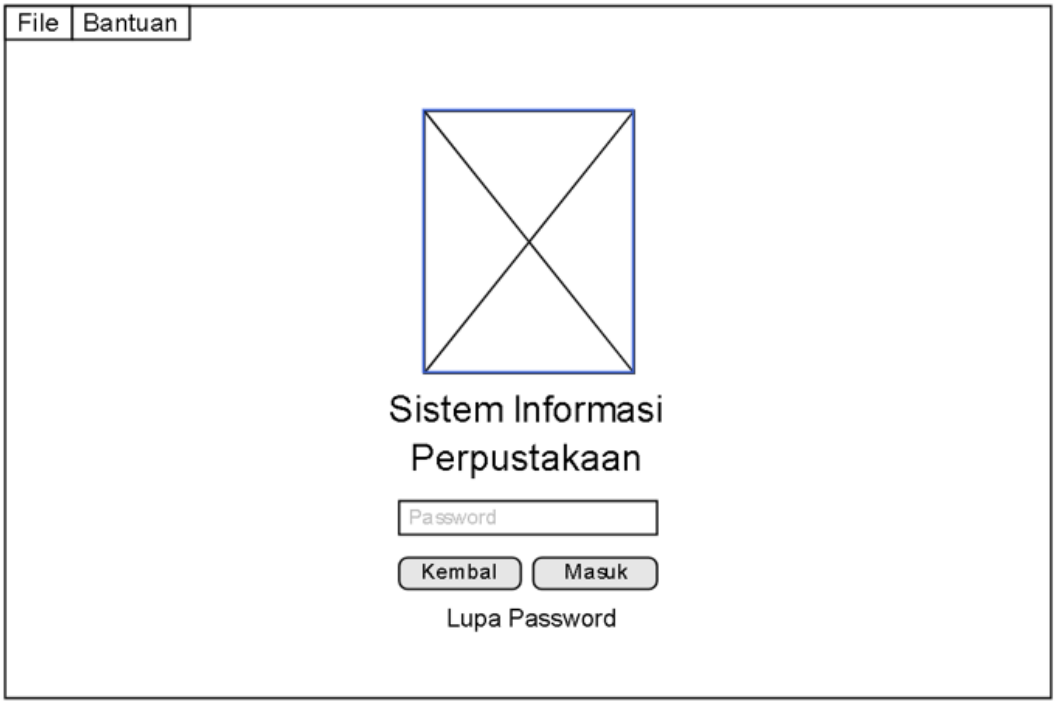
4.2.5.2 *Wireframe Dashboard* Pengunjung



Gambar 4.23 *Wireframe Dashboard* Pengunjung

Pada Gambar 4.22 menampilkan gambar *wireframe* tampilan dashboard pengunjung sistem jika *user* masuk kedalam sistem sebagai pengunjung. Pengunjung dapat mencari buku berdasarkan kategori yang dapat dipilih melalui combobox kategori, mengisi kata kunci dan meneka tombol cari. Pencarian buku akan muncul pada tabel daftar buku. Menu bar yang tersedia adalah file, bantuan, dan keluar.

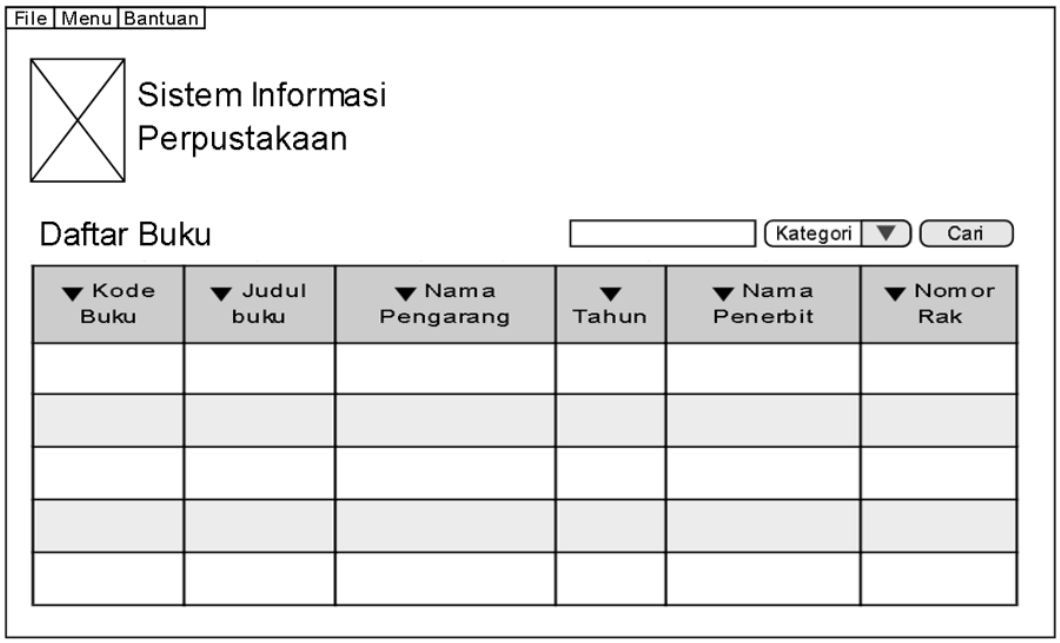
4.2.5.3 *Wireframe Login Admin*



Gambar 4.24 *Wireframe Login Admin*

Pada Gambar 4.23 menampilkan gambar *wireframe* tampilan *login admin* ketika *user* memilih masuk sebagai *admin.* Pada tampilan *wireframe login admin* terdapat *password field*, tombol kembali dan tombol masuk.

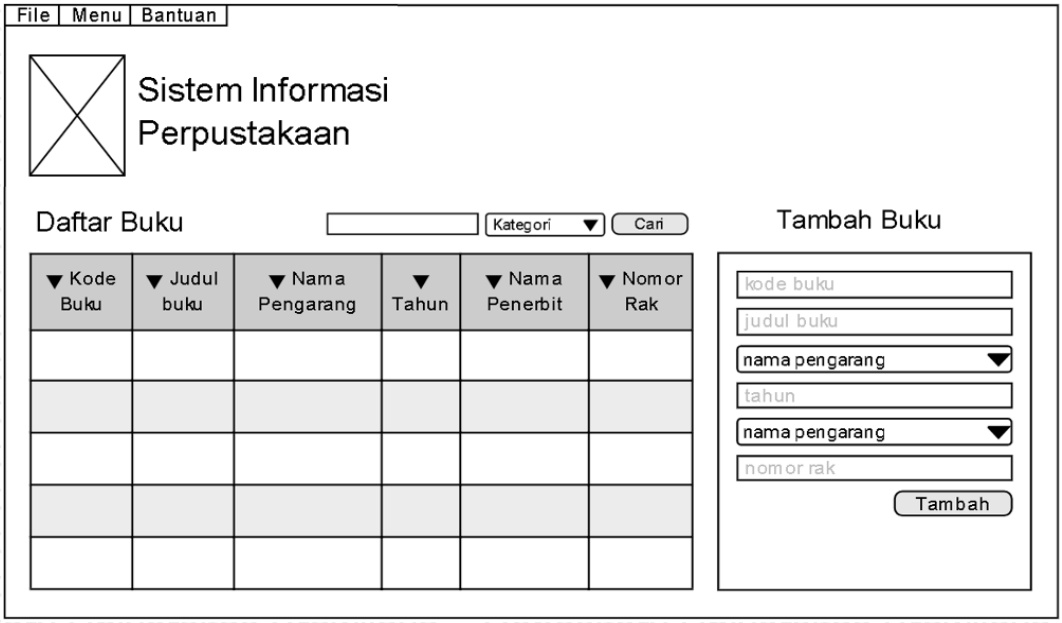
4.2.5.4 *Wireframe Admin Dashboard* Cari Buku



Gambar 4.25 *Wireframe Dashboard Admin* Cari Buku

Pada Gambar 4.24 menampilkan gambar *wireframe* tampilan *dashboard admin*. Tampilan *wireframe* ini akan tampil apabila *user* memilih masuk sebagai *admin* dan berhasil memasukkan *password*. Pada *wireframe* ini berisikan sama seperti *dashboard* pengunjung yang hanya dapat melakukan pencarian buku.

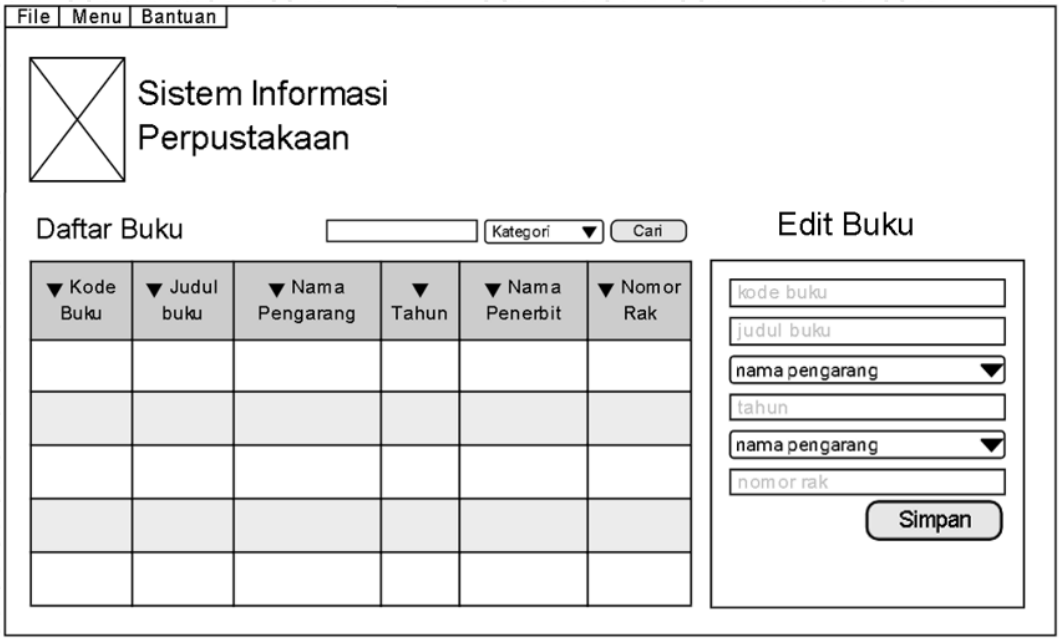
4.2.5.5 *Wireframe Admin* Tambah Buku



Gambar 4.26 *Wireframe Admin* Tambah Buku

Pada Gambar 4.25 menampilkan gambar *wireframe* tampilan *admin* tambah buku. Tampilan *wireframe* ini akan tampil apabila *admin* memilih sub menu "Kelola" dan memilih tambah buku. Pada *wireframe* ini terdapat tabel daftar buku yang dapat dilakukan pencarian dan panel tambah buku yang berisi data buku dan tombol "Tambah" untuk menambah buku pada daftar buku.

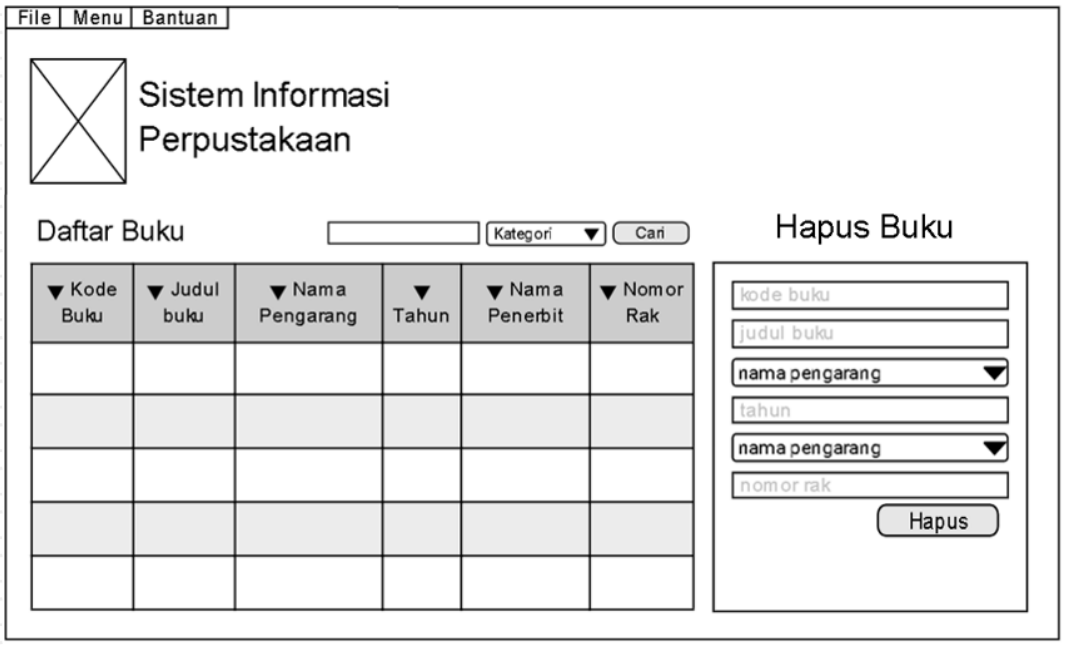
4.2.5.6 *Wireframe Admin* Edit Buku



Gambar 4.27 *Wireframe Admin* Edit Buku

Pada Gambar 4.26 menampilkan gambar *wireframe* tampilan *admin* edit buku. Tampilan *wireframe* ini akan tampil apabila *admin* memilih sub menu "Kelola" dan memilih edit buku. Pada *wireframe* ini terdapat tabel daftar buku yang dapat dilakukan pencarian dan panel edit buku yang berisi data buku dan tombol "Simpan" untuk menyimpan perubahan data buku pada daftar buku.

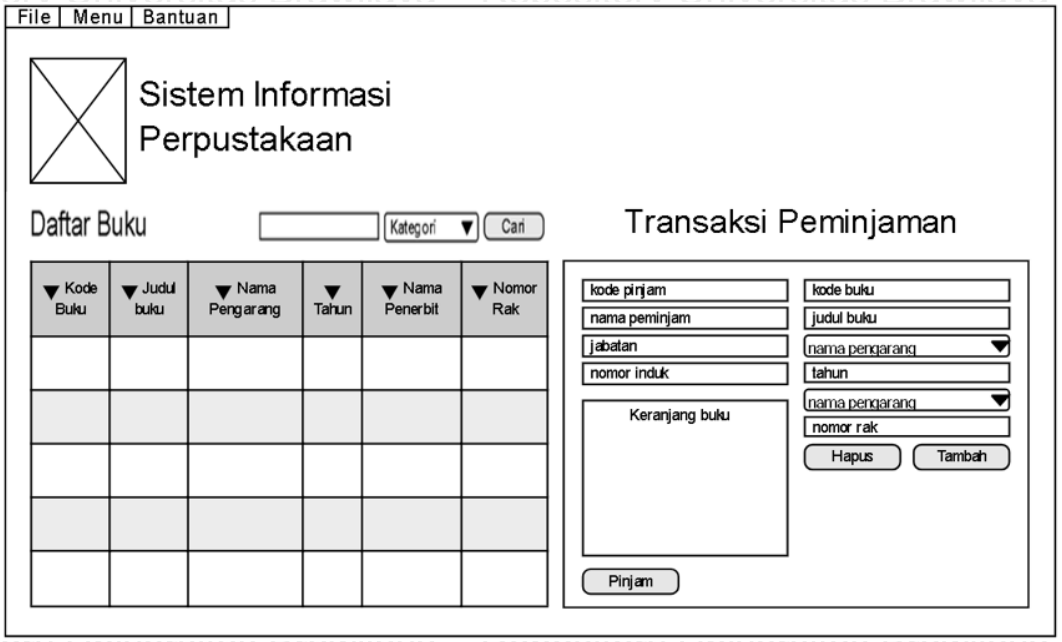
4.2.5.7 *Wireframe Admin* Hapus Buku



Gambar 4.28 *Wireframe Admin* Hapus Buku

Pada Gambar 4.27 menampilkan gambar *wireframe* tampilan *admin* hapus buku. Tampilan *wireframe* ini akan tampil apabila *admin* memilih sub menu "Kelola" dan memilih hapus buku. Pada *wireframe* ini terdapat tabel daftar buku yang dapat dilakukan pencarian dan panel hapus buku yang berisi data buku dan tombol "Hapus" untuk menghapus buku pada daftar buku.

