BAB III
METODE PENELITIAN

3.1 Objek Penelitian

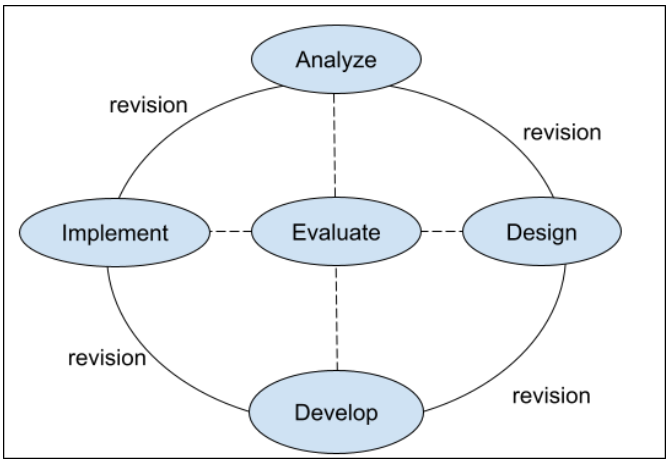
Objek penelitian dalam penelitian ini adalah sistem yang dikembangkan untuk mengelola hak cipta dan transaksi *e-book* menggunakan teknologi *blockchain* yang berbasis *website*. Sistem yang akan dirancang dan dikembangkan ini mencakup fitur-fitur seperti manajemen hak cipta *e-book*, pencatatan transaksi, dan manajemen izin akses. Sistem ini akan diimplementasikan dalam sebuah *website* yang memfasilitasi aksesibilitas dan penggunaan yang lebih mudah tanpa instalasi perangkat lunak tambahan.

3.2 Jenis Penelitian dan Metode yang Digunakan

Jenis penelitian yang digunakan pada penelitian ini adalah penelitian desain dan pengembangan atau *design and development* (D&D). Model pengembangan yang dipakai adalah menurut Richey dan Klein (2007) yaitu *"The systematic study of design, development and evaluation processes with the aim of establishing an empirical basis for the creation of instructional and non-instructional products and tools and new or enhanced models that govern their development."*. Dalam pengembangan ini, digunakan bahasa pemrograman Node.js dengan *gas* Express.js untuk web konvensional dan bahasa pemrograman Solidity dengan *gas* Hardhat untuk web *blockchain*. Seluruh pengembangan dilakukan menggunakan editor Visual Studio Code, dengan metode pengujian Blackbox Testing untuk menguji setiap input pada *website* BlockBook dan dengan metode pengujian Blackbox Testing untuk menguji setiap input pada *website* BlockBook dan metode pengujian TDD (Test Drive Development) untuk fungsionalitas *smart contract*.

3.3 Prosedur Penelitian

Prosedur penelitian yang digunakan oleh peneliti adalah ADDIE model menurut Branch (2009). Terdapat lima tahapan penelitian model ADDIE menurut Branch (2009) dalam (Asmayanti et al, 2020) yaitu *Analyze, Design, Develop, Implement,* dan *Evaluate.*



Gambar 3.1 Prosedur ADDIE

1. Tahap analisis (Analyze) merupakan analisis permasalahan.
2. Tahap desain (Design) dikenal dengan istilah membuat rancangan.
3. Tahap Pengembangan (Development) merupakan proses mewujudkan desain menjadi aplikasi.
4. Tahap Implementasi (Implement) merupakan langkah untuk menerapkan serta pengujian aplikasi web yang telah di kembangkan.
5. Tahap Evaluasi (Evaluate) dilakukan disetiap tahap.

Setiap tahapan penelitian pengembangan ini dijelaskan sebagai berikut (Sigit et al, 2021).

3.3.1 Tahap *Analyze* (Analisis)

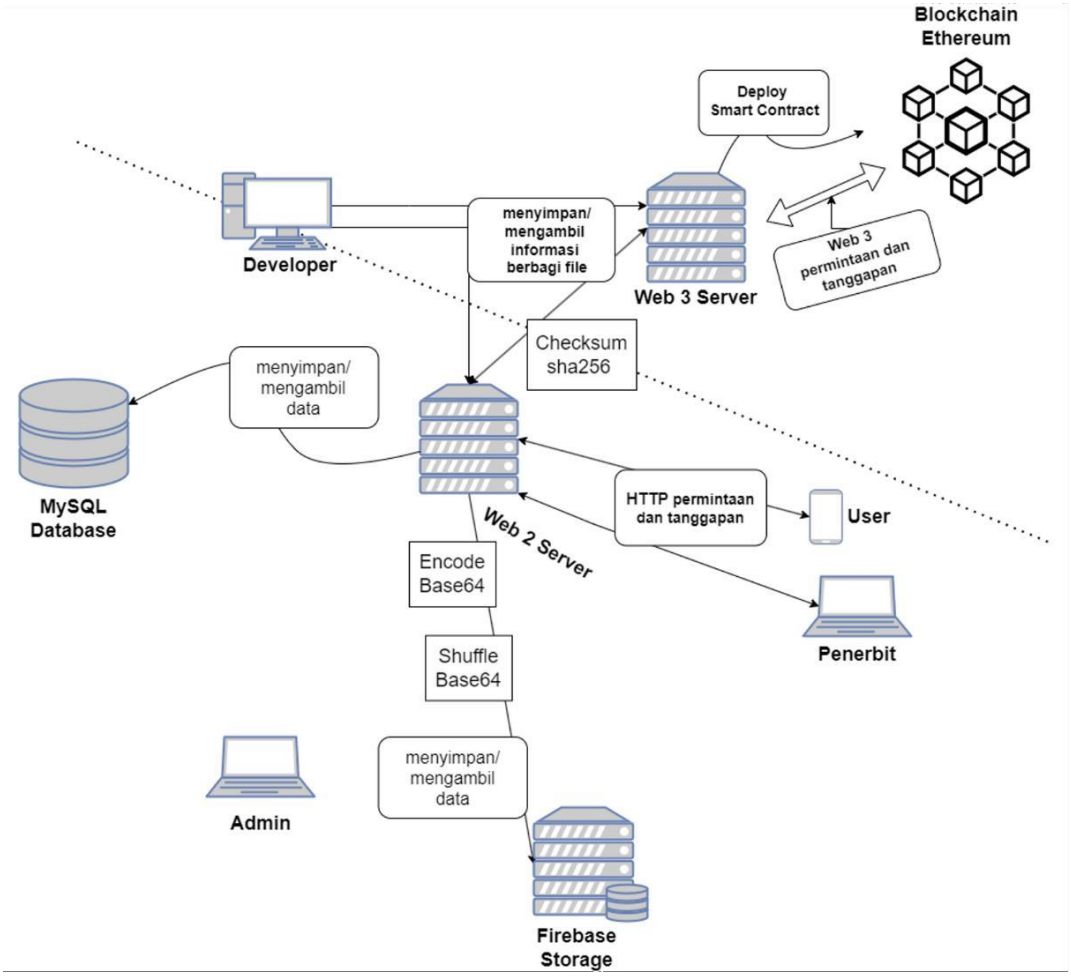
Tahap pertama yang dilakukan pada penelitian ini adalah menganalisis permasalahan yang terjadi mengenai perlindungan hak cipta *e-book* dari pelanggaran dan pencurian, tantangan dalam validasi transaksi *e-book* secara digital dan aman, serta kompleksitas integrasi teknologi *blockchain* dalam sistem manajemen hak cipta dan transaksi *e-book* berbasis web. Tahap analisis juga membahas mengenai kelayakan teknologi *blockchain* dalam membuat sistem manajemen yang dikembangkan disertai dengan keunggulan dari teknologi *blockchain*. Selain itu terdapat susunan rencana arsitektur yang mencakup komponen - komponen utama dalam perancangan *website* untuk pengimplemnetasian sistem manajemen hak cipta dan transaksi *e-book* serta integrasi antara *website* dan teknologi *blockchain*.