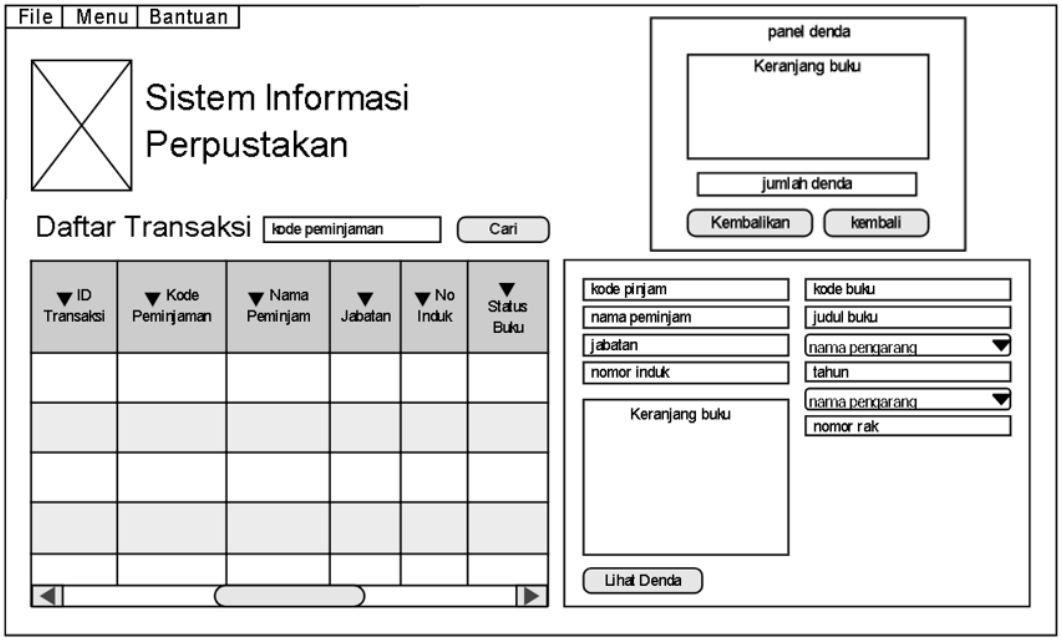
4.2.5.8 *Wireframe Admin* Transaksi Peminjaman



Gambar 4.29 *Wireframe Admin* Transaksi Peminjaman

Pada Gambar 4.28 menampilkan gambar *wireframe* tampilan *admin* melakukan transaksi peminjaman buku. Tampilan *wireframe* ini akan tampil apabila *admin* memilih sub menu "Transaksi" dan memilih peminjaman. Pada *wireframe* ini terdapat tabel daftar buku yang dapat dilakukan pencarian dan panel peminjaman buku yang berisi data buku, data peminjam buku, keranjang buku tombol tambah yang berfungsi menambahkan buku pada keranjang buku setelah memilih buku pada tabel daftar buku, tombol hapus yang berfungsi untuk menghapus buku yang ada pada keranjang buku dengan cara menekan buku yang ingin dihapus pada keranjang buku, dan tombol pinjam yang berfungsi meminjamkan buku yang ada pada keranjang buku.

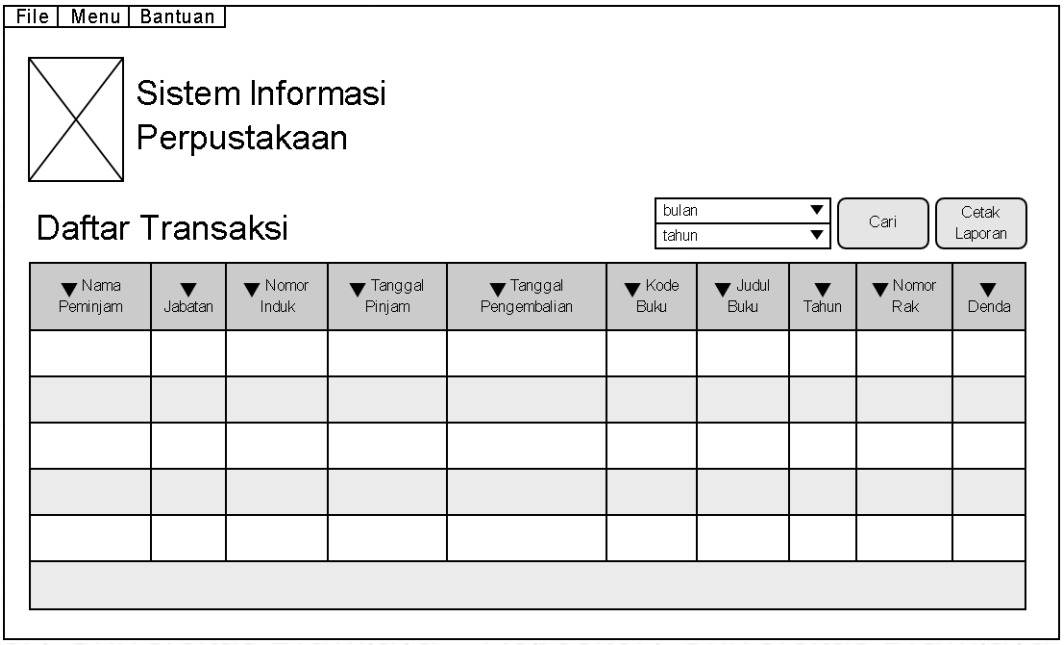
4.2.5.9 *Wireframe Admin* Transaksi Pengembalian



Gambar 4.30 *Wireframe Admin* Transaksi Pengembaian

Pada Gambar 4.29 menampilkan gambar *wireframe* tampilan *admin* melakukan transaksi peminjaman buku. Tampilan *wireframe* ini akan tampil apabila *admin* memilih sub menu "Transaksi" dan memilih pengembalian. Pada *wireframe* ini terdapat tabel daftar peminjaman buku yang dapat dilakukan pencarian, panel pemngembalian buku dan panel denda. Panel pengembalian buku yang berisi data buku, data peminjam buku, keranjang buku dan tombol lihat denda yang berfungsi untuk melihat total denda buku yang ada di keranjang buku yang akan dimunculkan pada panel denda. Panel denda berisikan keranjang buku yang berisikan daftar buku dan denda buku setiap bukunya, text box jumlah denda dari denda buku yang ada, tombol kembali, dan tombol kembalikan yang berfungsi untuk mengembalikan buku.

4.2.5.10 *Wireframe Admin* Riwayat



Gambar 4.31 *Wireframe Admin* Riwayat

Pada Gambar 4.30 menampilkan gambar *wireframe* tampilan *admin* melihat riwayat transaksi buku. Tampilan *wireframe* ini akan tampil apabila *admin* memilih sub menu "Riwayat". Pada *wireframe* ini terdapat tabel daftar transaksi buku, combo box bulan, combo box tahun, tombol cari dan tombol export data. Tombol cari berfungsi untuk melakukan pencarian transaksi berdasarkan bulan dan tahun dan tombol cetak laporan yang berfungsi untuk melakukan pencetakan data yang ada pada tabel daftar transaksi.

4.3 Implementasi (*Coding*)

Pada tahap implementasi ini, setelah melakukan tahap perencangan yang menghasilkan alur fungsional sistem yang digambarkan pada *sequence diagram* pada sub bab (4.2.2 *Squence Diagram)*. Pada sub bab (4.2.2 *Squence Diagram*) alur program akan diimplementasikan dalam bentuk *source code C-Sharp* (C#) seusai dengan rancangan *class diagram* yang telah dibuat. Pada tahap implementasi ini alur program yang akan diimplementasikan antara lain, alur program admin menambah buku, admin mengedit buku, admin menghapus buku, admin melakukan transaksi peminjaman, admin melakukan transaksi pengembalian, dan admin melihat riwayat. Pada tahap ini juga mengimplementasikan perancangan *database* yang telah dibuat.

4.3.1 Implementasi Algoritma *Admin* Menambah Buku

Implementasi algoritma *admin* melakukan penambahan buku akan menghasilkan *source code* yang terbagi dalam beberapa *class* antara lain, *class* Main, *class* Buku, dan *class* Database. *Source code* akan ditampilkan pada Tabel 4.15 – 4.17.

Tabel 4.15 Implementasi Algoritma *Admin* Menambah Buku (*Class* Main.cs)

```
1.  private void buttonKelolaTambah_Click(object sender,
2.  EventArgs e)
3.          {
4.
5.              if (textBoxKelolaKodeBuku.Text == "" ||
6.  textBoxKelolaJudul.Text == "" ||
7.  textBoxKelolaNamaPenerbit.Text == "" ||
8.  textBoxKelolaTahun.Text == "" ||
9.  textBoxKelolaNamaPengarang.Text == "" ||
10. textBoxKelolaRak.Text == "" ||
11. textBoxKelolaAlamatPenerbit.Text == "" ||
12. textBoxKelolaTelpPenerbit.Text == "")
13.             {
14.                 MessageBox.Show("Isi form yang kosong!",
15. "PESAN ERROR", MessageBoxButtons.OK,
16. MessageBoxIcon.Error);
17.             }
18.             else
19.             {
20.                 DateTime dt = DateTime.Now;
21.                 int i = 1900;
22.                 int y = dt.Year;
23.                 int xx =
24. Int32.Parse(textBoxKelolaTahun.Text);
25.                 if (xx > i && xx <= y)
26.                 {
27.                     Pengarang pg = new
28. Pengarang("",textBoxKelolaNamaPengarang.Text);
29.                     Penerbit pb = new
30. Penerbit(textBoxKelolaNamaPenerbit.Text,textBoxKelolaAlam
31. atPenerbit.Text, textBoxKelolaTelpPenerbit.Text);
32.                     temp.DataSource =
34. pg.tambahPengarang();
35.                     temp2.DataSource =
36. pb.tambahPenerbit();
37.                     textBoxidpengarang.Text =
38. temp.Rows[0].Cells[0].Value.ToString();
39.                     textBoxidpenerbit.Text =
40. temp2.Rows[0].Cells[0].Value.ToString();
41.                     if (textBoxidpengarang.Text == "-1")
42.                     {
43.                         MessageBox.Show("Nomor telepon
44. pengarang yang sama telah ada dalam sistem", "PESAN
45. ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
46.                     }
47.                     if (textBoxidpenerbit.Text=="-1")
48.                     {
49.                         MessageBox.Show("Nomor telepon
50. penerbit yang sama telah ada dalam sistem", "PESAN
51. ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
52.                     }
53.                     else
54.                     {
56.                         Buku buku = new
57. Buku(textBoxKelolaKodeBuku.Text, textBoxKelolaJudul.Text,
58. textBoxidpengarang.Text, textBoxKelolaTahun.Text,
59  textBoxidpenerbit.Text, textBoxKelolaRak.Text);
```

Tabel 4.15 Implementasi Algoritma *Admin* Menambah Buku (*Class* Main.cs) (Lanjutan)

```
60.                                                 DialogResult
61.   dialogResult = MessageBox.Show("Apakah anda yakin data
62.   yang anda isikan sudah benar?", "Pesan",
63.   MessageBoxButtons.YesNo);
64.                         if (dialogResult ==
65.   DialogResult.Yes)
67.                         {
68.                             string pesan =
69.   buku.tambahBuku();
70.                             if (pesan == "1")
71.                             {
72.                                 MessageBox.Show("Data
73.   buku tersimpan!", "Pesan", MessageBoxButtons.OK,
74.   MessageBoxIcon.Information);
75.
76.   textBoxKelolaNamaPengarang.Text = "";
78.
79.   textBoxKelolaNamaPenerbit.Text = "";
80.
81.   textBoxKelolaAlamatPenerbit.Text = "";
82.
83.   textBoxKelolaTelpPenerbit.Text = "";
84.                                 textBoxKelolaJudul.Text =
85.   "";
86.
87.   textBoxKelolaKodeBuku.Text = "";
88.
89.   comboBoxKelolaPenerbit.Items.Clear();
90.
91.   comboBoxKelolaPengarang.Items.Clear();
92.                                 textBoxKelolaRak.Text =
93.   "";
94.                                 textBoxKelolaTahun.Text =
95.   "";
96.
97.   dataGridViewPanelTambahBuku.DataSource = buku.loadBuku();
98.                             }
99.                             else if (pesan == "0")
100.                            {
101.                                MessageBox.Show("Data
102.  buku gagal tersimpan!", "PESAN ERROR",
103.  MessageBoxButtons.OK, MessageBoxIcon.Error);
103.                            }
104.                            else if (pesan == "Duplicate
105.  entry '" + textBoxKelolaKodeBuku.Text + "' for key
106.  'kode_buku'")
107.                            {
108.                                MessageBox.Show("Kode
109.  buku yang sama telah tersimpan dalam sistem", "PESAN
110.  ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
112.                            }
113.                            else
114.                            {
115.                                MessageBox.Show(pesan,
116.  "PESAN ERROR", MessageBoxButtons.OK,
117.  MessageBoxIcon.Error);
```

Tabel 4.15 Implementasi Algoritma *Admin* Menambah Buku (*Class* Main.cs) (Lanjutan)

```
116.  }
117.                          }
118.                          else if (dialogResult ==
119.  DialogResult.No)
120.                          {
121.
122.                          }
123.                      }
124.
125.
126.                  }
127.                  else
128.                  {
129.                      MessageBox.Show("Periksa kolom
130.  tahun!", "PESAN ERROR", MessageBoxButtons.OK,
131.  MessageBoxIcon.Error);
132.                  }
133.              }
134.          }
```

Pada Tabel 4.14 menampilkan *source code* fungsi tambah buku pada sistem yang ada di *class* Main.cs. Pada baris 56 hingga baris 59 menginisialisasikan data buku yang akan disimpan sebelum melakukan penyimpanan buku menggunakan method tambahBuku pada baris 69. Return data akan diinisialisasikan pada variabel pesan bertipe data *string* dan akan ditampilkan menggunakan fungsi MessageBox.

Tabel 4.16 Implementasi Algoritma *Admin* Menambah Buku (*Class* Buku.cs)

```
1.  public string tambahBuku()
2.          {
3.              return db.tambahBuku(kodeBuku, judulBuku,
4.  tahun, rakBuku, pengarang, penerbit);
5.          }
```

Pada Tabel 4.15 menampilkan *source code* fungsi tambahBuku pada sistem yang ada di *class* Buku.cs. Pada baris 1 hingga baris 5 menjelaskan method tambahBuku yang mereturn data atau mengembalikan data bervariabel *string* dari hasil proses method tambahBuku pada *class* Database.cs yang berparameter data buku yang akan disimpan.

Tabel 4.17 Implementasi Algoritma *Admin* Menambah Buku (*Class* Database.cs)

```
1.  public string tambahBuku(string kodeBuku, string
2.  judulBuku, string tahun , string rakBuku ,string
3.  idPengarang, string idPenerbit)
4.          {
5.              MySqlCommand command = con.CreateCommand();
6.              string query = "CALL tambah_buku ('"+
7.  kodeBuku + "','" + judulBuku + "','" + tahun + "','" +
8.  rakBuku + "','" + idPengarang + "','" + idPenerbit +
```

Tabel 4.17 Implementasi Algoritma *Admin* Menambah Buku (*Class* Database.cs) (Lanjutan)

```
9.      "')";
10.             con.Open();
11.             MySqlCommand cmd = new
12.     MySqlCommand(query,con);
13.             try
14.             {
15.                 if (cmd.ExecuteNonQuery() == 1)
16.                 {
17.                     con.Close();
18.                     return "1";
19.                 }
20.                 else
21.                 {
22.                     con.Close();
23.                     return "0";
24.                 }
25.             }
26.             catch (Exception e)
27.             {
28.                 con.Close();
29.                 return e.Message;
30.             }
31          }
```

Pada Tabel 4.16 menampilkan *source code* fungsi method tambahBuku pada sistem yang ada di *class* Database.cs. Pada baris 6 hingga baris 12 merupakan proses penginisialisasian *query database* dan pengeksekusian *query* pada method tambahBuku di *class* Database.cs.

4.3.2 Implementasi Algoritma *Admin* Mengedit Buku

Implementasi algoritma *admin* melakukan pengeditan data buku akan menghasilkan *source code* yang terbagi dalam beberapa *class* antara lain, *class* Main, *class* Buku, dan *class* Database. *Source code* akan ditampilkan pada Tabel 4.18 – 4.20.

Tabel 4.18 Implementasi Algoritma *Admin* Edit Buku (*Class* Main.cs)

```
1.      private void buttonKelolaEdit_Click(object sender,
2.      EventArgs e)
3.          {
4.              DateTime dt = DateTime.Now;
5.
6.              if (textBoxKelolaKodeBuku.Text == "" ||
7.      textBoxKelolaJudul.Text == "" ||
8.      textBoxKelolaNamaPenerbit.Text == "" ||
9.      textBoxKelolaTahun.Text == "" ||
10.     textBoxKelolaNamaPengarang.Text == "" ||
11.     textBoxKelolaRak.Text == "" ||
12.     textBoxKelolaAlamatPenerbit.Text == ""||
13.     textBoxKelolaTelpPenerbit.Text == "")
14.             {
15.                 MessageBox.Show("Isi form yang kosong!",
16.     "PESAN ERROR", MessageBoxButtons.OK,
```

Tabel 4.18 Implementasi Algoritma *Admin* Edit Buku (*Class* Main.cs) (Lanjutan)

```
17.  MessageBoxIcon.Error);
18.  }
19.              else
20.              {
21.                  int i = 1900;
22.                  int y = dt.Year;
23.                  int xx =
24.  Int32.Parse(textBoxKelolaTahun.Text);
25.                  if (xx > i && xx <= y)
26.                  {
27.
28.                      DialogResult dialogResult =
29.  MessageBox.Show("Apakah anda yakin data yang anda isikan
30.  sudah benar?", "PESAN INFORMASI",
31.  MessageBoxButtons.YesNo, MessageBoxIcon.Information);
32.                      if (dialogResult == DialogResult.Yes)
34.                      {
35.                          Pengarang pg = new
36.  Pengarang(textBoxKelolaIdBuku.Text,
37.  textBoxKelolaNamaPengarang.Text);
38.                          Penerbit pb = new
39.  Penerbit(textBoxKelolaIdBuku.Text,
40.  textBoxKelolaNamaPenerbit.Text,
41.  textBoxKelolaAlamatPenerbit.Text,
42.  textBoxKelolaTelpPenerbit.Text);
43.                          temp.DataSource =
44.  pg.editPengarang();
45.                          temp2.DataSource =
46.  pb.editPenerbit();
47.                          if
48.  (temp.Rows[0].Cells[0].Value.ToString() =="-1")
49.                          {
50.                              MessageBox.Show("Nomor
51.  telepon pengarang yang sama telah ada dalam sistem",
52.  "PESAN ERROR", MessageBoxButtons.OK,
53.  MessageBoxIcon.Error);
54.                          }
56.
57.  if(temp2.Rows[0].Cells[0].Value.ToString() == "-1")
58.                          {
59.                              MessageBox.Show("Nomor
60.  telepon penerbit yang sama telah ada dalam sistem",
61.  "PESAN ERROR", MessageBoxButtons.OK,
62.  MessageBoxIcon.Error);
63.                          }
64.                          else
65.                          {
67.                              Buku buku = new
68.  Buku(textBoxKelolaIdBuku.Text,
69.  textBoxKelolaKodeBuku.Text, textBoxKelolaJudul.Text,
70.  temp.Rows[0].Cells[0].Value.ToString(),
71.  textBoxKelolaTahun.Text,
72.  temp2.Rows[0].Cells[0].Value.ToString(),
73.  textBoxKelolaRak.Text);
```

Tabel 4.18 Implementasi Algoritma *Admin* Edit Buku (*Class* Main.cs) (Lanjutan)

```
75.  string pesan = buku.editBuku();
76.                             if (pesan == "1")
77.                             {
78.  MessageBox.Show("Data buku tersimpan!", "PESAN
81.  INFORMASI", MessageBoxButtons.OK,
82.  MessageBoxIcon.Information);
83.  textBoxKelolaNamaPengarang.Text = "";
84.  textBoxKelolaNamaPenerbit.Text = "";
85.  textBoxKelolaAlamatPenerbit.Text = "";
86.  textBoxKelolaTelpPenerbit.Text = "";
87.                                 textBoxKelolaJudul.Text =
88.  "";
89.
90.  textBoxKelolaKodeBuku.Text = "";
91.
92.  comboBoxKelolaPenerbit.Items.Clear();
93.
94.  comboBoxKelolaPengarang.Items.Clear();
95.                                 textBoxKelolaRak.Text =
96.  "";
97.                                 textBoxKelolaTahun.Text =
98.  "";
99.
100. dataGridViewPanelTambahBuku.DataSource = buku.loadBuku();
101.                            }
102.                            else if (pesan == "0")
103.                            {
103.                                MessageBox.Show("Data
104. buku gagal tersimpan!", "PESAN ERROR",
105. MessageBoxButtons.OK, MessageBoxIcon.Error);
106.                            }
107.                            else if (pesan == "Duplicate
108. entry '" + textBoxKelolaKodeBuku.Text + "' for key
109. 'kode_buku'")
110.                            {
111. MessageBox.Show("Kode buku yang sama telah tersimpan
112. dalam sistem", "PESAN ERROR", MessageBoxButtons.OK,
113. MessageBoxIcon.Error); } else{
114.                                MessageBox.Show(pesan,
115. "PESAN ERROR", MessageBoxButtons.OK,
116. MessageBoxIcon.Error);
117.                            }
118.                        }
119.                    }
120.                    else if (dialogResult ==
121. DialogResult.No)
122.                    {
123.                    }
124.                }else
125.                {
126.                    MessageBox.Show("Periksa kolom
127. tahun!", "PESAN ERROR", MessageBoxButtons.OK,
128. MessageBoxIcon.Error);
129.                }
130.            }
131.        }
```

Tabel 4.18 Implementasi Algoritma *Admin* Edit Buku (*Class* Main.cs) (Lanjutan)

```
132.        {
133.                        MessageBox.Show("Periksa kolom
134. tahun!", "Error", MessageBoxButtons.OK);
135.                    }
136.                }
137.            }
```

Pada Tabel 4.17 menampilkan *source code* fungsi edit buku pada sistem yang ada di *class* Main.cs. Pada baris 68 hingga baris 73 merupakan proses penginisialisasian data buku untuk dilakukan perubahan data buku. Pada baris 76 adalah proses merubah data buku dengan data yang telah diinisialisasikan pada baris 50 hingga baris 56 menggunakan method editBuku yang ada pada *class* Buku.cs.

Tabel 4.19 Implementasi Algoritma *Admin* Edit Buku (*Class* Buku.cs)

```
1.  public string editBuku()
2.          {
3.              return db.editBuku(idBuku, kodeBuku,
4.  judulBuku, tahun, rakBuku,pengarang, penerbit) ;
5.          }
```

Pada Tabel 4.18 menampilkan *source code* fungsi edit buku pada sistem yang ada di *class* Buku.cs. Pada baris 1 hingga baris 5 menjelaskan method editBuku yang mereturn data atau mengembalikan data bervariabel *string* dari hasil proses method editBuku pada *class* Database.cs yang berparameter data buku yang akan disimpan.

Tabel 4.20 Implementasi Algoritma *Admin* Edit Buku (*Class* Database.cs)

```
1.  public string editBuku(string idBuku, string kodeBuku,
2.  string judulBuku, string tahun, string rakBuku, string
3.  pengarang, string penerbit){
4.              MySqlCommand command = con.CreateCommand();
5.              string query = "CALL edit_buku ('" + idBuku +
6.  "','" + kodeBuku + "','" + judulBuku + "','" + tahun +
7.  "','" + rakBuku + "','" + pengarang + "','" + penerbit +
8.  "')";
9.              con.Open();
10. MySqlCommand cmd = new MySqlCommand(query, con);
11. try{
12.             if (cmd.ExecuteNonQuery() == 1){
13.                 con.Close();
14.                 return "1";
15.             }else
16.             {
17.                 con.Close();
18.                 return "0";
19.             }
20.         }
21. catch (Exception e){
22.             con.Close();
23.             return e.Message;
24.         }
25.     }
```

Pada Tabel 4.19 menampilkan *source code* fungsi method editBuku pada sistem yang ada di *class* Database.cs. Pada baris 5 hingga baris 12 merupakan proses penginisialisasian *query database* dan pengeksekusian *query* pada method editBuku di *class* Database.cs.

4.3.3 Implementasi Algoritma *Admin* Menghapus Buku

Implementasi algoritma *admin* melakukan penghapusan data buku akan menghasilkan *source code* yang terbagi dalam beberapa *class* antara lain, *class* Main, *class* Buku, dan *class* Database. *Source code* akan ditampilkan pada Tabel 4.21 – 4.23.

Tabel 4.21 Implementasi Algoritma *Admin* Hapus Buku (*Class* Main.cs)

```
1.  private void buttonKelolaHapus_Click(object sender,
2.  EventArgs e) {
3.              Buku buku = new
4.  Buku(textBoxKelolaIdBuku.Text);
5.              if(textBoxKelolaKodeBuku.Text == ""){
6.                  MessageBox.Show("Silahkan pilih buku
7.  terlebih dahulu!", "PESAN INFORMASI",
8.  MessageBoxButtons.OK, MessageBoxIcon.Information);
9.              }else{
10.                 DialogResult dialogResult =
11. MessageBox.Show("Apakah anda yakin akan menghapus data
12. buku yang telah anda pilih?", "PESAN INFORMASI",
13. MessageBoxButtons.YesNo, MessageBoxIcon.Question);
14.                 if (dialogResult == DialogResult.Yes) {
15.                     temp.DataSource = buku.hapusBuku();
16.     if (temp.Rows[0].Cells[0].Value.ToString() == "0"){
17. MessageBox.Show("Data buku terhapus!", "PESAN INFORMASI",
18. MessageBoxButtons.OK, MessageBoxIcon.Information);
19.             textBoxKelolaNamaPengarang.Text = "";
20.             textBoxKelolaNamaPenerbit.Text = "";
21.             textBoxKelolaAlamatPenerbit.Text = "";
22.             textBoxKelolaTelpPenerbit.Text = "";
23.             textBoxKelolaJudul.Text = "";
24.             textBoxKelolaKodeBuku.Text = "";
25. comboBoxKelolaPenerbit.Items.Clear();
26. comboBoxKelolaPengarang.Items.Clear();
27.                         textBoxKelolaRak.Text = "";
28.                         textBoxKelolaTahun.Text = "";
29. dataGridViewPanelTambahBuku.DataSource = buku.loadBuku();
30.                     }else if
31. (temp.Rows[0].Cells[0].Value.ToString() == "-1"){
32. MessageBox.Show("Tidak dapat menghapus data buku
34. dikarenakan data buku diperlukan pada laporan
35. transaksi!", "PESAN ERROR", MessageBoxButtons.OK,
36. MessageBoxIcon.Error);
37.                 }
38.             }else if (dialogResult == DialogResult.No)
39.                 {
40.                 }
41.             }
42.         }
```

Pada Tabel 4.20 menampilkan *source code* fungsi hapus buku pada sistem yang ada di *class* Main.cs. Pada baris 3 hingga baris 4 merupakan proses

penginisialisasian data buku untuk dilakukan penghapusan data buku. Pada baris 15 adalah proses penghapusan data buku dengan data yang telah diinisialisasikan pada baris menggunakan method hapusBuku yang ada pada *class* Buku.cs.

Tabel 4.22 Implementasi Algoritma *Admin* Hapus Buku (*Class* Buku.cs)

```
1.  public BindingSource hapusBuku()
2.          {
3.              return db.hapusBuku(idBuku);
4.          }
```

Pada Tabel 4.21 menampilkan *source code* fungsi hapus buku pada sistem yang ada di *class* Buku.cs. Pada baris 1 hingga baris 8 menjelaskan method editBuku yang mereturn data atau mengembalikan data bervariabel *string* dari hasil proses method editBuku pada *class* Database.cs yang berparameter idBuku yang akan dihapus.

Tabel 4.23 Implementasi Algoritma *Admin* Hapus Buku (*Class* Database.cs)

```
1.  public BindingSource hapusBuku(string idBuku)
2.          {
3.              con.Open();
4.              BindingSource bs = new BindingSource();
5.
6.              MySqlDataAdapter adapter = new
7.  MySqlDataAdapter();
8.
9.              MySqlCommand cmd;
10.
11.             DataSet ds = new DataSet();
12.
13.             string sql = "CALL hapus_buku('" + idBuku +
14. "')";
15.             cmd = new MySqlCommand(sql, con);
16.
17.             adapter.SelectCommand = cmd;
18.
19.             adapter.Fill(ds);
20.
21.             bs.DataSource = ds.Tables[0];
22.
23.             con.Close();
24.
25.             return bs;
26.
27.         }
```

Pada Tabel 4.22 menampilkan *source code* fungsi method hapusBuku pada sistem yang ada di *class* Database.cs. Pada baris 13 hingga baris 17 merupakan proses penginisialisasian *query database* dan pengeksekusian *query* pada method hapusBuku di *class* Database.cs.

4.3.4 Implementasi Algoritma *Admin* Peminjaman Buku

Implementasi algoritma *admin* melakukan peminjaman buku akan menghasilkan *source code* yang terbagi dalam beberapa *class* antara lain, *class*

Main, *class* Buku, dan *class* Database. *Source code* akan ditampilkan pada Tabel 4.24 – 4.26.

Tabel 4.24 Implementasi Algoritma *Admin* Peminjaman Buku (*Class* Main.cs)

```
private void buttonTransaksiPinjam_Click(object sender,
EventArgs e)
        {
            textBoxTransaksiJabatan.Text =
comboBoxTransaksiJabatan.Text;
            if (listViewKeranjang.Items.Count == 0)
            {
                MessageBox.Show("Buku dalam keranjang
kosong!", "PESAN ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
            else
            {
                if (textBoxTransaksiNama.Text == "" ||
textBoxTransaksiNoInduk.Text == "" ||
textBoxTransaksiJabatan.Text == "")
                {
                    MessageBox.Show("Lengkapi data
peminjam!", "PESAN ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                }
                else
                {

                    DialogResult dialogResult =
MessageBox.Show("Apakah anda yakin data yang anda isikan
sudah benar?", "PESAN INFORMASI",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);
                    if (dialogResult == DialogResult.Yes)
                    {
                        string pesan = "";
                        int cek = 0;
                        foreach (ListViewItem item in
listViewKeranjang.Items)
                        {
                            var pinjam =
DateTime.Now.Date;
                            var kembali =
pinjam.AddDays(2);
                            Transaksi transaksi = new
Transaksi(item.Text, textBoxTransaksiKode.Text,
textBoxTransaksiNama.Text, textBoxTransaksiJabatan.Text,
textBoxTransaksiNoInduk.Text,
pinjam.ToString("yy/MM/dd"),
kembali.ToString("yy/MM/dd"), "");
                            temp.DataSource =
transaksi.cekStatusBuku();
                            if
(temp.Rows[0].Cells[0].Value.ToString() == "1")
```

Table 4.24 Implementasi Algoritma *Admin* Peminjaman Buku (*Class* Main.cs) (Lanjutan)

```
{
                                MessageBox.Show("Buku
berjudul '" + item.SubItems[1].Text + "' masih dalam
peminjaman", "PESAN INFORMASI", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                                cek++;
                            }

                        }
                        if (cek == 0)
                        {
                            foreach (ListViewItem item in
listViewKeranjang.Items)
                            {
                                var pinjam =
DateTime.Now.Date;
                                var kembali =
pinjam.AddDays(2);
                                Transaksi transaksi = new
Transaksi(item.Text, textBoxTransaksiKode.Text,
textBoxTransaksiNama.Text, textBoxTransaksiJabatan.Text,
textBoxTransaksiNoInduk.Text,
pinjam.ToString("yy/MM/dd"),
kembali.ToString("yy/MM/dd"), "0");
                                pesan =
transaksi.peminjaman();
                            }
                            if (pesan == "1")
                            {
                                DialogResult dr =
MessageBox.Show("Data transaksi tersimpan! Tekan OK untuk
mencetak bukti peminjaman.", "PESAN INFORMASI",
MessageBoxButtons.OK, MessageBoxIcon.Information);
if (dr == DialogResult.OK){
            Database db = new Database();
            temp.DataSource =
db.cetakBuktiPeminjaman(textBoxTransaksiKode.Text);
            DGVPrinter printer = new DGVPrinter();
                                printer.Title =
"Perpustakaan SMK 1 Muhammadiyah Malang\n
Bukti Peminjaman Buku";
                                printer.SubTitle =
"\nKode                : "+ textBoxTransaksiKode.Text +
"\nNama              : " + textBoxTransaksiNama.Text +
"\nJabatan           : " + textBoxTransaksiJabatan.Text +
"\nNomor Induk : " + textBoxTransaksiNoInduk.Text;

printer.SubTitleSpacing = 10;
                                printer.TitleSpacing
= 10;
printer.TitleFont = new Font("Tahoma", 9, FontStyle.Bold,
GraphicsUnit.Point);
                                printer.SubTitleFont
= new Font("Tahoma", 8, FontStyle.Bold,
GraphicsUnit.Point);
printer.TitleAlignment = StringAlignment.Near;
```