3.3.2 Tahap *Design* (Perancangan)

Pada tahap ini peneliti melakukan perancangan sistem manajemen hak cipta dan transaksi *e-book* berbasis *website* berdasarkan hasil analisis. Terdapat desain arsitektur sistem secara keseluruhan. Meliputi arsitektur diagram, *use case diagram*, *Activity* diagram, *flowchart*, desain database dan pemilihan platform *blockchain*.

3.3.2.1 Arsitektur Diagram

Terlihat bahwa arsitektur diagram pada penelitian kali ini dibuat berbeda, dikarenakan terbagi menjadi 2 bagian yaitu web server 2 dan web server 3 pada gambar 3.2. Bagian tersebut dijelaskan sebagai berikut.



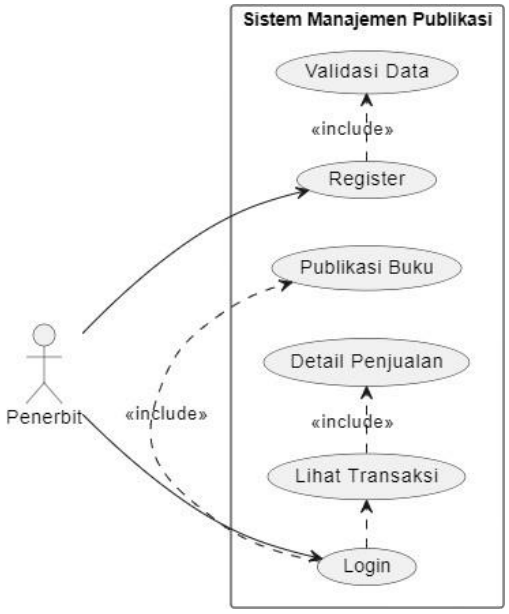
Gambar 3.2 Arsitektur Diagram Sistem BlockBook

Arsitektur diagram sistem diatas menunjukan bahwa terdapat Developer yang berinteraksi dengan Web 3 Server untuk menyimpan atau mengambil informasi terkait berbagi *file*. Web 3 Server ini terhubung dengan Node *Blockchain* dimana

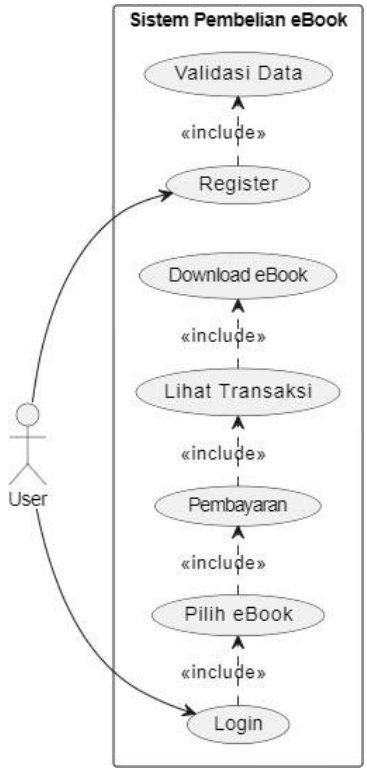
smart contract dapat dideploy. Di sisi lain, *User* dan Penerbit berinteraksi dengan sistem melalui HTTP Request and Response yang ditangani oleh Web 2 Server, yang juga bertanggung jawab menyimpan dan mengambil data dari MySQL Database dan Firebase Storage. Sedangkan untuk Web 2 server dan Web 3 server berhubungan langsung satu sama lain antar server, ini mencerminkan kombinasi basis data relasional tradisional dan penyimpanan berbasis cloud, memfasilitasi operasi yang beragam seperti pengelolaan data pengguna, konten, dan transaksi. Arsitektur ini menunjukkan integrasi antara infrastruktur server lama dan teknologi *blockchain* terbaru, yang memungkinkan pengembangan aplikasi yang lebih aman, desentralisasi, dan efisien.

3.3.2.2 Use case diagram

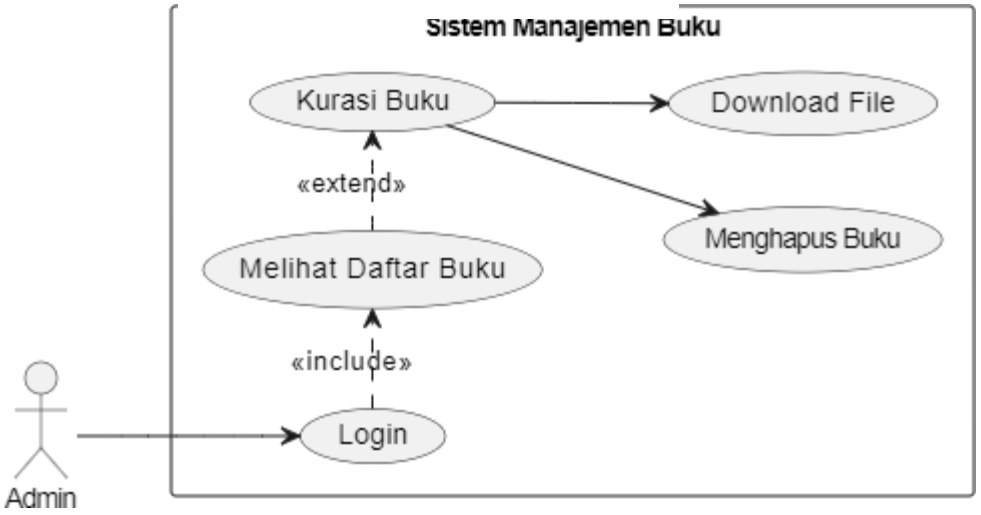
Pada bagian ini merupakan *use case digram* pada penilitian kali ini, *use cas diagram* terdiri dari sistem manajemen publikasi pada gambar 3.3 lalu terdapat sistem pembelian e-*book* pada gambar 3.4 dan *use case diagram* kurasi admin pada gambar 3.5. selain itu terdapat tahapan *activity* untuk *role user*, *role* admin dan *role* penerbit. Bagian tersebut dijelaskan sebagai berikut.



Gambar 3.3 Sistem Manajemen Publikasi



Gambar 3.4 Sistem Pembelian e-book



Gambar 3. 5 Use case diagram

Terdapat 3 aktor penting dalam berjalannya *website* BlockBook, terlihat pada gambar 3.3 diantaranya ada sistem manajemen publikasi , lalu terlihat pada gambar 3.4 terdapat sistem pembelian *e-book* oleh *role user* dan 3.5 merupakan *use case diagram* admin. Setiap aktor memiliki peranan masing-masing diantranya :

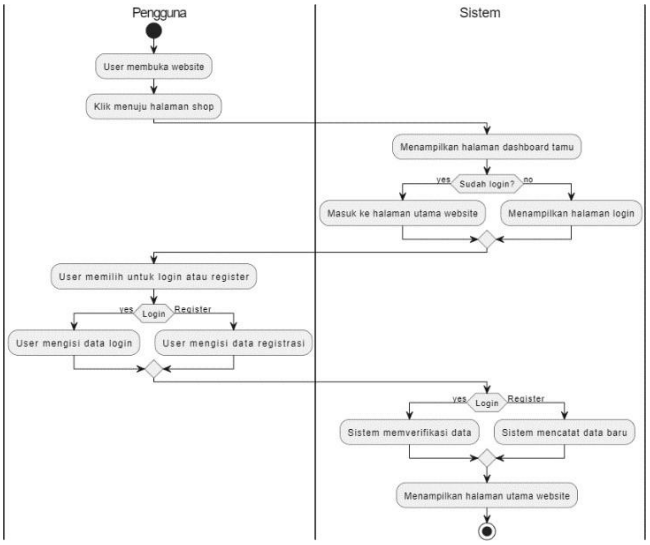
1. *User* memulai dengan kemampuan untuk *login* dan *logout*, memastikan bahwa akses mereka ke sistem terjaga dan pribadi. Setelah masuk, mereka dapat membeli ***e-book***, memilih dari katalog yang tersedia, yang memudahkan mereka mendapatkan konten digital secara instan. Mereka juga dapat melihat transaksi yang telah dilakukan, memberikan transparansi dan rekam jejak pembelian mereka. Setelah pembelian, *user* memiliki opsi untuk mendownload buku, memungkinkan mereka mengakses bacaan di perangkat mereka secara offline.
2. Admin memiliki kemampuan untuk *login* dan *logout* dari sistem untuk memulai dan mengakhiri sesi mereka. Sebagai bagian dari tugas administratif, mereka melakukan seleksi *e-book* yang diupload oleh penerbit, memastikan hanya konten yang memenuhi standar yang ditampilkan kepada *user*. Admin juga dapat menghapus buku dari sistem untuk alasan seperti konten yang tidak sesuai atau usang. Selain itu, admin bertanggung jawab untuk membuat buku tampil kepada *user*, menyelesaikan proses kurasi dan memastikan ketersediaan buku. Admin juga dapat melihat transaksi untuk memonitor penjualan dan mengelola aspek keuangan dari platform.
3. Penerbit, di sisi lain, bertanggung jawab atas upload *e-book*, memasukkan karya mereka ke dalam sistem untuk dijual. Mereka memulai dengan *login* dan *logout* untuk mengelola sesi kerja mereka. Selain mengupload, penerbit juga dapat melihat transaksi yang terkait dengan *e-book* yang telah mereka publikasikan, memberikan wawasan tentang performa penjualan dan preferensi pasar.

Setiap aktor memiliki serangkaian tanggung jawab yang terhubung melalui berbagai fungsi sistem yang memfasilitasi interaksi mereka, tidak hanya antara satu sama lain tetapi juga dengan sistem penjualan buku online secara keseluruhan. Diagram ini membantu menggambarkan proses operasional dan alur kerja dalam ekosistem digital ini, memudahkan pengertian tentang bagaimana data dan tugas mengalir melalui aplikasi *website*.

3.3.2.3 *Activity* Diagram

Terdapat *Activity* diagram sesuai perancangan pada penelitian kali ini, *Activity* diagram meliputi fitur yang tersedia untuk admin, penerbit dan *user*. *Activity* diagram dijelaskan sebagai berikut :

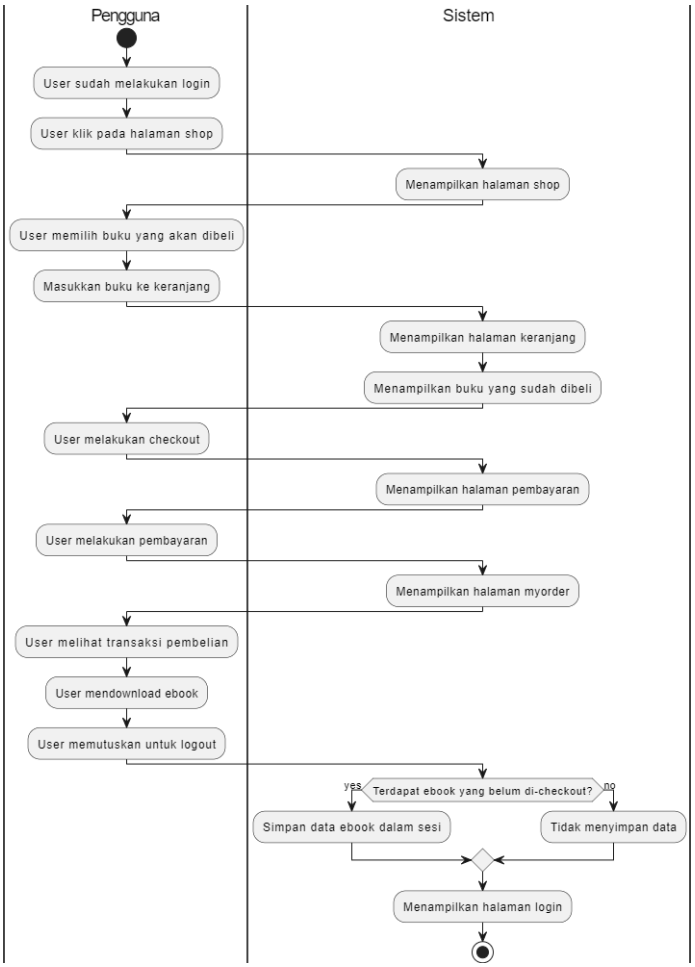
1. **Tahapan *Activity User***



Gambar 3.6 Activity User

Dalam konteks mengakses *website* untuk melakukan pembelian, interaksi antara pengguna dan sistem digambarkan dalam *Activity user* di Gambar 3.6. Diagram ini dimulai dengan pengguna membuka *website* dan mengklik halaman *shop*. Sistem kemudian menampilkan halaman *dashboard* tamu untuk mengetahui apakah pengguna sudah *login*. Jika mereka belum, pengguna diarahkan ke halaman *login* untuk memilih masuk atau mendaftar.

Pengguna diminta untuk memasukkan data *login* mereka jika mereka memilih untuk *login*, atau mereka diminta untuk memasukkan data registrasi mereka jika mereka memilih untuk registrasi. Setelah data dimasukkan, sistem akan melakukan verifikasi data pada pilihan *login* atau mencatat data baru pada pilihan registrasi. Setelah proses ini selesai, halaman utama situs web akan dibuka, memungkinkan pengguna untuk melanjutkan aktivitas mereka di sana.