Table 4.24 Implementasi Algoritma *Admin* Peminjaman Buku (*Class* Main.cs) (Lanjutan)

```
109.  printer.SubTitleAlignment = StringAlignment.Near;
110.  printer.PrintPreviewDataGridView(temp);
111.  panelLast.Visible = false;
112.  textBoxTransaksiNoInduk.Text = "";
113.  textBoxPanelTransaksiCariBuku.Text = "";
114.  textBoxTransaksiJabatan.Text = "";
116.  textBoxTransaksiJudul.Text = "";
117.  textBoxTransaksiKode.Text = "";
118.  textBoxTransaksiKodeBuku.Text = "";
119.  textBoxCariTransaksi.Text = "";
120.  textBoxTransaksiNama.Text = "";
121.  textBoxTransaksiNomorRak.Text = "";
122.  textBoxTransaksiPenerbit.Text = "";
123.  textBoxTransaksiPengarang.Text = "";
124.  textBoxTransaksiTahun.Text = "";
125.  listViewKeranjang.Items.Clear();
126.  textBoxTransaksiKode.Text = RandomString(20);
127.  comboBoxTransaksiJabatan.Visible = false;
128.                                  }
129.                              }else if (pesan == "0")
130.                              {
131.                                  MessageBox.Show("Data
132.  gagal tersimpan!", "PESAN ERROR", MessageBoxButtons.OK,
133.  MessageBoxIcon.Error);
134.                              }
135.                          }
136.
137.                      }else if (dialogResult ==
138.  DialogResult.No)
139.                      {
140.                      }
141.                  }
142.      }       }
```

Pada Tabel 4.23 menampilkan *source code* fungsi peminjaman buku pada sistem yang ada di *class* Main.cs. Pada baris 71 hingga baris 76 merupakan proses penginisialisasian data buku dan data peminjam untuk dilakukan proses peminjaman buku. Pada baris 79 adalah proses peminjaman buku dengan data yang telah diinisialisasikan pada baris 71 hingga baris 76 menggunakan method peminjaman yang ada pada *class* Transaksi.cs.

Tabel 4.25 Implementasi Algoritma *Admin* Peminjaman Buku (*Class* Transaksi.cs)

```
1.  public string peminjaman()
2.          {
3.              return
4.  db.transaksiPeminjaman(idBuku,kodePeminjaman,nama,jabatan
5.  ,noInduk,tglPinjam,tglKembali,denda);
6.          }
```

Pada Tabel 4.24 menampilkan *source code* fungsi peminjaman buku pada sistem yang ada di *class* Transaksi.cs. Pada baris 1 hingga baris 6 menjelaskan method peminjaman yang mereturn data atau mengembalikan data bervariabel *string* dari hasil proses method transaksiPeminjaman pada *class* Database.cs yang berparameter data peminjaman buku yang akan dipinjam.

Tabel 4.26 Implementasi Algoritma *Admin* Peminjaman Buku (*Class* Database.cs)

```
1.  public string transaksiPeminjaman(string idBuku, string
2.  kodePeminjaman, string nama, string jabatan, string
3.  noInduk, string tglPinjam, string tglKembali, string
4.  denda)
5.          {
6.              MySqlCommand command = con.CreateCommand();
7.              string query = "CALL transaksi_peminjaman('"+
8.  idBuku + "','" + kodePeminjaman + "','" + nama + "','" +
9.  jabatan + "','" + noInduk + "','" + tglPinjam + "','" +
10. tglKembali + "','" + denda + "')";
11.             con.Open();
12.             MySqlCommand cmd = new MySqlCommand(query,
13. con);
14.             try{
15.                 if (cmd.ExecuteNonQuery() == 1) {
16.                     con.Close();
17.                     return "1";
18.                 }else
19.                 {
20.                     con.Close();
21.                     return "0";
22.                 }
23.             }catch (Exception e) {
24.                 con.Close();
25.                 return e.Message;
26.             }
27.         }
```

Pada Tabel 4.25 menampilkan *source code* fungsi method transaksiPeminjaman pada sistem yang ada di *class* Database.cs. Pada baris 6 hingga baris 13 merupakan proses penginisialisasian *query database* dan pengeksekusian *query* pada method transaksiPeminjaman di *class* Database.cs.

4.3.5 Implementasi Algoritma *Admin* Pengembalian Buku

Implementasi algoritma *admin* melakukan pengembalian buku akan menghasilkan *source code* yang terbagi dalam beberapa *class* antara lain, *class* Main, *class* Buku, dan *class* Database. *Source code* akan ditampilkan pada Tabel 4.27 – 4.29.

Tabel 4.27 Implementasi Algoritma *Admin* Pengembalian Buku (*Class* Main.cs)

```
1.  private void buttonLihatDendaKembalikan_Click(object
2.  sender, EventArgs e)
3.          {
4.
5.              int o=
6.  listViewLihatDenda.CheckedIndices.Count;
7.
8.              if (o == 0)
```

Tabel 4.27 Implementasi Algoritma *Admin* Pengembalian Buku (*Class* Main.cs) (Lanjutan)

```
9.              {
10.                 MessageBox.Show("Silahkan ceklis buku
11.  yang ingin dikembalikan terlebih dahulu", "PESAN ERROR",
12.  MessageBoxButtons.OK, MessageBoxIcon.Error);
13.             }
14.             else
15.             {
16.                 o = 0;
17.                 for (int i = 0; i < (temp2.Rows.Count -
18.  1); i++)
19.                 {
20.                     if
21.  (listViewLihatDenda.Items[i].Checked == true)
22.                     {
23.                         Transaksi tr = new
24.  Transaksi(textBoxTransaksiKode.Text,
25.  temp2.Rows[i].Cells[0].Value.ToString(),"");
26.                         tr.pengembalian();
27.                         o++;
28.                     }
29.
30.                 }
31.
32.                 MessageBox.Show("Sejumlah "+o+" buku
34.  telah dikembalikan", "PESAN INFORMASI",
35.  MessageBoxButtons.OK, MessageBoxIcon.Information);
36.
37.                 dataGridViewTransaksi2.DataSource =
38.  transaksi.loadTransaksi();
39.                 textBoxTransaksiNoInduk.Text = "";
40.                 textBoxPanelTransaksiCariBuku.Text = "";
41.                 textBoxTransaksiJabatan.Text = "";
42.                 textBoxTransaksiJudul.Text = "";
43.                 textBoxTransaksiKode.Text = "";
44.                 textBoxTransaksiKodeBuku.Text = "";
45.                 textBoxCariTransaksi.Text = "";
46.                 textBoxTransaksiNama.Text = "";
47.                 textBoxTransaksiNomorRak.Text = "";
48.                 textBoxTransaksiPenerbit.Text = "";
49.                 textBoxTransaksiPengarang.Text = "";
50.                 textBoxTransaksiTahun.Text = "";
51.                 listViewKeranjang.Items.Clear();
52.                 panelLihatDenda.Visible = false;
53.             }
54.
56.         }
```

Pada Tabel 4.26 menampilkan *source code* fungsi pengembalian buku pada sistem yang ada di *class* Main.cs. Pada baris 43 hingga baris 44 merupakan proses penginisialisasian *class* Transaksi untuk dilakukan proses peminjaman buku. Pada baris 45 adalah proses pengembalian buku dengan menggunakan method pengembalian yang ada pada *class* Transaksi.cs.

Tabel 4.28 Implementasi Algoritma *Admin* Pengembalian Buku (*Class* Transaksi.cs)

```
1.  public void pengembalian()
2.          {
3.          db.transaksiPengembalian(kodePeminjaman,idBuku);
4.          }
```

Pada Tabel 4.27 menampilkan *source code* fungsi method pengembalian pada sistem yang ada di *class* Transaksi.cs. Pada baris 3 merupakan pemanggilan method transaksiPengembalian yang ada pada *class* Database.

Tabel 4.29 Implementasi Algoritma *Admin* Pengembalian Buku (*Class* Database.cs)

```
1.  public void transaksiPengembalian(string kode, string id)
2.          {
3.              MySqlCommand command = con.CreateCommand();
4.              string query = "CALL
5.  transaksi_pengembalian('" + kode + "','" + id + "')";
6.              con.Open();
7.              MySqlCommand cmd = new MySqlCommand(query,
8.  con);
9.              cmd.ExecuteNonQuery();
10.             con.Close();
11.         }
```

Pada Tabel 4.28 menampilkan *source code* fungsi method transaksiPengembalian pada sistem yang ada di *class* Database.cs. Pada baris 3 hingga baris 8 merupakan proses penginisialisasian *query database* dan pengeksekusian *query* pada method transaksiPengembalian di *class* Database.cs.

4.3.6 Implementasi Algoritma *Admin* Melihat Riwayat

Implementasi algoritma *admin* melihat riwayat transaksi buku yang akan menghasilkan *source code* yang terbagi dalam beberapa *class* antara lain, *class* Main, *class* Buku, dan *class* Database. *Source code* akan ditampilkan pada Tabel 4.30 – 4.32.

Tabel 4.30 Implementasi Algoritma *Admin* Melihat Riwayat(*Class* Main.cs)

```
1.  private void buttonRiwayatCari_Click(object sender,
2.  EventArgs e)
3.          {
4.              int bulan = 0;
5.              int tahun = 0;
6.              switch (comboBoxRiwayatBulan.Text)
7.              {
8.                  case "Januari":
9.                      bulan = 1;
10.                     break;
11.                 case "Februari":
12.                     bulan = 2;
13.                     break;
14.                 case "Maret":
15.                     bulan = 3;
16.                     break;
```

Tabel 4.30 Implementasi Algoritma *Admin* Melihat Riwayat(*Class* Main.cs) (Lanjutan)

```
17.                     case "April":
18.                         bulan = 4;
19.                         break;
20.                     case "Mei":
21.                         bulan = 5;
22.                         break;
23.                     case "Juni":
24.                         bulan = 6;
25.                         break;
26.                     case "Juli":
27.                         bulan = 7;
28.                         break;
29.                     case "Agustus":
30.                         bulan = 8;
31.                         break;
32.                     case "September":
34.                         bulan = 9;
35.                         break;
36.                     case "Oktober":
37.                         bulan = 10;
38.                         break;
39.                     case "November":
40.                         bulan = 11;
41.                         break;
42.                     case " Desember":
43.                         bulan = 12;
44.                         break;
45.                     case "Semua":
46.                         bulan = 13;
47.                         break;
48.                     case "--Bulan--":
49.                         bulan = 14;
50.                         break;
51.                 }switch (comboBoxRiwayatTahun.Text){
52.                     case "--Tahun--":
53.                         tahun = 14;
54.                         break;
56.                     case "Semua":
57.                         tahun = 13;
58.                         break;
59.                     default:
60.                         tahun =
61.   int.Parse(comboBoxRiwayatTahun.Text);
62.                         break;
63.                 }
64.                 if(bulan == 14 || tahun == 14 || bulan == 14
65.   && tahun == 14 || comboBoxRiwayatStatus.Text == "--Status
67.   Buku--")
68.                 {MessageBox.Show("Periksa kembali pilihan
69.   pencarian!", "PESAN ERROR", MessageBoxButtons.OK,
70.   MessageBoxIcon.Error);
71.                 }
72.   dataGridViewPanelRiwayat.DataSource =
73.   transaksi.cariRiwayat(bulan,tahun,comboBoxRiwayatStatus.T
74.   ext);
75.             }
```

Pada Tabel 4.29 menampilkan *source code* fungsi melihat riwayat transaksi buku pada sistem yang ada di *class* Main.cs. Pada baris 73 dan baris 74 merupakan proses pemanggilan method cariRiwayat pada *class* transaksi yang dimuatkan pada tabel dalam bentuk fungsi dataGridView.

Tabel 4.31 Implementasi Algoritma *Admin* Melihat Riwayat(*Class* Transaksi.cs)

```
1.  public BindingSource cariRiwayat(int bulan, int tahun,
2.  string status) {
3.              return db.cariRiwayat(bulan, tahun, status);
4.          }
```

Pada Tabel 4.30 menampilkan *source code* fungsi method loadRiwayat pada sistem yang ada di *class* Riwayat.cs. Pada baris 3 merupakan pemanggilan method cariRiwayat yang ada pada *class* Database yang berfungsi mengambil semua data riwayat transaksi yang ada dalam *database*.

Tabel 4.32 Implementasi Algoritma *Admin* Melihat Riwayat(*Class* Database.cs)

```
1.  public BindingSource cariRiwayat(int bulan, int tahun,
2.  string status) {
3.              con.Open();
4.          MySqlDataAdapter adapter = new MySqlDataAdapter();
5.              MySqlCommand cmd;
6.              DataSet ds = new DataSet();
7.              BindingSource bs = new BindingSource();
8.              string sql = "CALL cari_riwayat('" + bulan +
9.  "','" + tahun + "','" + status + "')";
10.             cmd = new MySqlCommand(sql, con);
11.             adapter.SelectCommand = cmd;
12.             adapter.Fill(ds);
13.             bs.DataSource = ds.Tables[0];
14.             con.Close();
15.             DataGridView dg = new DataGridView();
16.             return bs;
17..        }
```

Pada Tabel 4.31 menampilkan *source code* fungsi method loadRiwayat pada sistem yang ada di *class* Database.cs. Pada baris 8 dan baris 10 merupakan proses penginisialisasian *query database* dan pengeksekusian *query* pada method cariRiwayat di *class* Database.cs.

4.3.7 Implementasi *Database* (*Data Definition Language*)

Implementasi *database* akan menggunakan data definition language yang diimplementasikan menggunakan bahasa pemrograman *database SQL*. *Data definition language* dalam bentuk bahasa pemrograman *SQL* ini akan mengimplementasikan *database* yang telah dibuat entitas-entitasnya pada sub bab 4.2.4 Perancangan *Database*.

Tabel 4.33 Implementasi Tabel buku

```
1.  CREATE TABLE `buku` (
2.    `id_buku` int(255) NOT NULL,
3.    `kode_buku` varchar(100) DEFAULT NULL,
4.    `judul_buku` varchar(100) DEFAULT NULL,
5.    `id_pengarang` int(100) DEFAULT NULL,
6.    `tahun` year(4) DEFAULT NULL,
7.    `id_penerbit` int(100) DEFAULT NULL,
8.    `rak_buku` varchar(100) DEFAULT NULL
9.  );
10. ALTER TABLE `buku`
11.   ADD PRIMARY KEY (`id_buku`),
12.   ADD UNIQUE KEY `kode_buku` (`kode_buku`);
13. ALTER TABLE `buku`
14.   MODIFY `id_buku` int(255) NOT NULL AUTO_INCREMENT,
15. AUTO_INCREMENT=0;
```

Pada Tabel 4.33 menjelaskan DDL *query SQL* untuk membuat tabel buku yang berfungsi untuk menyimpan data buku yang ada pada sistem informasi perpustakaan.