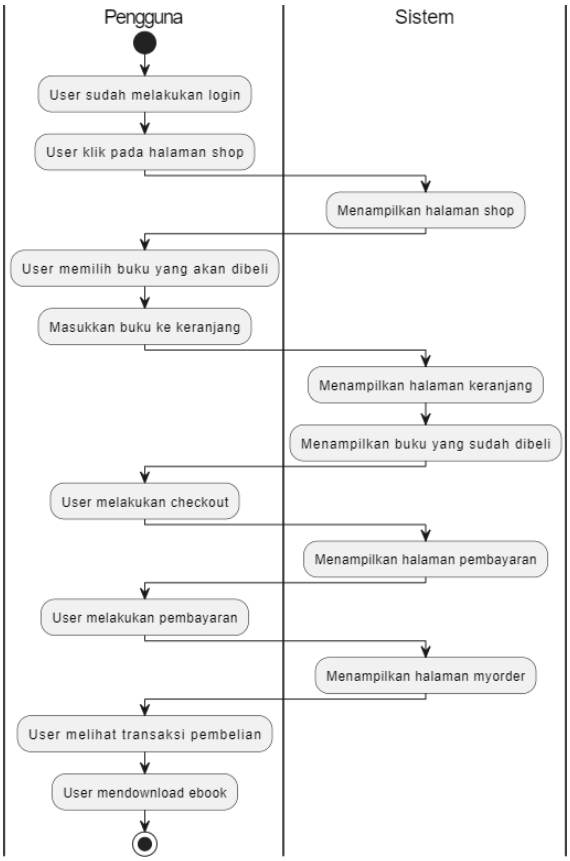


Gambar 3.7 Activity User Login Logout

Proses pembelian dan manajemen buku yang dilakukan oleh pengguna yang telah masuk ke dalam sistem *shop* buku digambarkan dalam aktivitas diagram di Gambar 3.7. Proses ini dimulai dengan pengguna melakukan klik pada halaman shop, yang mengarahkan sistem untuk menampilkan halaman tersebut. Selanjutnya, pengguna memilih buku yang mereka inginkan dan memasukkannya ke dalam keranjang belanja, di mana sistem menampilkan halaman keranjang yang menunjukkan buku tersebut. Setelah memilih buku, pengguna melanjutkan ke checkout dan sistem menampilkan halaman pembayaran, di mana mereka dapat membayar buku yang telah dipilih. Setelah pembayaran selesai, sistem menampilkan halaman myorder, di mana pengguna dapat melihat detail transaksi dan mendownload buku yang telah dibeli, jika diinginkan.

Selanjutnya, jika pengguna memutuskan untuk *logout*, sistem akan memeriksa apakah masih ada ebook dalam keranjang yang belum di-checkout. Nantinya sistem akan

menyimpan data keranjang pada sesi *user* supaya saat *user* kembali *login* maka data keranjang tetap ada.

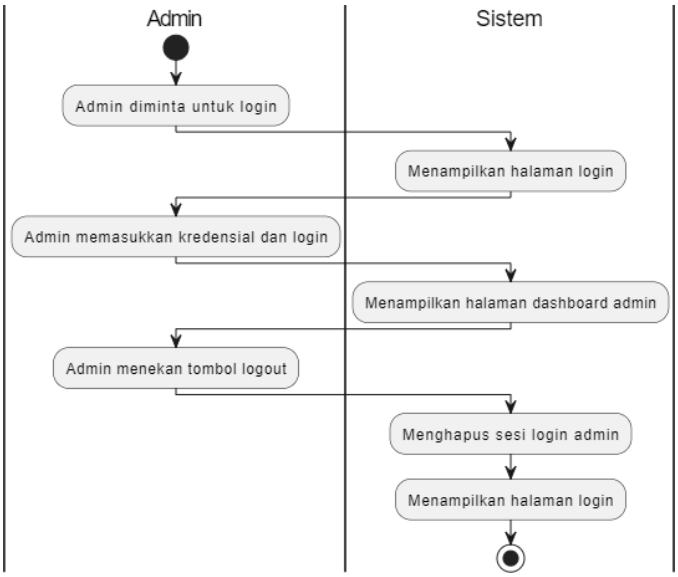


Gambar 3.8 Activity User Transaksi Buku

Dalam Gambar 3.8, aktivitas diagram menunjukkan bagaimana pengguna yang telah masuk ke sistem *shop* melakukan pembelian buku. Diagram ini dimulai dengan pengguna mengklik tautan yang sesuai untuk mengakses halaman *shop*, yang kemudian menampilkan halaman *shop*, di mana pengguna dapat memilih buku yang ingin mereka beli. Setelah memilih buku, sistem menampilkan halaman keranjang yang menampilkan daftar semua buku yang telah dipilih oleh pengguna. Kemudian, pengguna menekan tombol atau link pembayaran untuk melanjutkan proses pembayaran, dan sistem menampilkan halaman pembayaran.

Pengguna diminta untuk menyelesaikan transaksi pada halaman pembayaran dengan membayar. Setelah pembayaran berhasil, mereka dibawa ke halaman myorder, tempat mereka dapat melihat semua transaksi yang telah mereka lakukan. Akhirnya, pengguna memiliki opsi untuk melihat detail transaksi dan mendownload ebook yang telah dibeli.

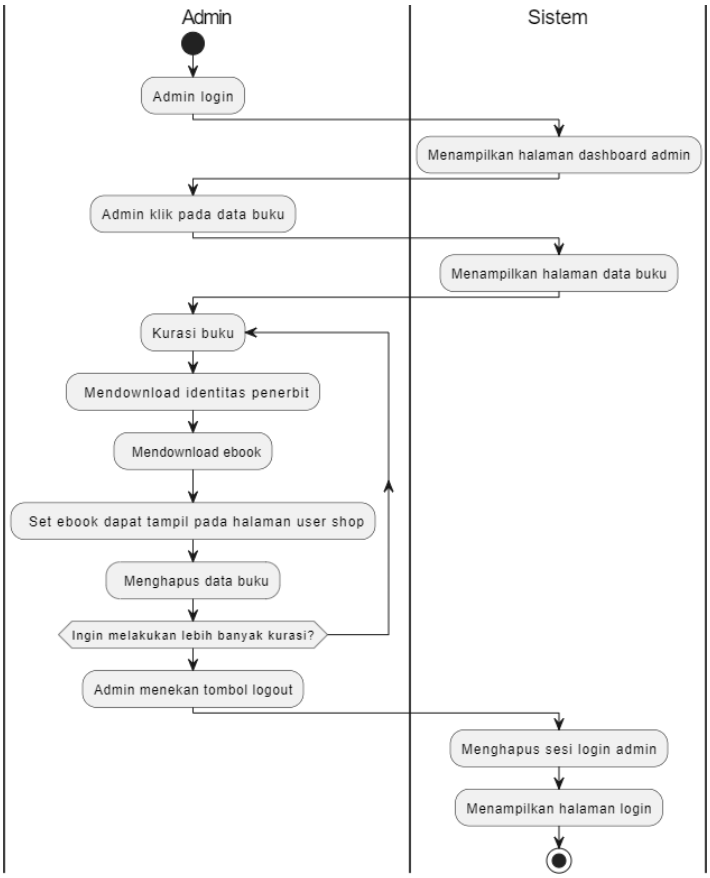
2. Tahapan *Activity* Admin



Gambar 3. 9 Admin Login Logout

Proses yang dilakukan oleh administrator dalam sebuah sistem manajemen digambarkan dalam aktivitas diagram di Gambar 3.9. Ketika manajer diminta untuk *login*, proses ini dimulai. Sistem menanggapi permintaan ini dengan menampilkan halaman *login*, di mana manajer memasukkan kredensialnya. Setelah mengisi data yang benar, manajer mengklik tombol *login*. Selanjutnya, sistem memverifikasi kredensial dan menampilkan halaman dashboard admin. Halaman ini biasanya berisi alat dan informasi yang berkaitan dengan tugas administratif seperti mengelola pengguna, mengatur sistem, dan melacak aktivitas sistem.

Setelah selesai dengan aktivitas administratif, admin dapat memilih untuk *logout* dari sistem dengan menekan tombol *logout*. Sistem akan merespons dengan menghapus sesi *login* admin, memastikan bahwa semua informasi sesi yang sensitif telah dibersihkan, sehingga menjaga keamanan sistem. Langkah terakhir dalam proses ini adalah sistem yang menampilkan kembali halaman *login*, siap untuk sesi *login* berikutnya oleh admin yang sama atau pengguna lain.



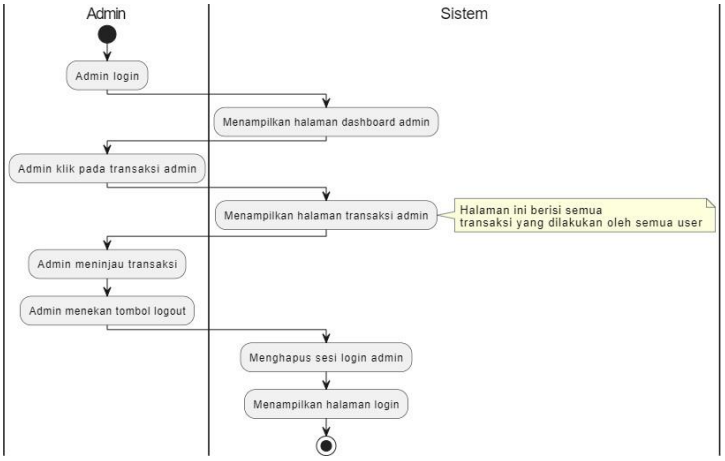
Gambar 3.10 Activity Admin Kurasi Buku

Alur kerja administrator untuk mengelola data buku pada sistem manajemen konten buku digambarkan dalam aktivitas diagram 3.10. Alur kerja dimulai dengan pengguna melakukan *login* ke sistem, yang menghasilkan halaman dashboard admin. Dari *dashboard* ini, pengguna dapat mengakses halaman data buku dengan mengklik opsi yang sesuai.

Setelah masuk ke halaman data buku, admin harus menyelesaikan sejumlah tugas kurasi, seperti memilih buku untuk dikurasi, mendownload identitas penerbit yang terkait dengan buku tersebut, dan mendownload ebook itu sendiri. Setelah mendownload, admin dapat mengatur buku untuk ditampilkan di halaman toko pengguna, yang menandakan bahwa buku tersebut disetujui untuk dijual atau dibagikan kepada pengguna.

Admin juga memiliki opsi untuk menghapus data buku jika diperlukan. Setelah kurasi, sistem menanyakan apakah admin ingin melakukan lebih banyak kurasi atau tidak. Jika admin memilih untuk tidak melakukan lebih banyak kurasi, mereka

dapat menekan tombol *logout*, yang mengarahkan sistem untuk menghapus sesi *login* admin dan kembali menampilkan halaman *login*.

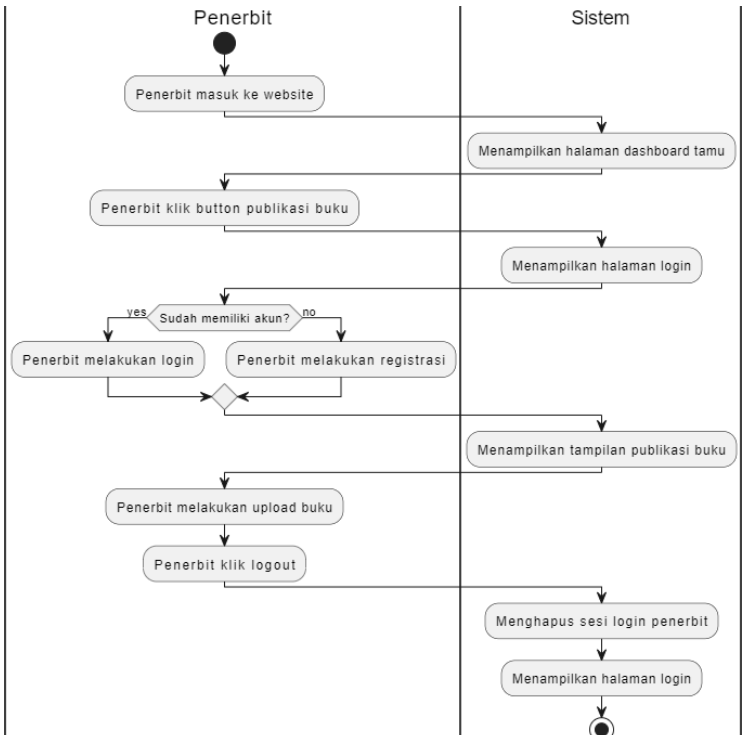


Gambar 3.11 Activity Admin Transaksi

Proses administratif yang dilakukan administrator untuk mengelola dan memantau transaksi yang dilakukan oleh semua pengguna yang terdaftar dalam sistem digambarkan dalam aktivitas diagram di Gambar 3.11. Proses ini dimulai dengan pengguna *login*, yang memicu sistem untuk menampilkan halaman *dashboard* administrator. Setelah masuk, administrator harus memilih pilihan "transaksi admin" di *dashboard*. Ini akan membuka halaman sistem responsif yang menampilkan rincian semua transaksi yang telah dilakukan oleh pengguna dalam sistem. Ini memungkinkan admin untuk meninjau dan mengawasi aktivitas transaksi yang terjadi.

Admin kemudian dapat meninjau transaksi-transaksi tersebut untuk memastikan semuanya berjalan sesuai dengan ketentuan dan kebijakan sistem atau untuk melakukan analisis keuangan. Setelah selesai meninjau transaksi, admin memutuskan untuk *logout* dari sistem. Sistem kemudian menanggapi aksi ini dengan menghapus sesi *login* admin untuk memastikan keamanan data, sebelum akhirnya kembali menampilkan halaman *login*, siap untuk sesi *login* berikutnya oleh admin yang sama atau pengguna lain.