Tabel 4.34 Implementasi Tabel penerbit

```
1.  CREATE TABLE `penerbit` (
2.    `id_penerbit` int(11) NOT NULL,
3.    `nama_penerbit` varchar(100) NOT NULL,
4.    `alamat_penerbit` varchar(100) NOT NULL,
5.    `noTelp_penerbit` varchar(100) NOT NULL
6.  );
7.  ALTER TABLE `penerbit`
8.    ADD PRIMARY KEY (`id_penerbit`),
9.    ADD UNIQUE KEY `noTelp_penerbit` (`noTelp_penerbit`);
10. ALTER TABLE `penerbit`
11.   MODIFY `id_penerbit` int(11) NOT NULL AUTO_INCREMENT,
12. AUTO_INCREMENT=0;
```

Pada Tabel 4.34 menjelaskan DDL *query SQL* untuk membuat tabel penerbit buku yang berfungsi untuk menyimpan data penerbit buku secara detail yang ada pada sistem informasi perpustakaan.

Tabel 4.35 Implementasi Tabel pengarang

```
1.  CREATE TABLE `pengarang` (
2.    `id_pengarang` int(11) NOT NULL,
3.    `nama_pengarang` varchar(100) NOT NULL
4.  );
5.  ALTER TABLE `pengarang`
6.    ADD PRIMARY KEY (`id_pengarang`),
7.  ALTER TABLE `pengarang`
8.    MODIFY `id_pengarang` int(11) NOT NULL AUTO_INCREMENT,
9.  AUTO_INCREMENT=0;
```

Pada Tabel 4.35 menjelaskan DDL *query SQL* untuk membuat tabel pengarang buku yang berfungsi untuk menyimpan data pengarang buku secara detail yang ada pada sistem informasi perpustakaan.

Tabel 4.36 Implementasi Tabel transaksi

```
1.  CREATE TABLE `transaksi` (
2.    `id_transaksi` int(100) NOT NULL,
3.    `kode_pinjam` varchar(100) NOT NULL,
4.    `id_peminjam` int(11) NOT NULL,
5.    `tgl_pinjam` date NOT NULL,
6.    `tgl_kembali` date NOT NULL,
7.    `jumlah_buku` int(10) NOT NULL
8.  ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
9.  ALTER TABLE `transaksi`
10. ADD PRIMARY KEY (`id_transaksi`);
11. ALTER TABLE `transaksi`
12. MODIFY `id_transaksi` int(100) NOT NULL AUTO_INCREMENT;
```

Pada Tabel 4.36 menjelaskan DDL *query SQL* untuk membuat tabel transaksi buku yang berfungsi untuk menyimpan data transaksi buku secara detail yang ada pada sistem informasi perpustakaan.

Tabel 4.37 Implementasi Tabel peminjam

```
1.  CREATE TABLE `peminjam` (
2.    `id_peminjam` int(11) NOT NULL,
3.    `nama_peminjam` varchar(30) NOT NULL,
4.    `jabatan` varchar(30) NOT NULL,
5.    `no_induk` varchar(30) NOT NULL
6.  );
7.  ALTER TABLE `peminjam`
8.    ADD PRIMARY KEY (`id_peminjam`);
9.  ALTER TABLE `peminjam`
10.   MODIFY `id_peminjam` int(11) NOT NULL AUTO_INCREMENT,
11. AUTO_INCREMENT=0;
```

Pada Tabel 4.37 menjelaskan DDL *query SQL* untuk membuat tabel peminjam buku yang berfungsi untuk menyimpan data peminjam buku secara detail yang ada pada sistem informasi perpustakaan.

Tabel 4.38 Implementasi Detail Transaksi

```
1.  CREATE TABLE `detail_transaksi` (
2.    `kode_pinjam` varchar(20) NOT NULL,
3.    `id_buku` int(20) NOT NULL,
4.    `status` varchar(30) NOT NULL,
5.    `denda` int(10) NOT NULL,
6.    `tgl_pengembalian` date DEFAULT NULL
7.  );
8.  ALTER TABLE `detail_transaksi`
9.    ADD PRIMARY KEY (`kode_pinjam`,`id_buku`);
```

Pada Tabel 4.38 menjelaskan DDL *query SQL* untuk membuat tabel detail_transaksi buku yang berfungsi untuk menyimpan data peminjam buku secara detail yang ada pada sistem informasi perpustakaan.

4.3.8 Hasil Implementasi Antar Muka Sistem

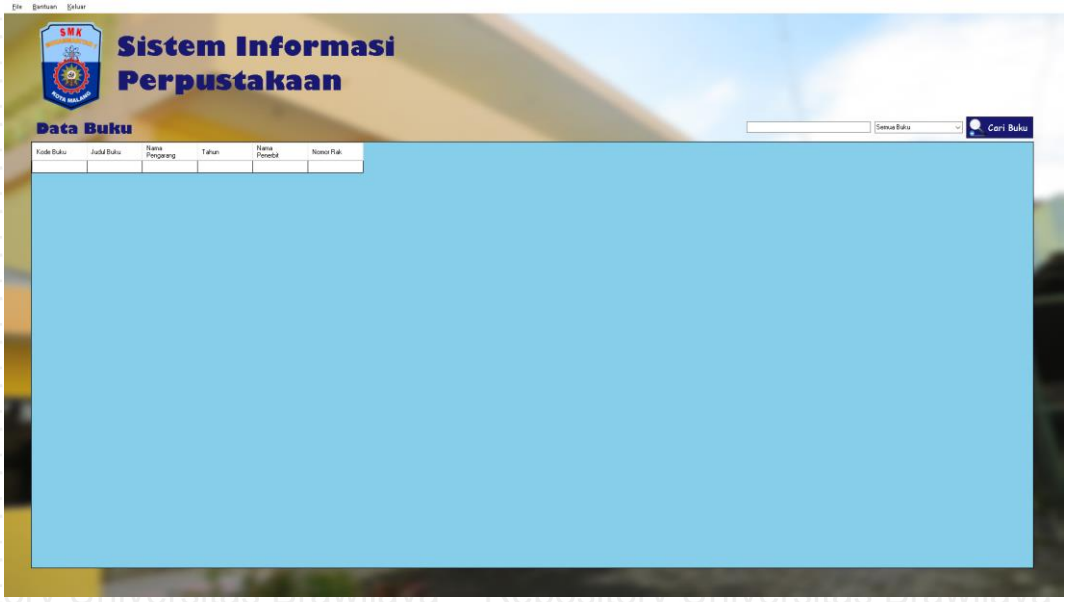
Hasil implementasi antar muka sistem adalah hasil dari implementasi perancangan antar muka sistem yang telah dibuat pada sub bab 4.2.5 Perancangan Antar Muka Sistem.

4.3.8.1 Tampilan Antar Muka Awal Sistem

Gambar 4.32 Tampilan Antar Muka Awal Sistem

Pada Gambar 4.31 menggambarkan tampilan antar muka di awal saat sistem baru dijalankan. *User* dapat memilih masuk kedalam sistem menggunakan hak akses pengunjung atau pun menggunakan hak akses *admin.*

4.3.8.2 Tampilan Antar Muka Dashboard Pengunjung



Gambar 4.33 Tampilan Antar Muka *Dashboard* Pengunjung

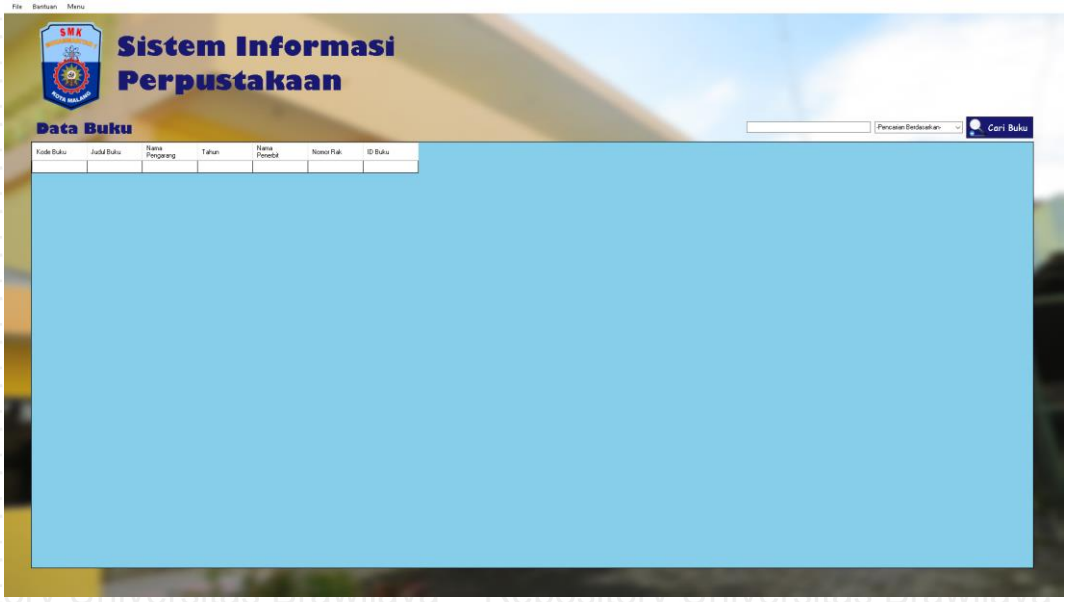
Pada Gambar 4.32 menggambarkan tampilan antar muka sistem di saat *user* menggunakan hak akses pengunjung untuk masuk kedalam sistem. Pada *dashboard* pengunjung sistem menampilkan panel pencarian buku yang dapat dilakukan pengunjung.

4.3.8.3 Tampilan Antar Muka *Login Admin*

Gambar 4.34 Tampilan Antar Muka *Login Admin*

Pada Gambar 4.33 menggambarkan tampilan antar muka sistem pada saat *user* memilih masuk sebagai *admin* dan diharuskan untuk memasukkan password dengan benar untuk menampilkan *dashboard admin.*

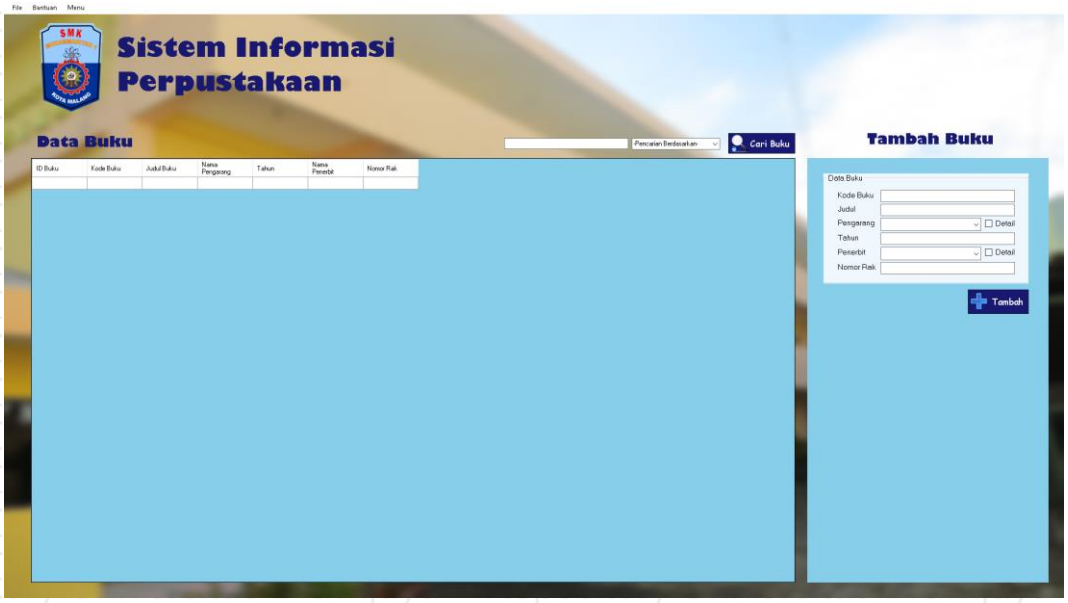
4.3.8.4 Tampilan Antar Muka *Dashboard Admin*



Gambar 4.35 Tampilan Antar Muka *Dashboard Admin*

Pada Gambar4.34 menggambarkan tampilan antar muka sistem apabila *user* memilih masuk sebagai *admin* dan memasukkan data password dengan benar. Sistem akan menampilkan *dashboard admin* dengan panel pencarian yang dapat dilakukan oleh *admin.*

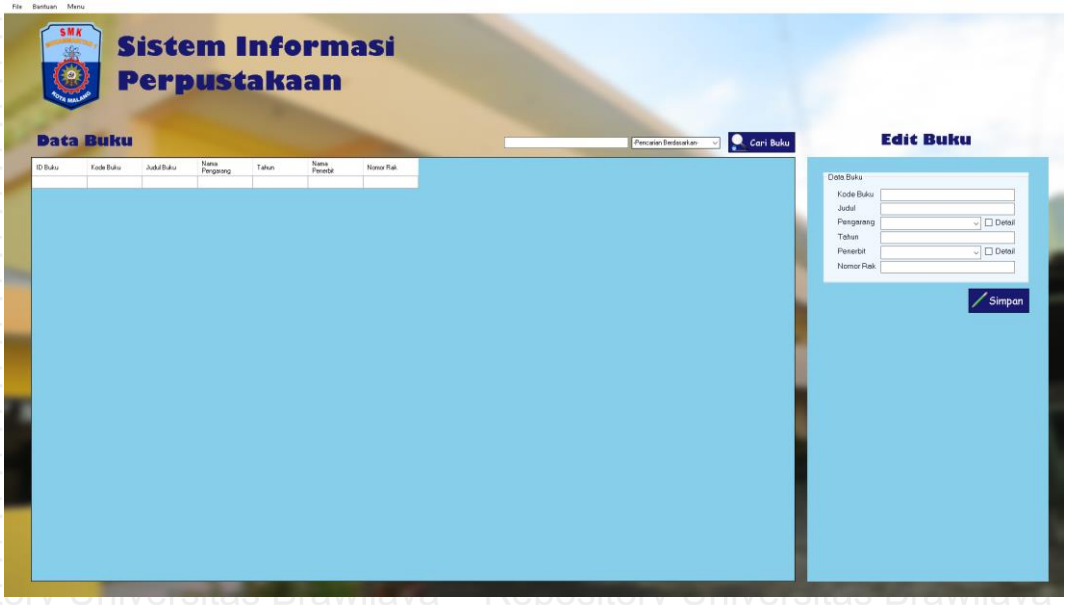
4.3.8.5 Tampilan Antar Muka *Admin* Tambah Buku



Gambar 4.36 Tampilan Antar Muka *Admin* Tambah Buku

Pada Gambar 4.35 mengambarkan tampilan antar muka sistem saat *admin* memilih "Menu" pada bagian menu bar dan memilih untuk melakukan kelola buku dengan sub menu "Tambah".

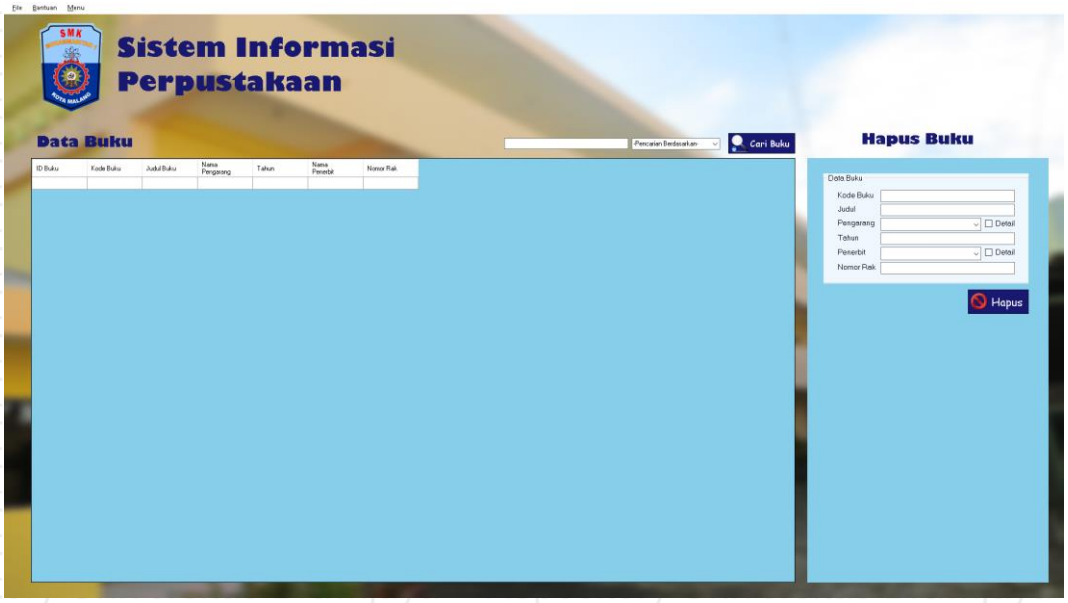
4.3.8.6 Tampilan Antar Muka *Admin* Edit Buku



Gambar 4.37 Tampilan Antar Muka *Admin* Edit Buku

Pada Gambar 4.36 mengambarkan tampilan antar muka sistem saat *admin* memilih "Menu" pada bagian menu bar dan memilih untuk melakukan kelola buku dengan sub menu "Edit".