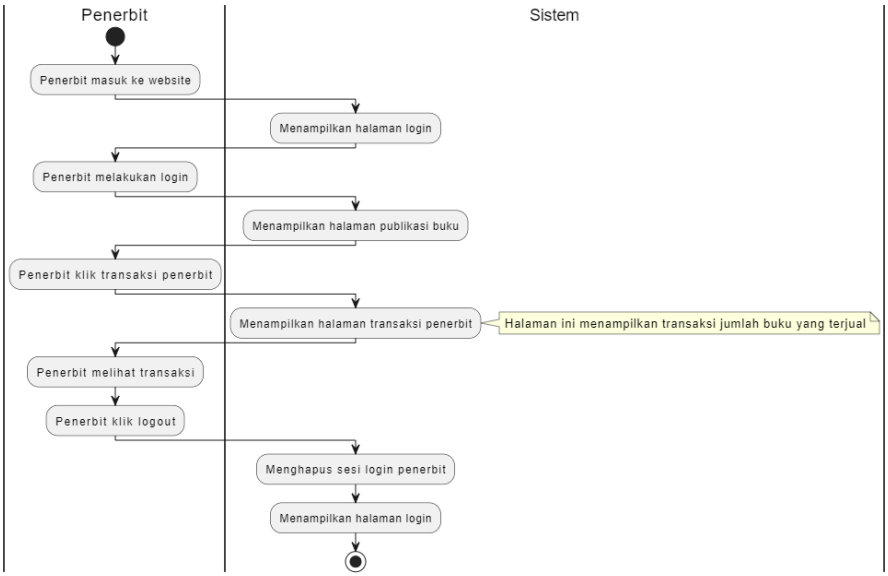
3. Tahapan *Activity* Penerbit



Gambar 3. 12 Activity Penerbit Publikasi Buku

Dalam sistem publikasi buku, aktivitas diagram Gambar 3.12 menunjukkan langkah-langkah yang dilakukan oleh seorang penerbit dari saat mereka memasuki *website* hingga *logout*. Proses ini dimulai saat sistem menampilkan halaman dashboard tamu sebagai halaman awal. Setelah masuk ke *dashboard*, penerbit harus mengklik tombol publikasi buku. Ini membawa sistem ke halaman *login*, di mana penerbit dihadapkan pada pilihan untuk memasukkan akun jika sudah memilikinya atau melakukan registrasi jika belum. Penerbit dapat mengakses halaman publikasi buku jika mereka memilih untuk *login* dan memasukkan kredensial mereka. Jika mereka belum memiliki akun, penerbit harus melakukan registrasi dengan mengisi data yang diperlukan, dan sistem akan mengarahkan mereka ke halaman publikasi.

Setelah berhasil *login* atau registrasi, penerbit dapat melakukan upload buku ke dalam sistem. Ini mencakup mengunggah *file* dan informasi terkait buku yang ingin dipublikasikan. Setelah proses upload selesai, penerbit dapat memutuskan untuk *logout* dari sistem. Sistem kemudian menghapus sesi *login* penerbit untuk menjaga keamanan dan privasi, dan akhirnya menampilkan kembali halaman *login*, siap untuk *login* berikutnya atau pengguna lain.

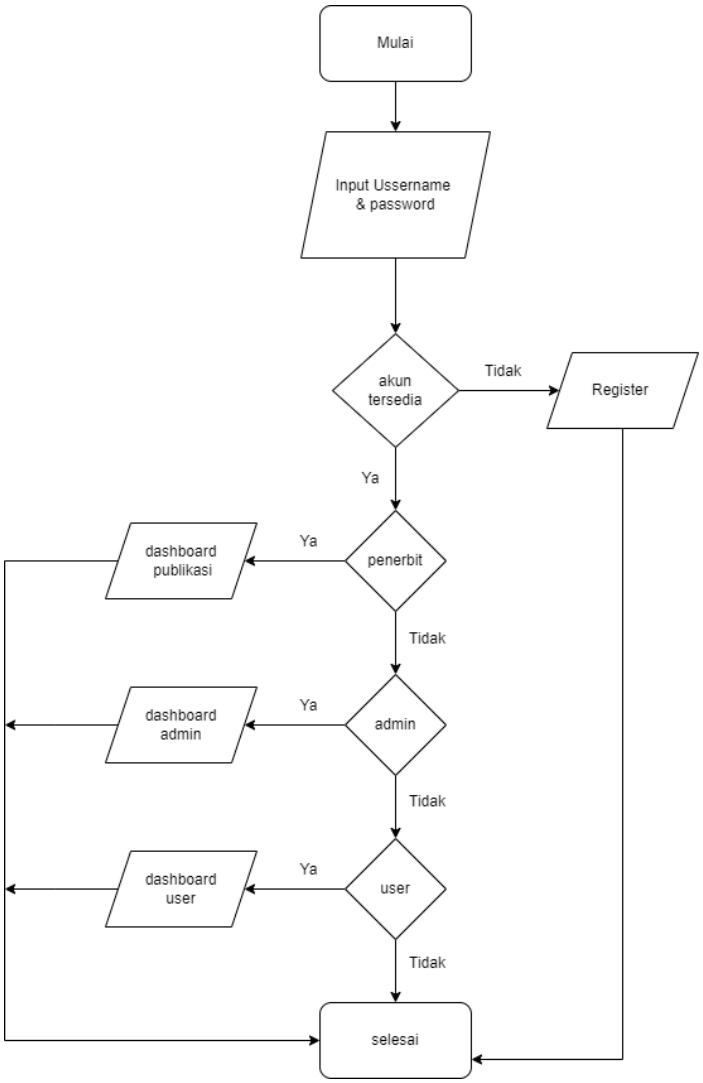


Gambar 3.13 Activity Penerbit Transaksi

Gambar 3.13 menunjukkan proses interaksi seorang penerbit dengan sistem publikasi buku online, mulai dari memasuki situs web hingga melakukan *logout*. Proses ini dimulai ketika penerbit memasuki situs web, di mana sistem menampilkan halaman *login* di mana mereka harus memasukkan informasi pribadi mereka. Setelah mereka memasukkan informasi pribadi mereka, sistem mengarahkan mereka ke halaman publikasi buku. Penerbit dapat mengelola atau menambahkan buku baru yang ingin mereka publikasikan di halaman ini. Penerbit juga dapat mengklik opsi transaksi penerbit untuk melihat transaksi yang terkait dengan buku-buku yang telah dipublikasikan. Ini mengarahkan sistem untuk menampilkan halaman transaksi penerbit, yang menunjukkan semua aktivitas penjualan yang terkait dengan buku-buku penerbit, termasuk jumlah buku yang telah terjual, memberikan wawasan penting tentang kinerja penjualan.

3.3.2.4 *Flowchart* Sistem

Flowchart pada penelitian kali ini, menggambarkan tentang alur cara kerja dari *login, Register*, admin, *user* dan penerbit untuk menunjukan bahwa alur kerja dari semua sistem diatas berhubungan antara satu sama lain

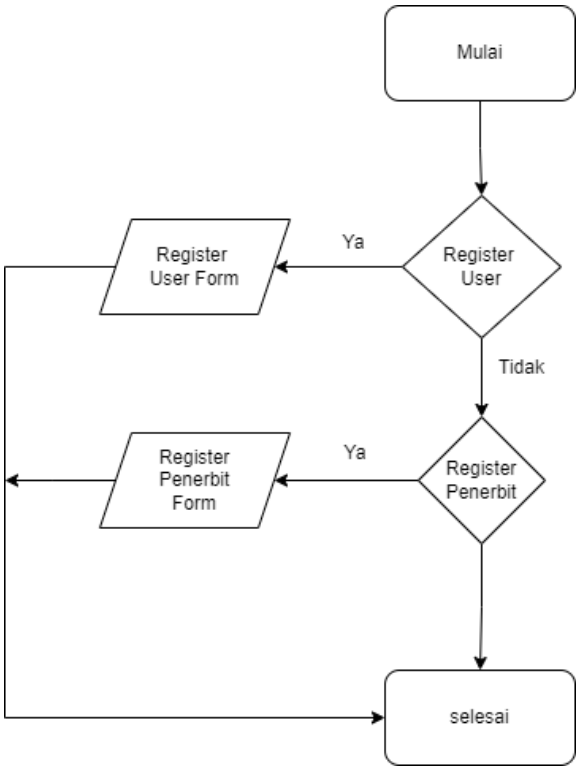


Gambar 3.14 Flowchart Login

Gambar 3.14 menunjukkan *flowchart* terperinci yang menjelaskan proses *login* dan penugasan *dashboard* berdasarkan peran pengguna dalam sistem manajemen konten. Saat pengguna memulai sesi mereka, mereka langsung diminta untuk memasukkan *user*name dan password mereka. Jika sistem tidak dapat menemukan akun pengguna, pengguna diarahkan untuk mendaftar. Jika tidak, ini menunjukkan bahwa akun tersebut belum terdaftar atau kredibilitasnya salah.

Sistem mengidentifikasi jenis akun pengguna setelah *login* untuk memilih tampilan *dashboard* yang sesuai. Pengguna yang memiliki peran penerbit akan diarahkan ke *dashboard* publikasi, di mana mereka dapat mengelola dan mempublikasikan karya mereka. Pengguna dengan peran admin juga dapat mengakses *dashboard* admin, yang memiliki alat kontrol untuk mengelola pengguna dan konten secara keseluruhan. Sedangkan, pengguna biasa diarahkan ke *dashboard user* yang menawarkan fitur dan

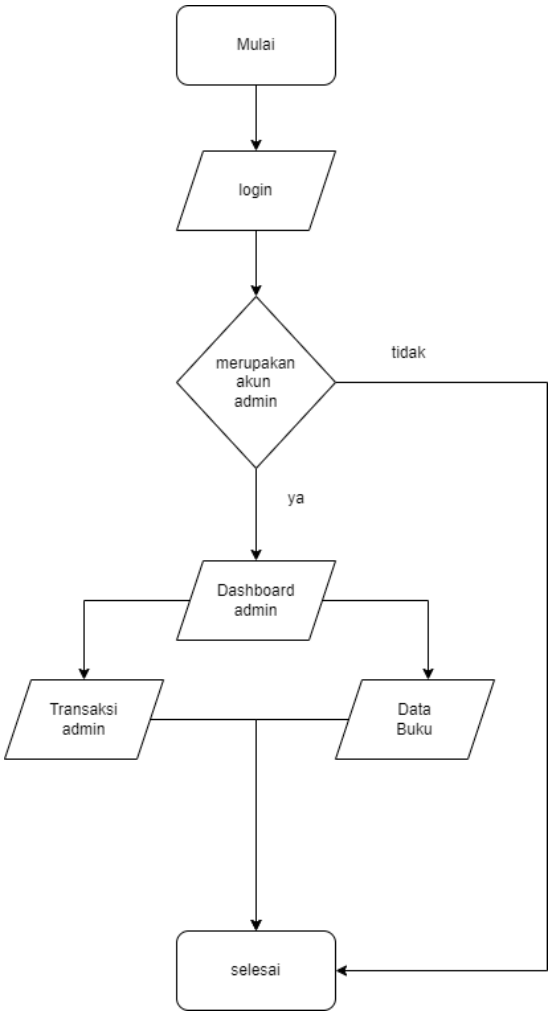
fungsi sesuai dengan akses pengguna umum, seperti melihat dan berinteraksi dengan konten. Setiap sesi diakhiri dengan pengguna yang menyelesaikan tugas mereka di *dashboard* masing-masing dan proses diakhiri.



Gambar 3.15 Flowchart Register

Proses pendaftaran untuk dua jenis pengguna sistem umum dan penerbit digambarkan dalam *flowchart* rinci di Gambar 3.15. Proses ini dimulai dengan tahap "Mulai", yang membawa pengguna langsung ke langkah pengambilan keputusan untuk mendaftar. Pada awalnya, pengguna diberi opsi untuk mendaftar sebagai *user* biasa dengan mengisi formulir "Daftar *User*". Jika pengguna memilih untuk mendaftar sebagai *user*, sistem memproses informasi tersebut melalui "Daftar *User*", dan jika proses pendaftaran berhasil, maka berlanjut ke tahap "selesai". Namun, jika pengguna memilih untuk tidak mendaftar sebagai *user* biasa, sistem akan memberikan opsi untuk mendaftar sebagai penerbit.

Jika pengguna memilih untuk mendaftar sebagai penerbit, mereka harus mengisi formulir "Daftar Penerbit". Seperti sebelumnya, sistem memproses pilihan pengguna melalui "Daftar Penerbit", dan proses diakhiri jika berhasil.

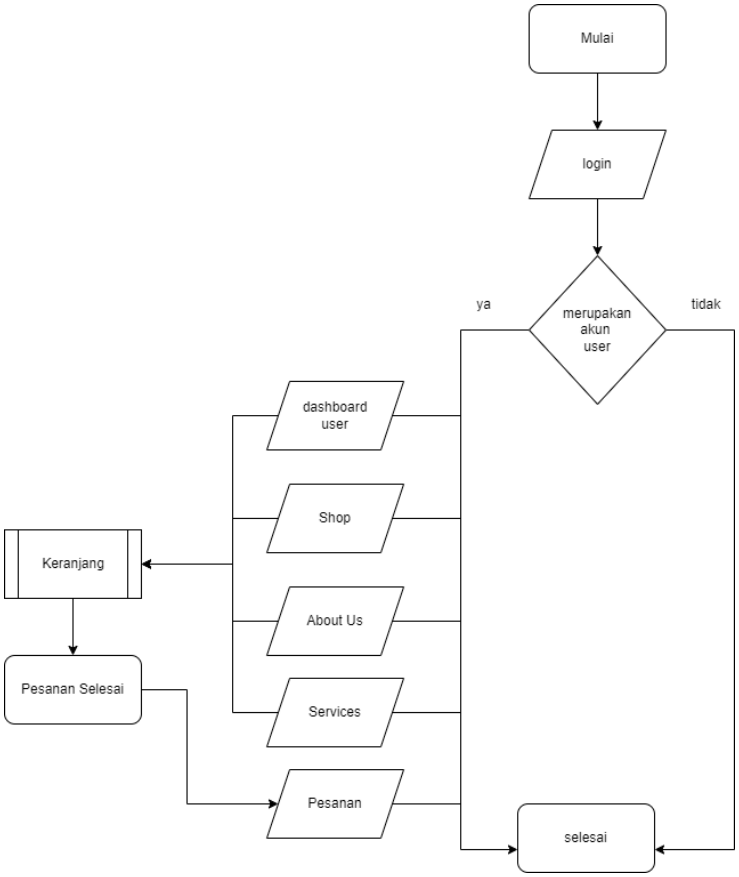


Gambar 3.16 Flowchart Admin

Proses logistik seorang manajer ketika mereka masuk ke sistem manajemen digambarkan dalam *flowchart* di Gambar 3.16. Proses dimulai dengan tombol "Mulai" dan berakhir di tahap "*login*", di mana administrator harus memasukkan kredensialnya untuk dapat mengakses sistem. Setelah proses *login*, sistem mengidentifikasi apakah akun tersebut merupakan akun manajer.