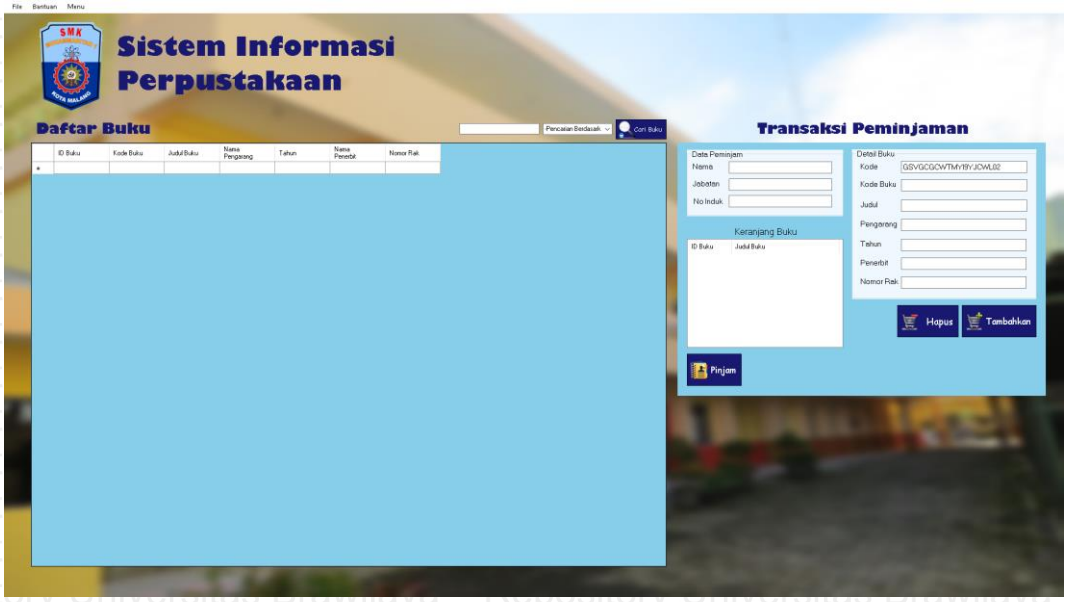
4.3.8.7 Tampilan Antar Muka *Admin* Hapus Buku



Gambar 4.38 Tampilan Antar Muka *Admin* Hapus Buku

Pada Gambar 4.37 mengambarkan tampilan antar muka sistem saat *admin* memilih "Menu" pada bagian menu bar dan memilih untuk melakukan kelola buku dengan sub menu "Hapus".

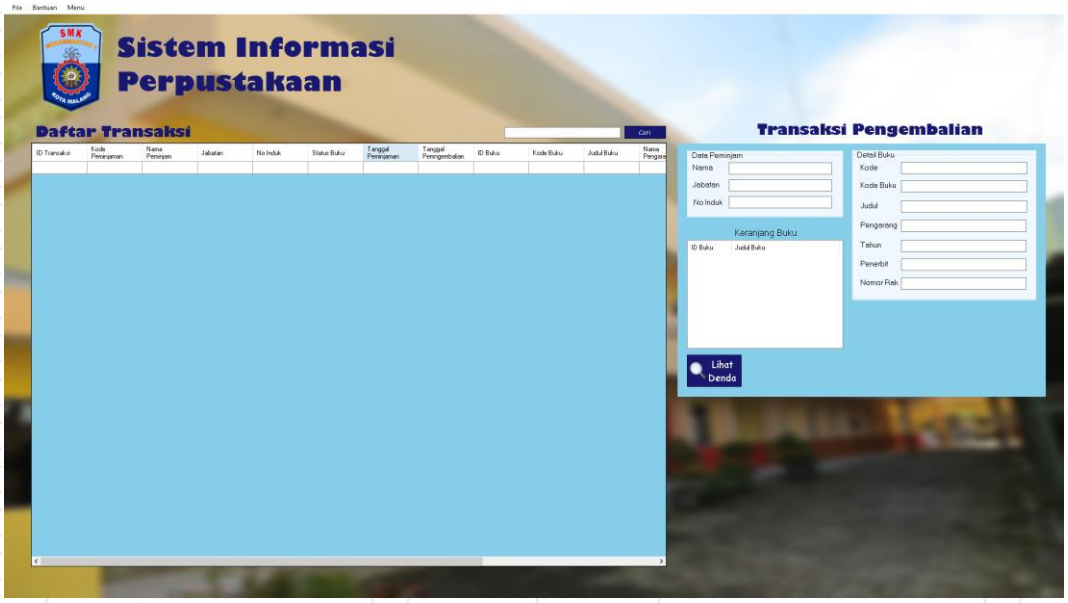
4.3.8.8 Tampilan Antar Muka *Admin* Transaksi Peminjaman Buku



Gambar 4.39 Tampilan Antar Muka *Admin* Transaksi Peminjaman Buku

Pada Gambar 4.38 mengambarkan tampilan antar muka sistem saat *admin* memilih "Menu" pada bagian menu bar dan memilih untuk melakukan transaksi buku dengan sub menu "Peminjaman".

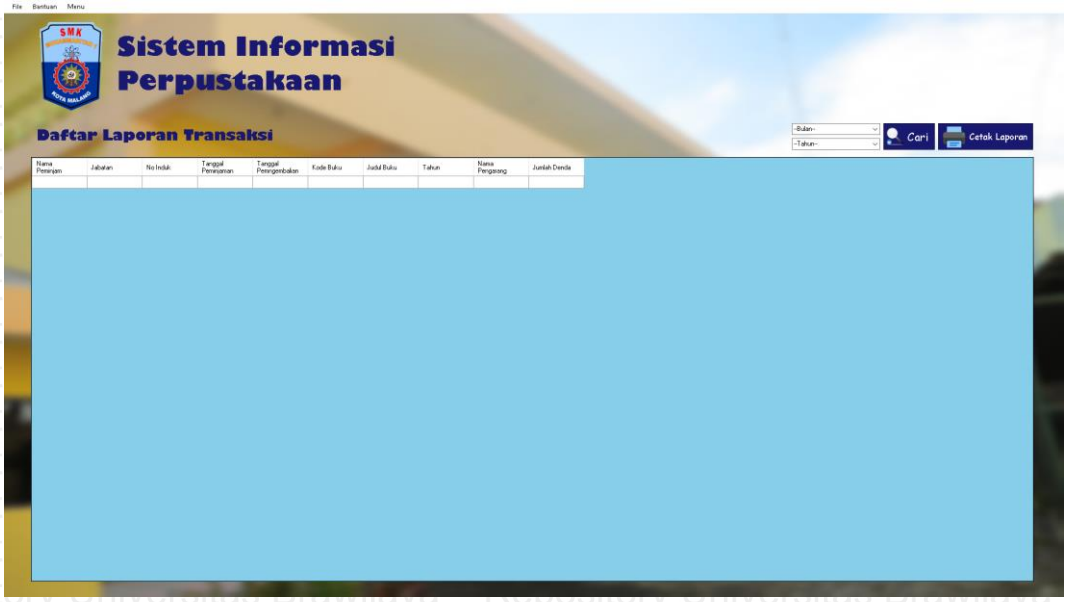
4.3.8.9 Tampilan Antar Muka *Admin* Pengembalian Buku



Gambar 4.40 Tampilan Antar Muka *Admin* Pengembalian Buku

Pada Gambar 4.39 mengambarkan tampilan antar muka sistem saat *admin* memilih "Menu" pada bagian menu bar dan memilih untuk melakukan transaksi buku dengan sub menu "Pengembalian".

4.3.8.10 Tampilan Antar Muka *Admin* Melihat Riwayat



Gambar 4.41 Tampilan Antar Muka *Admin* Melihat Riwayat

Pada Gambar 4.40 mengambarkan tampilan antar muka sistem saat *admin* memilih "Menu" pada bagian menu bar dan memilih untuk melakukan lihat riwayat buku dengan menu "Riwayat".

4.4 Pengujian dan Analisis (*Testing*)

Pada tahap pengujian, yang mana sebelum sistem diterapkan atau digunakan secara langsung oleh *user*, maka sistem diharuskan melewati tahap pengujian dan analisis terlebih dahulu agar sistem tidak melenceng cari yang diharapkan oleh *stackholder* ataupun munculnya error pada sistem saat sistem diterapkan secara langsung. Pada tahap ini, sistem akan dilakukan pengujian menggunakan teknik pengujian *white-box testing, black-box testing,* serta *User Acceptance Testing* (UAT).

4.4.1 White-Box Testing

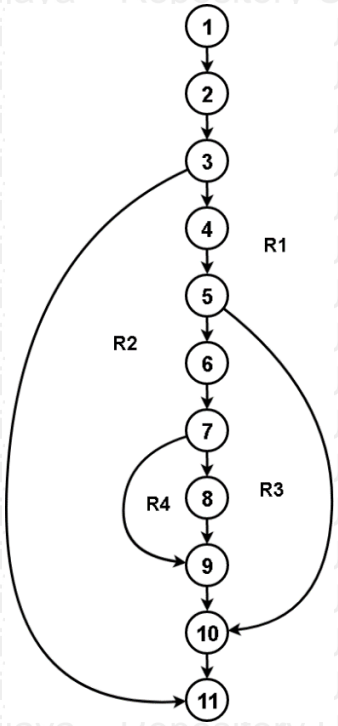
Dalam pengujian *white-box testing* hal yang pertama dilakukan adalah membuat *test case* dalam bentuk alur program dengan menggunakan *flowchart* atau *pseudocode*. Setelah *pseudocode* dibuat langkah selanjutnya adalah membuat *flowgraph* dari *pseudocode* yang telah dibuat. Dengan *basis path* maka dapat menghitung *complexity cyclometic* dan didapatkan jalur *independent path.*

4.4.1.1 *White-Box Testing Admin* Transaksi Pengembalian Buku

Tabel 4.39 *Pseudocode* Transaksi Pengembalian Buku

Node	Alogitma
(1)	mulai
(2)	cari kode peminjaman
(3)	jika ditemukan kode peminjaman maka
(4)	cek status buku
(5)	jika belum dikembalikan maka
(6)	hitung selisih tanggal antara tanggal sekarang dan tanggal pengembalian
(7)	jika selisih tanggal melebihi maka
(8)	hitung denda buku
(9)	kembalikan buku
(10)	buku telah dikembalikan
(11)	selesai

Pada Tabel 4.39 menjelaskan algoritma *Pseudocode* untuk melakukan pengembalian buku dan melakukan perhitungan denda yang akan didapatkan apabila melakukan keterlambatan dalam pengembalian buku di perpustakaan.



Gambar 4.42 *Flowgraph* Transaksi Pengembalian Buku

Gambar 4.41 menjelaskan *flow graph* yang didefinisikan dari tabel algoritma *pseudocode*. Dengan telah dibuatnya *flowgraph* maka dapat menghitung rumus *cyclometic complexity* sebagai berikut:

V(G) = Jumlah edge – jumlah node + 2

= 13 – 11 + 2

= 4

V(G) = Jumlah predicated node + 1

= 3 + 1

= 4

V(G) = Jumlah region

= 4

Dengan *flowgraph* yang telah dibuat menghasilkan jalur independent sebagai berikut:

1. 1 – 2 – 3 – 11
2. 1 – 2 – 3 – 4 – 5 – 10 – 11
3. 1 – 2 – 3 – 4 – 5 – 6 – 7 – 9 – 10 – 11
4. 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11

4.4.2 *Black-Box Testing*

Pengujian *black-box testing* yang dilakukan terhadap sistem pada tahap ini menggunakan metode validasi (*validation*). Metode validasi yang digunakan dalam melakukan pengujian ini berfungsi untuk mengetahui valid atau tidaknya sebuah fungsi dari sistem yang dibangun. Melakukan pengujian *black-box* dengan metode validasi ini juga menentukan apakah sistem telah sesuai seperti apa yang diinginkan oleh *stackholder* pada tahap perencanaan. Pengujian *black-box* ini dilakukan dengan jumlah *test case* sebanyak 25 (dua puluh lima).

4.4.2.1 *Black-Box Testing* Pengunjung Mencari Buku

Tabel 4.40 *Black-Box Testing* Pengunjung Mencari Buku

Kode	Nama Tes	Kasus Uji	Hasil Yang Diinginkan	Hasil	Status
SIMP-F-01	Pengujian pengunjung mencari data buku	Pengujian dengan pengunjung memasukkan kata kunci dan kategori dengan benar kemudian menekan tombol “Cari”.	Sistem akan menampilkan daftar buku yang dicari sesuai dengan kategori.	Sistem akan menampilkan daftar buku yang dicari sesuai dengan kategori.	Valid
SIMP-F-01	Pengujian pengunjung mencari data buku	Pengujian dengan pengunjung tidak memasukkan kata kunci dan kategori dengan benar dan menekan tombol “Cari”.	Sistem akan menampilkan pesan “Periksa kembali data yang ingin dicari”.	Sistem menampilkan pesan “Periksa kembali data yang ingin dicari”.	Valid

4.4.2.2 *Black-Box Testing Admin* Mencari Buku

Tabel 4.41 *Black-Box Testing Admin* Mencari Buku

Kode	Nama Tes	Kasus Uji	Hasil Yang Diinginkan	Hasil	Status
SIMP-F-03	Pengujian *admin* mencari data buku	Pengujian dengan *admin* memasukkan	Sistem akan menampilkan daftar buku yang dicari	Sistem akan menampilkan daftar buku yang dicari	Valid

Tabel 4.40 *Black-Box Testing Admin* Mencari Buku (Lanjutan)

Kode	Nama Tes	Kasus Uji	Hasil Yang Diinginkan	Hasil	Status
		kata kunci dan kategori dengan benar dan menekan tombol "Cari".	sesuai dengan kategori.	sesuai dengan kategori.	
SIMP-F-03	Pengujian *admin* mencari data buku	Pengujian dengan *admin* tidak memasukkan kata kunci dan kategori dengan benar dan menekan tombol "Cari".	Sistem akan menampilkan pesan "Periksa kembali data yang ingin dicari".	Sistem akan menampilkan pesan "Periksa kembali data yang ingin dicari".	Valid

4.4.2.3 *Black-Box Testing Admin* Menambah Buku

Tabel 4.42 *Black-Box Testing Admin* Menambah Buku

Kode	Nama Tes	Kasus Uji	Hasil Yang Diinginkan	Hasil	Status
SIMP-F-04-A	Pengujian *admin* menambah data buku	Pengujian dengan *admin* memasukkan data buku dengan benar kemudian menekan tombol "Tambah".	Sistem akan menampilkan pesan "Data buku tersimpan" dan menampilkan seluruh buku yang ada .	Sistem akan menampilkan pesan "Data buku tersimpan" dan menampilkan seluruh buku yang ada.	Valid
SIMP-F-04-A	Pengujian *admin* menambah data buku	Pengujian dengan *admin* tidak memasukkan data buku secara lengkap kemudian menekan tombol "Tambah".	Sistem akan menampilkan pesan "Isi form yang kosong".	Sistem akan menampilkan pesan "Isi form yang kosong".	Valid

Tabel 4.41 *Black-Box Testing Admin* Menambah Buku (Lanjutan)

Kode	Nama Tes	Kasus Uji	Hasil Yang Diinginkan	Hasil	Status
SIMP-F-04-A	Pengujian *admin* menambah data buku	Pengujian dengan *admin* memasukkan data kode buku yang sama kemudian menekan tombol "Tambah".	Sistem akan menampilkan pesan "Kode buku yang sama telah tersimpan dalam sistem".	Sistem akan menampilkan pesan "Kode buku yang sama telah tersimpan dalam sistem".	Valid
SIMP-F-04-A	Pengujian *admin* menambah data buku	Pengujian dengan *admin* memasukkan data nomor telpon pengarang buku yang sama kemudian menekan tombol "Tambah".	Sistem akan menampilkan pesan "Nomor telepon pengarang yang sama telah ada dalam sistem".	Sistem akan menampilkan pesan "Nomor telepon pengarang yang sama telah ada dalam sistem".	Valid
SIMP-F-04-A	Pengujian *admin* menambah data buku	Pengujian dengan *admin* memasukkan data nomor telpon penerbit buku yang sama kemudian menekan tombol "Tambah".	Sistem akan menampilkan pesan "Nomor telepon penerbit yang sama telah ada dalam sistem".	Sistem akan menampilkan pesan "Nomor telepon penerbit yang sama telah ada dalam sistem".	Valid

4.4.2.4 *Black-Box Testing Admin* Mengedit Buku

Tabel 4.43 *Black-Box Testing Admin* Mengedit Buku

Kode	Nama Tes	Kasus Uji	Hasil Yang Diinginkan	Hasil	Status
SIMP-F-04-B	Pengujian *admin* mengedit data buku	Pengujian dengan *admin* mengedit data buku dengan benar kemudian menekan tombol "Simpan".	Sistem akan menampilkan pesan "Data buku tersimpan" dan menampilkan seluruh daftar buku yang ada.	Sistem akan menampilkan pesan "Data buku tersimpan" dan menampilkan seluruh buku yang ada.	Valid
SIMP-F-04-B	Pengujian *admin* mengedit data buku	Pengujian dengan *admin* tidak memasukkan data buku secara lengkap kemudian menekan tombol "Simpan".	Sistem akan menampilkan pesan "Isi form yang kosong".	Sistem akan menampilkan pesan "Isi form yang kosong".	Valid
SIMP-F-04-B	Pengujian *admin* mengedit data buku	Pengujian dengan *admin* memasukkan data kode buku yang sama kemudian menekan tombol "Simpan".	Sistem akan menampilkan pesan "Kode buku yang sama telah tersimpan dalam sistem".	Sistem akan menampilkan pesan "Kode buku yang sama telah tersimpan dalam sistem".	Valid
SIMP-F-04-B	Pengujian *admin* mengedit data buku	Pengujian *admin* memasukkan data nomor telpon pengarang buku yang sama dan menekan tombol "Simpan".	Sistem akan menampilkan pesan "Nomor telepon pengarang yang sama telah ada pada sistem".	Sistem akan menampilkan pesan "Nomor telepon pengarang yang sama telah ada pada sistem".	Valid

Tabel 4.42 *Black-Box Testing Admin* Mengedit Buku (Lanjutan)

Kode	Nama Tes	Kasus Uji	Hasil Yang Diinginkan	Hasil	Status
SIMP-F-04-B	Pengujian *admin* men mengedit ambah data buku	Pengujian dengan *admin* memasukkan data nomor telpon penerbit buku yang sama kemudian menekan tombol "Simpan".	Sistem akan menampilkan pesan "Nomor telepon penerbit yang sama telah ada dalam sistem".	Sistem akan menampilkan pesan "Nomor telepon penerbit yang sama telah ada dalam sistem".	Valid

4.4.2.5 *Black-Box Testing Admin* Menghapus Buku

Tabel 4.44 *Black-Box Testing Admin* Menghapus Buku

Kode	Nama Tes	Kasus Uji	Hasil Yang Diinginkan	Hasil	Status
SIMP-F-04-C	Pengujian *admin* menghapus data buku	Pengujian dengan *admin* memilih buku dalam daftar buku kemudian menekan tombol "Hapus"	Sistem akan menampilkan pesan "Data buku telah terhapus" dan menampilkan seluruh daftar buku yang ada.	Sistem akan menampilkan pesan "Data buku telah terhapus" dan menampilkan seluruh daftar buku yang ada.	Valid
SIMP-F-04-C	Pengujian *admin* menghapus data buku	Pengujian dengan *admin* menekan tombol "Hapus" tanpa memilih buku.	Sistem akan menampilkan pesan "Pilih buku terlebih dahulu".	Sistem akan menampilkan pesan "Pilih buku terlebih dahulu".	Valid

4.4.2.6 *Black-Box Testing Admin* Transaksi Peminjaman Buku

Tabel 4.45 *Black-Box Testing Admin* Transaksi Peminjaman Buku

Kode	Nama Tes	Kasus Uji	Hasil Yang Diinginkan	Hasil	Status
SIMP-F-05	Pengujian *admin* melakukan transaksi peminjaman buku	Pengujian dengan *admin* menambahkan buku kedalam keranjang, mengisi data peminjam dengan lengkap kemudian menekan tombol "Pinjam".	Sistem akan menampilkan pesan "Data transaksi tersimpan! Tekan OK untuk mencetak bukti peminjaman" dan sistem akan mencetak bukti peminjaman.	Sistem akan menampilkan pesan "Data transaksi tersimpan! Tekan OK untuk mencetak bukti peminjaman" dan sistem akan mencetak bukti peminjaman.	Valid
SIMP-F-05	Pengujian *admin* melakukan transaksi peminjaman buku	Pengujian dengan *admin* menambahkan buku kedalam keranjang, tanpa mengisi data peminjam dengan lengkap kemudian menekan tombol "Pinjam".	Sistem akan menampilkan pesan "Lengkapi data peminjam".	Sistem akan menampilkan pesan "Lengkapi data peminjam".	Valid
SIMP-F-05	Pengujian *admin* melakukan transaksi peminjaman buku	Pengujian dengan *admin* tidak menambahkan buku kedalam keranjang, mengisi data peminjam dengan lengkap dan menekan tombol "Pinjam".	Sistem akan menampilkan pesan "Buku dalam keranjang kosong".	Sistem akan menampilkan pesan "Buku dalam keranjang kosong".	Valid

4.4.2.7 *Black-Box Testing Admin* Transaksi Pengembalian Buku

Tabel 4.46 *Black-Box Testing Admin* Transaksi Pengembalian Buku

Kode	Nama Tes	Kasus Uji	Hasil Yang Diinginkan	Hasil	Status
SIMP-F-06	Pengujian *admin* melakukan transaksi pengembalian buku	Pengujian dengan *admin* memilih buku untuk dikembalikan menekan tombol "Lihat Denda", menceklis buku yang ingin dikembalikan kemudian menekan tombol "Kembalikan".	Sistem akan menampilkan pesan "Sejumlah x buku telah dikembalikan".	Sistem akan menampilkan pesan "Sejumlah x buku telah dikembalikan".	Valid
SIMP-F-06	Pengujian *admin* melakukan transaksi pengembalian buku	Pengujian dengan *admin* tidak memilih buku untuk dikembalikan menekan tombol "Lihat Denda".	Sistem akan menampilkan pesan "Pilih buku terlebih dahulu".	Sistem akan menampilkan pesaan "Pilih buku terlebih dahulu".	Valid
SIMP-F-05	Pengujian *admin* melakukan transaksi peminjaman buku	Pengujian dengan *admin* memilih buku untuk dikembalikan menekan tombol "Lihat Denda", tidak menceklis buku yang ingin dikembalikan dan menekan tombol "Kembalikan".	Sistem akan menampilkan pesan "Silahkan ceklis buku yang ingin dikembalikan terlebih dahulu".	Sistem akan menampilkan pesan "Silahkan ceklis buku yang ingin dikembalikan terlebih dahulu".	Valid

4.4.2.8 *Black-Box Testing Admin* Melihat Riwayat Transaksi

Tabel 4.47 *Black-Box Testing* Admin Melihat Riwayat Transaksi

Kode	Nama Tes	Kasus Uji	Hasil Yang Diinginkan	Hasil	Status
SIMP-F-07	Pengujian *admin* melihat riwayat	Pengujian dengan *admin* memilih sub menu "Riwayat" pada menu bar "Menu".	Sistem akan menampilkan seluruh daftar transaksi buku yang pernah tercatat.	Sistem akan menampilkan seluruh daftar transaksi buku yang pernah tercatat.	Valid
SIMP-F-07	Pengujian *admin* melihat riwayat dengan pencarian	Pengujian dengan *admin* memilih spesifik bulan dan tahun kemudian menekan tombol "Cari".	Sistem akan menampilkan seluruh transaksi pada bulan dan tahun yang dipilih.	Sistem akan menampilkan seluruh transaksi pada bulan dan tahun yang dipilih.	Valid
SIMP-F-07	Pengujian *admin* melakukan cetak laporan	Pengujian dengan *admin* menekan tombol "Cetak Laporan".	Sistem akan mencetak riwayat transaksi yang ada pada tabel.	Sistem akan mencetak laporan transaksi yang ada pada tabel.	Valid

4.4.3 *User Acceptance Testing* (UAT)

Pengujuan *user acceptance* ini akan dilakukan dengan menggunakan *test case* yang hanya dilakukan oleh pengguna *admin* yaitu fungsi peminjaman buku dan pengembalian buku. Untuk mengetahui penilaian yang akan diberikan oleh pengguna maka dibuat pertanyaan yang akan diajukan kepada pengguna berdasarkan *test case* yang akan diujikan kepada pengguna. Tabel kriteria pengujian akan dijelaskan pada Tabel 4.48. *Test case* fungsi peminjaman buku dan pengembalian buku akan dijelaskan pada Tabel 4.49-4.50. Pertanyaan yang akan diberikan kepada pengguna akan dijelaskan pada Tabel 4.49.

Tabel 4.48 Kriteria *User Acceptance Testing* (UAT)

Kriteria	Deskripsi
Efisiensi	• Pengguna mencapai tujuan tertentu dan sumber daya yang dikeluarkan untuk mencapainya. • Waktu penyelesaian tugas dan waktu untuk mempelajari sistem. • Pengguna dapat menemukan informasi lebih cepat.
Efektivitas	• Pengguna dapat mencapai tujuan tertentu. • Termasuk kualitas solusi dan tingkat kesalahan. • Tercapainya tujuan untuk keseluruhan sistem.
Kepuasan	• Kenyamanan pengguna dan sikap positif terhadap penggunaan aplikasi. • Kepuasan pengguna dapat diukur dengan skala penilaian sikap. • Tercapainya tujuan untuk keseluruhan sistem. • Pengguna menganggap aplikasi keseluruhan dapat diterima.
Error	• Tingkat kesalahan yang didapat selama waktu menjalakan sistem.

Tabel 4.49 *Test Case* Peminjaman Buku

Step No. 1
Deskripsi : *Admin* telah berhasil login dan memilih menu peminjaman buku
Hasil yang diharapkan : Halaman peminjaman buku ditampilkan.
ss
Step No. 2
Deskripsi : Setelah menampilkan halaman peminjaman buku. Mengisi form peminjaman buku dengan valid.

a) Melakukan pengisian pada form data peminjam secara valid.
b) Memilih buku yang akan dipinjam dan menambahkannya kedalam keranjang.
c) Menekan tombol “Pinjam”.
d) Mencetak bukti peminjaman buku.

Hasil yang diharapkan :

a) Buku dapat dipinjam dan merubah status buku menjadi “dipinjam” pada *database* sistem dan sistem mencetak bukti peminjaman.

Tabel 4.50 *Test Case* Pengembalian Buku

Step No. 1 Deskripsi : *Admin* telah berhasil login dan memilih menu pengembalian buku Hasil yang diharapkan : Setelah menampilkan halaman peminjaman buku. Mengisi form peminjaman buku dengan valid. Step No. 2 Deskripsi : Setelah menampilkan halaman login. Mengisi form login dengan valid a) Melakukan pencarian kode peminjaman. b) Menekan tombol "Lihat Denda" untuk menghitung denda yang akan didapatkan. c) Memilih buku yang ingin dikembalikan dengan mencentang buku. d) Menekan tombol Hasil yang diharapkan : a) Menampilkan jumlah denda. b) Buku dapat dikembalikan dan dapat dipinjam kembali serta merubah status buku menjadi "telah dikembalikan" pada *database* sistem.

Tabel 4.51 Pertanyaan *User Acceptance Testing* (UAT)

Kriteria	ID	Pertanyaan	Kemungkinan Jawaban
Efisiensi	Q1	Seberapa baik tingkat akurasi dan kelengkapan aplikasi berbasis desktop dapat memberikan informasi dan fungsionalitas sesuai dengan:	• Efisien • Sedang • Tidak efisien
	Q1.1	1.Kualitas dari solusi untuk menyelesaikan permasalahan beserta waktu yang didapatkan.	
	Q1.1	2.Penyelesaian permasalahan	
Efektifitas	Q2	Seberapa efektif aplikasi berbasis desktop itu berdasarkan:	• Efektif • Sedang • Tidak efektif
	Q2.1	1.Partisi informasi yang logis.	
	Q2.2	2.Penempatan tautan yang tepat.	
	Q2.3	3.Pemuatan tabel sheet yang tepat.	
	Q2.4	4.Pesan dan manajemen data yang dinamis.	
	Q2.5	5.Konsistensi dan tata letak atau layout halaman.	

Tabel 4.50 Pertanyaan *User Acceptance Testing* (UAT) (Lanjutan)

Kriteria	ID	Pertanyaan	Kemungkinan Jawaban
Kepuasan	Q3	Berapa tingkat kesulitan yang dapat ditemukan pengguna dalam aplikasi berdasarkan:	• Mudah • Sedang • Tidak mudah
	Q3.1	1.Penyelesaian sebuah tugas.	
	Q3.2	2.Melakukan interaksi pada sistem	
Error	Q4	Seberapa maksimum tingkat kesalahan dapat dideteksi pada aplikasi berdasarkan:	• Tinggi • Sedang • Rendah
	Q4.1	1. Penyelesaian sebuah tugas.	

Hasil dari kuisioner yang telah diberikan kepada pengguna *admin* (Lampiran D) dengan mengujicoba dan mengevaluasi sistem sesuai dengan pertanyaan yang dibuat oleh pengembang akan dijelaskan pada Tabel 4.52.

Tabel 4.52 Hasil Kuisioner Pertanyaan *User Acceptance Testing* (UAT)

No.	ID	Jawaban
1.	Q1.1	Efisien
2.	Q1.2	Efisien
3.	Q2.1	Sedang
4.	Q2.2	Sedang
5.	Q2.3	Efektif
6.	Q2.4	Efektif
7.	Q2.5	Efektif
8.	Q3.1	Sedang
9.	Q3.2	Mudah
10.	Q4.1	Tinggi

4.4.4 Analisis Hasil Pengujian

Terhadap pengujian yang telah dilakukan, terjawab bahwa pengujian *white-box testing* berhasil dilakukan dengan tingkat kebenaran 100%. Ditinjau dari hasil perhitungan *cyclomatic complexity* yang dihitung dengan 3 (tiga) rumus dan menghasilkan hasil yang sama. Hasil dari pengujian *white-box testing* juga menghasilkan *independent path* atau jalur independent sebanyak 4 (empat).

Sedangkan pengujian *black-box testing* juga demikian, pengujian yang dilakukan menghasilkan tingkat validitas 100% . Yang berarti sistem yang dibangun telah mengimplementasikan seluruh kebutuhan fungsional yang telah disepakati dengan *stakeholder* pada tahap perencanaan sistem.

Hasil dari pengujian *User Acceptance Testing* (UAT) menunjukkan pada kriteria keefisienan sebuah sistem, sistem berfungsi secara efisiensi berdasarkan kualitas dari solusi untuk menyelesaikan masalah maupun berdasarkan penyelesaian masalah. Waktu yang didapatkan untuk menyesaikan tugas peminjaman maupun pengembalian buku masing-masing mendapatkan waktu kurang lebih 2 (dua) menit. Pada kriteria keefektifan sistem, sistem berfungsi secara efektif pada segi pemuatan tabel sheet yang tepat, pesan beserta manajemen data yang dinamis, dan kosistensi dan tata letak atau layout halaman. Sistem mendapatkan tingkat keefektifan pada skala sedang pada segi partisi informasi yang logis dan penempatan tautan yang tepat. Pada kriteria tingkat kepuasan pengguna kepada sistem, responden merasa puas pada segi kemudahan melakukan interaksi dengan sistem. Responden menjawab dengan skala sedang pada segi penyelesaian tugas. Pada kriteria *error*, sistem mendapatkan skala tinggi pada segi tingkat kesalahan yang dapat dideteksi pada sistem pada saat menyelesaikan sebuah tugas. Ditinjau dari jawaban responden yang telah menguji sistem yang tidak menunjukkan adanya jawaban “Tidak”, maka sistem menghasilkan kesimpulan *go-decision* yang artinya sistem telah dapat digunakan dan layak digunakan oleh pengguna.

BAB 5 PENUTUP

5.1 Kesimpulan

Kesimpulan akan dibuat berdasarkan hasil penelitian yang telah dilakukan oleh peneliti sebagai berikut:

1. Hasil dari analisis yang dilakukan peneliti pada iterasi ke-1 (satu) menghasilkan 2 (dua) pengguna (pengunjung dan admin). Dari 2 (dua) pengguna tersebut menghasilkan seluruhnya 8 (delapan) fitur dan 10 (sepuluh) kebutuhan fungsional. Pada setiap penggunga mempunyai kebutuhan sistem masin-masing. Pengguna pengunjung mempunyai 1 (satu) fitur sistem yaitu mencari buku dan pengguna admin mempunyai 7 (tujuh) fitur sistem yaitu *login,* mengelola data buku, melakukan peminjaman buku, melakukan pengembalian buku, melihat riwayat dan *logout*. Pada interasi ke-2 (dua) dilakukan perubahan fungsi export data menjadi cetak laporan.
2. Hasil perancangan dari melakukan proses pemodelan sistem menggunakan *Unified Modeling Language* (UML), *use case diagram* menghasilkan 2 (dua) *use case* yaitu pengunjung dan *admin. Class diagram* menghasilkan 5 (lima) kelas yaitu buku, penerbit, pengarang, transaksi, dan database. Hasil perancangan *database* menghasilkan *database* sistem dengan 6 (enam) tabel yaitu buku, pengarang, penerbit, transaksi, detail transaksi dan peminjam. Hasil implementasi sistem menghasilkan sebuah source code *C-Sharp* (C#) yang mengimplementasikan *class diagram* yang telah dirancang.
3. Pengujian yang dilakukan ialah pengujian sistem yang dilakukan dengan menggunakan metode *white-box testing, black-box testing* dan *User Acceptance Testing* (UAT).
    a. Hasil dari pengujian menggunakan metode *white-box* mendapatkan tingkat kebenaran 100% yang ditinjau dari hasil perhitungan *cyclomatic complexity* yang dihitung dengan 3 (tiga) rumus dan menghasilkan hasil perhitungan yang sama dan menghasilkan 4 (empat) *independent path*, yang menunjukkan sistem dapat digunakan oleh pengguna.
    b. Hasil dari pengujian menggunakan metode *black-box* mendapatkan tingkat validitas sistem sebesar 100% ditinjau dari hasil valid dari setiap *test case* yang diujikan.
    c. Hasil dari pengujian menggunakan *User Acceptance Tensting* (UAT), ditinjau dari jawaban responden yang telah menguji sistem yang tidak menunjukkan adanya jawaban “Tidak-”, maka sistem menghasilkan kesimpulan *go-decision* yang artinya sistem telah dapat digunakan dan layak digunakan oleh pengguna.

5.2 Saran

Saran akan dibuat berdasarkan hasil penelitian yang telah dilakukan oleh peneliti sebagai berikut:

1. Sistem dapat dikembangkan menjadi sistem dengan berbasis online (web) yang dapat diakses dimanapun dan kapanpun apabila sekolah menyediakan jaringan internet pada perpustakaan.

DAFTAR REFERENSI

Indrajani, 2015. *Database Design*. [e-book] Jakarta: Elex Media Komputindo. Tersedia di: Google Books <http://booksgoogle.com> [Diakses 7 September 2018].

Jogiyanto, 2002. *Analisis dan Desain Sistem Informasi*. Yogyakarta: Penerbit Andi.

Kubde, P., Sable, D., 2014. *The Role of Verification and Validation in System Development Life Cycle*. 2(2). International Journal of Research in Advent Technology.

Kumar, M., Singh, S.K., dan Dwivedi, R.K., 2015. *A Detail Study of Agile Software Developnment with Extreme Programming*. 5(10). International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE).

Kurniawan, T.A., 2018. *Pemodelan Use Case (UML): Evaluasi Terhadap Beberapa Kesalahan dalam Praktik*. 5(1). Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK).

Lasa, H. S., 2007. *Manajemen Perpustakaan Sekolah*. Yogyakarta: Pinus Book Publiser.

Munawar, (2005). *Pemodelan Visual dengan UML*. Yogyakarta: Graha Ilmu

Mustaqbal, M.S., Firdaus , R.S, dan Rahmadi, H., 2015. Pengujian Aplikasi Menggunakan *Black Box Testing Boundary Value Analysis*. 1(3). Jurnal Ilmiah Teknologi Informasi Terapan (JITTER).

Nama, G.F., Arnoldi, F., 2016. *Rancang Bangun Aplikasi Game Edukasi Pembelajaran Aksara Lampung Ajo dan Atu — Belajar Aksara Lampung Berbasis Android Dengan Sistem Multi-Ending Menggunakan Engine Ren'pf*. 3(4). Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK).

Nul, Lukman, 2013. *Ulasan Metodelogi Kualitatif : Wawancara Terhadap Elit*. 4(2). Pusat Pengkajian, Pengolahan Data dan Informasi (P3DI).

Pender, T., 2003. *UML Bible*. Indianapolis: Wiley Publishing, Inc.

Pratama, E.B., 2017. *Pendekatan Metodologi Extreme Programming pada Aplikasi eCommerce Berbasis M-Commerce*, [e-journall 5(2). Tersedia di: < http://ejournal.bsi.ac.id/> [Diakses 25 September 2018].

Pressman, R.S., 2012. *Rekayasa Perangkat Lunak*. Yogyakarta: ANDI.

Rahmi, R., Sari R.P., Suhatman, R., 2016. *Pendekatan Metodologi Extreme Programing pada Aplikasi E-Commerce*. 2(2). Jurnal Komputer Terapan.

Schneider, Geri dan Winters, Jason, 2001. *Applying Use Cases 2nd Edition: A Practical Guide*. Addison-Wesley.

Shodiqin, F.F., 2016 *Rancang Bangun Sistem Informasi Manajemen Perpustakaan pada Ruang Baca Fakultas Ilmu Komputer Universitas Brawijaya*. S1, Universitas Brawijaya.

Sommerville, I. (2003). *Software Engineering.* Boston: Pearson.

Sualim, S.A., Yassin, N.M., dan Mohamad, R., 2016. *Comparative Evaluation of Automated User Acceptance Testing Tool for Web Based Application*. 2(2). International Journal of Software Engineering and Technology (IJSET).

Sutarno, NS., 2006. *Manajemen Perpustakaan*.Jakarta: Sagung Seto.

Whitaker, RB,. 2012. *The C# Player's Guide*. [e-book] Logan: Starbound Software. Tersedia di: Google Books <http://booksgoogle.com> [Diakses 7 September 2018].

Widhiartha, Putu, 2008. Extreme Programming : Melakakukan Pengembangan Perangkat Lunak dengan Lebih Sederhana. [online] Tersedia di: <www.ilmukomputer.org> [Diakses 9 September 2018].

LAMPIRAN A WAWANCARA

A.1 Kerangka Wawancara

Daftar pertanyaan wawancara :

1. Apakah perpustakaan SMK 1 Muhammadiyah 1 Malang ini menggunakan komputerisasi atau masih manual?
2. Penyimpanan data transaksi untuk peminjaman atau pengembalian data duku menggunakan apa?
3. Bagaimana alur dari peminjaman buku serta pengembalian koleksi buku yang ada di perpustakaan dan berapa waktu yang diperlukan?
4. Apa saja tugas pustakawan di SMK 1 Muhammadiyah Malang ?
5. Apa saja pencatatan data buku pada pembukuan ?
6. Untuk data laporan, data apa saja yang dibutuh kan ? Berapa interval waktu untuk melakukan data laporan ?
7. Data apa saja yang dimasukkan kedalam laporan ?
8. Bagaimana perhitungan denda keterlambatan ?
9. Sistem seperti apa yang dibutuhkan oleh perpustakaan SMK 1 Muhammadiyah Malang ?

A.2 Hasil Wawancara

Judul	: Wawancara Kinerja Pustakawan
Narasumber	: Ibu Esti Hartini (Pustakawan)
Pewawancara	: Rizky Edyatna Putra
Lokasi	: Perpustakaan SMK 1 Muhammadiyah Malang
Tanggal	: 20 September 2018

Hasil wawancara :

1. Apakah perpustakaan SMK 1 Muhammadiyah 1 Malang ini menggunakan komputerisasi atau masih manual?
   Jawab: Pembuatan laporan dan transaksi yang terjadi di perpustakaan masih manual.
2. Penyimpanan data transaksi untuk peminjaman atau pengembalian data buku menggunakan apa?
   Jawab: Data transaksi untuk peminjaman buku dan pengembalian buku masih menggunakan buku folio bergaris besar untuk melakukan pencatatan semua transaksi buku.
3. Bagaimana alur dari peminjaman buku serta pengembalian koleksi buku yang ada di perpustakaan ?
   Jawab: Pertama-tama peminjam menanyakan buku apa yang ingin dipinjam, pustakawan mencari buku yang ingin dipinjam pada rak buku perpustakaan atau peminjam dapat mencari buku yang telah tersedia di rak buku, untuk transaksi buku, petugas meminta identitas dari peminjam dan kemudian melakukan pencatatan pada buku transaksi. Begitu sebaliknya untuk pengembalian buku dilakukan sama dengan cara mencatat pada buku transaksi dan buku dikembalikan dengan waktu yang didapat adalah kurang lebih 10 menit
4. Apa saja tugas pustakawan di SMK 1 Muhammadiyah Malang ?
   Jawab: Melayani transaksi peminjaman maupun pengembalian buku yang ada diperpustakaan, membuat laporan, mendata koleksi buku dan mengatur tata letak buku.
5. Apa saja pencatatan data buku pada pembukuan ?
   Jawab: Kode buku, judul buku, pengarang utama, tahun, penerbit, nomoer rak buku.
6. Untuk data laporan, data apa saja yang dibutuh kan ?
   Jawab: Transaksi peminjaman buku yang telah selesai dikembalikan kembali ke perpustakaan dengan interval setiap bulan.
7. Data apa saja yang dimasukkan kedalam laporan ?
   Jawab: Data yang dimasukkan kedalam laporan adalah data buku yang berupa judul buku, kode buku, tahun buku, data peminjam yang berupa nama peminjam, jabatan peminjam, nomor induk peminjam dan data transaksi yang berupa tanggal peminjaman buku, tanggal pengembalian buku serta jumlah denda.
8. Bagaimana perhitungan denda keterlambatan ?

Jawab: Perhitungan denda akan dilakukan setelah buku tidak dikembalikan setelah melewati batas hari yang ditentukan yaitu 2 (dua) hari. Keterlambatan pada kelipatan per-harinya yaitu sebanyak 1000 (seribu) rupiah.

9. Sistem seperti apa yang dibutuhkan oleh perpustakaan SMK 1 Muhammadiyah Malang ?
   Jawab: Sistem yang mempermudah pekerjaan pustakawan dalam proses transaksi peminjaman buku maupun pengembalian buku dan pembuatan laporan.

Malang, 20 Juni 2013
Narasumber

Ibu Esti Hartini
NUPTK. 7761746648300012

LAMPIRAN B SURAT PERNYATAAN VERIFIKASI

SURAT PERNYATAAN VERIFIKASI ANALISIS

KEBUTUHAN SISTEM SEBAGAI INSTRUMEN PENELITIAN

Saya yang bertanda tangan dibawah ini :

Nama : Esty Hartini

NUPTK : 7761746648300012

Jabatan : Staff Perpustakaan

menyatakan bahwa kebutuhan sistem yang telah dianalasis dan dibuat atas nama mahasiswa dalam skripsi :

Nama : Rizky Edyatna Putra

NIM : 156150601111014

Prodi : Pendidikan Teknologi Informasi

Judul Skripsi : Pengembangan Sistem Informasi Perpustakaan Menggunakan Metode *Extreme Programming.*

Setelah dilakukan kajian dan review atas kebutuhan sistem atau fungsionalitas sistem yang telah dibuat oleh peneliti dinyatakan :

☐ Layak digunakan dalam sistem

☑ Layak digunakan dalam sistem dengan perbaikan

Penggantian fungsi "export data" menjadi "cetak laporan".

☐ Tidak layak digunakan dalam sistem

Demikian agar dapat digunakan sebagaimana mestinya.

Malang, 27 Mei 2019

Verifikator

Esty Hartini

NUPTK. 7761746648300012

Catatan:

☐ Beri tanda centang (✔)

LAMPIRAN C SURAT PERNYATAAN VALIDASI

SURAT PERNYATAAN VALIDASI

PENGUJIAN DAN PENERAPAN SISTEM INFORMASI PERPUSTAKAAN

DI SMK 1 MUHAMMADIYAH MALANG

Saya yang bertanda tangan dibawah ini :

Nama : Esty Hartini

NUPTK : 7761746648300012

Jabatan : Staff Perpustakaan

menyatakan bahwa sistem yang telah dibangun oleh mahasiswa dalam skripsi :

Nama : Rizky Edyatna Putra

NIM : 156150601111014

Prodi : Pendidikan Teknologi Informasi

Judul Skripsi : Pengembangan Sistem Informasi Perpustakaan Menggunakan Metode *Extreme Programming.*

Setelah dilakukan pengujian dan penerapan sistem yang telah dibuat oleh peneliti dinyatakan:

☑ Sistem layak digunakan

☐ Sistem layak digunakan dengan perbaikan

☐ Sistem tidak layak digunakan

Catatan:

.....

.....

.....

Demikian agar dapat digunakan sebagaimana mestinya.

Malang, 20 Juli 2019

Validator

Esty Hartini

NUPTK. 7761746648300012

Catatan:

☐ Beri tanda centang (✔)

LAMPIRAN D HASIL KUISIONER

KUISIONER SISTEM INFORMASI

PERPUSTAKAAN DI SMK 1 MUHAMMADIYAH MALANG

Nama : Esty Hartini

Status : Staff Perpustakaan

Jawablah pertanyaan kuisioner dibawah ini dengan mencentang jawaban (☑) yang tersedia.

No.	Pertanyaan	Jawaban		
1.	Seberapa baik tingkat akurasi dan kelengkapan aplikasi berbasis desktop dapat memberikan informasi dan fungsionalitas sesuai dengan:	**Efisien**	**Sedang**	**Tidak Efisien**
	1.1 Kualitas dari solusi untuk menyelesaikan permasalahan beserta waktu yang didapatkan	☑	☐	☐
	1.2 Penyeleseaian permasalahan	☑	☐	☐
2.	Seberapa efektif aplikasi berbasis desktop itu berdasarkan:	**Efektif**	**Sedang**	**Tidak Efektif**
	2.1 Partisi informasi yang logis	☐	☑	☐
	2.2 Penempatan tautan yang tepat	☐	☑	☐
	2.3 Pemuatan tabel sheet yang tepat	☑	☐	☐
	2.4 Pesan dan manajemen data yang dinamis	☑	☐	☐
	2.5 Konsistensi dan tata letak atau layout halaman	☑	☐	☐
3.	Berapa tingkat kesulitan yang dapat ditemukan pengguna dalam aplikasi berdasarkan	**Mudah**	**Sedang**	**Tidak Mudah**
	3.1 Penyelesaian sebuah tugas	☐	☑	☐
	3.2 Melakukan interaksi pada sistem	☑	☐	☐
4.	Seberapa maksimum tingkat kesalahan dapat dideteksi pada aplikasi berdasarkan	**Tinggi**	**Sedang**	**Rendah**
	4.1 Penyelesaian sebuah tugas	☑	☐	☐

Catatan :

Waktu penyelesaian tugas pengembalian dan peminjaman buku diselesaikan dengan waktu kurang lebih (+-) 2 menit

Malang, 20 JULI 2019
Staff Perpustakaan

Esty Hartini
NUPTK. 7761746648300012