Jika kredensial yang dimasukkan tidak cocok dengan akun admin, proses diarahkan kembali ke langkah awal *login* dan meminta pengguna untuk mencoba lagi. Namun, jika kredensial yang dimasukkan cocok dengan akun admin, proses berlanjut ke "Dashboard admin". Memiliki dua opsi utama untuk admin di *dashboard* ini: mengelola "Transaksi admin" atau "Data Buku". Admin dapat memilih untuk meninjau transaksi yang telah dilakukan, yang mencakup melihat detail transaksi, memverifikasi transaksi, atau melakukan tindakan administratif lain yang berkaitan dengan transaksi. Alternatif lainnya, admin dapat mengakses dan mengelola

"Data Buku", yang bisa mencakup mengedit informasi buku, menambahkan buku baru, atau menghapus entri buku yang sudah ada.

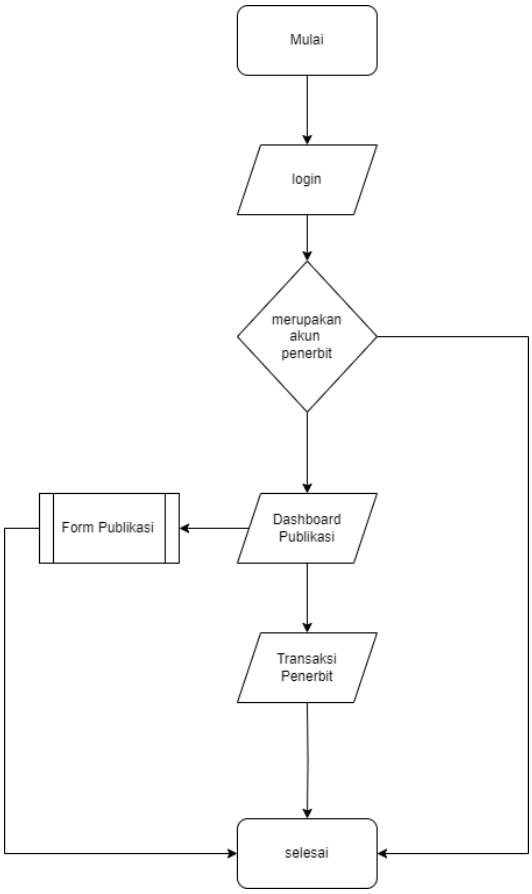


Gambar 3.17 Flowchart User

Proses navigasi pengguna di sebuah situs web setelah *login* berhasil digambarkan dalam *flowchart* Gambar 3.17. Proses ini dimulai dengan pengguna memasukkan kredensial *login* mereka, yang kemudian divalidasi oleh sistem untuk memastikan bahwa kredensial tersebut sesuai dengan data akun pengguna yang didaftarkan. Pengguna akan dibawa ke *dashboard* pengguna, yang berfungsi sebagai pusat kontrol yang memungkinkan mereka mengakses berbagai fitur dan layanan yang disediakan situs web.

Pengguna memiliki beberapa opsi navigasi dari *dashboard* ini, seperti menjelajahi toko (Shop), mendapatkan informasi tentang perusahaan melalui halaman "About Us", melihat layanan yang tersedia di "Services", atau mengelola dan melihat pesanan mereka di "Pesanan". Selain itu, pengguna juga dapat menambahkan produk atau layanan ke dalam "Keranjang" mereka dan melanjutkan untuk membuat pesanan berikutnya. Setelah semua kegiatan belanja atau

pengelolaan pesanan selesai, proses diakhiri. Sebaliknya, jika kredensial yang dimasukkan tidak valid atau tidak sesuai dengan akun pengguna yang terdaftar, sistem akan segera mengakhiri sesi, membatasi akses ke fitur situs web, dan mengarahkan pengguna kembali ke akhir proses, ditandai sebagai 'selesai' dalam *flowchart*.



Gambar 3.18 Flowchart Penerbit

Proses unik yang dilakukan oleh penerbit dalam sistem manajemen publikasi online digambarkan dalam *flowchart* di Gambar 3.18. Proses ini dimulai dengan tahap "Mulai" dan berakhir dengan tahap *login*, di mana penerbit diminta untuk memasukkan kredensial mereka. Setelah masuk, sistem akan memverifikasi akun penerbit. Penerbit diarahkan ke "Dashboard Publikasi" jika kredensial sesuai dengan akun mereka. Di sana, mereka dapat mengakses "Form Publikasi" dan mengisi detail dan mengunggah konten baru ke publikasi. Sebelum proses ini selesai, penerbit mungkin kembali ke form publikasi untuk mengedit atau mengubah informasi. Selanjutnya, penerbit bisa melihat "Transaksi Penerbit", yang

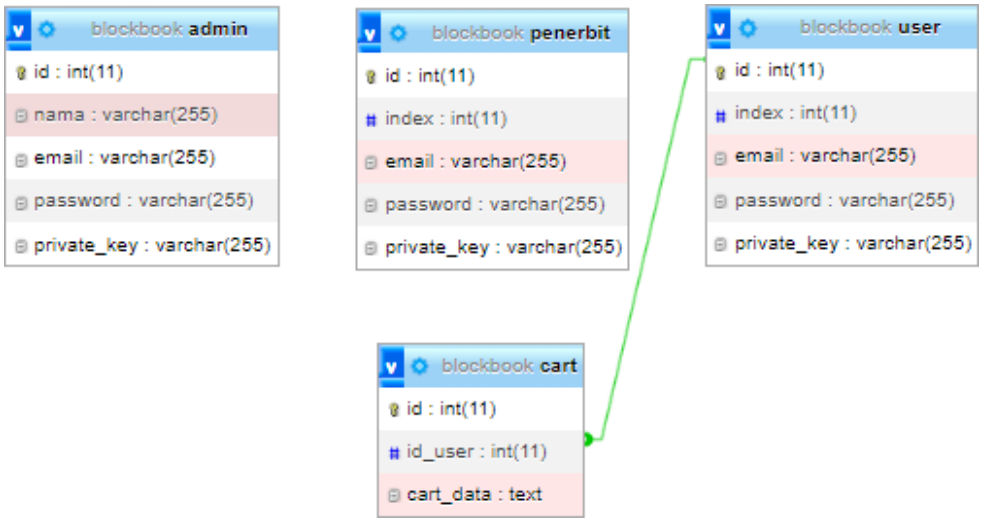
menyediakan informasi tentang penjualan atau distribusi publikasi yang telah dilakukan. Fitur ini memungkinkan penerbit untuk melacak kinerja publikasi mereka secara real-time dan membuat penyesuaian yang diperlukan berdasarkan feedback atau data penjualan.

Proses ini berakhir dengan tahap "selesai", di mana penerbit mungkin memutuskan untuk *logout* atau menutup sesi mereka setelah menyelesaikan aktivitas publikasi dan review transaksi.

3.3.2.5 Desain Database

Design database penelitian ini menggunakan MySQL untuk Web2 server dan *Smart contract* untuk Web3 *blockchain* server.

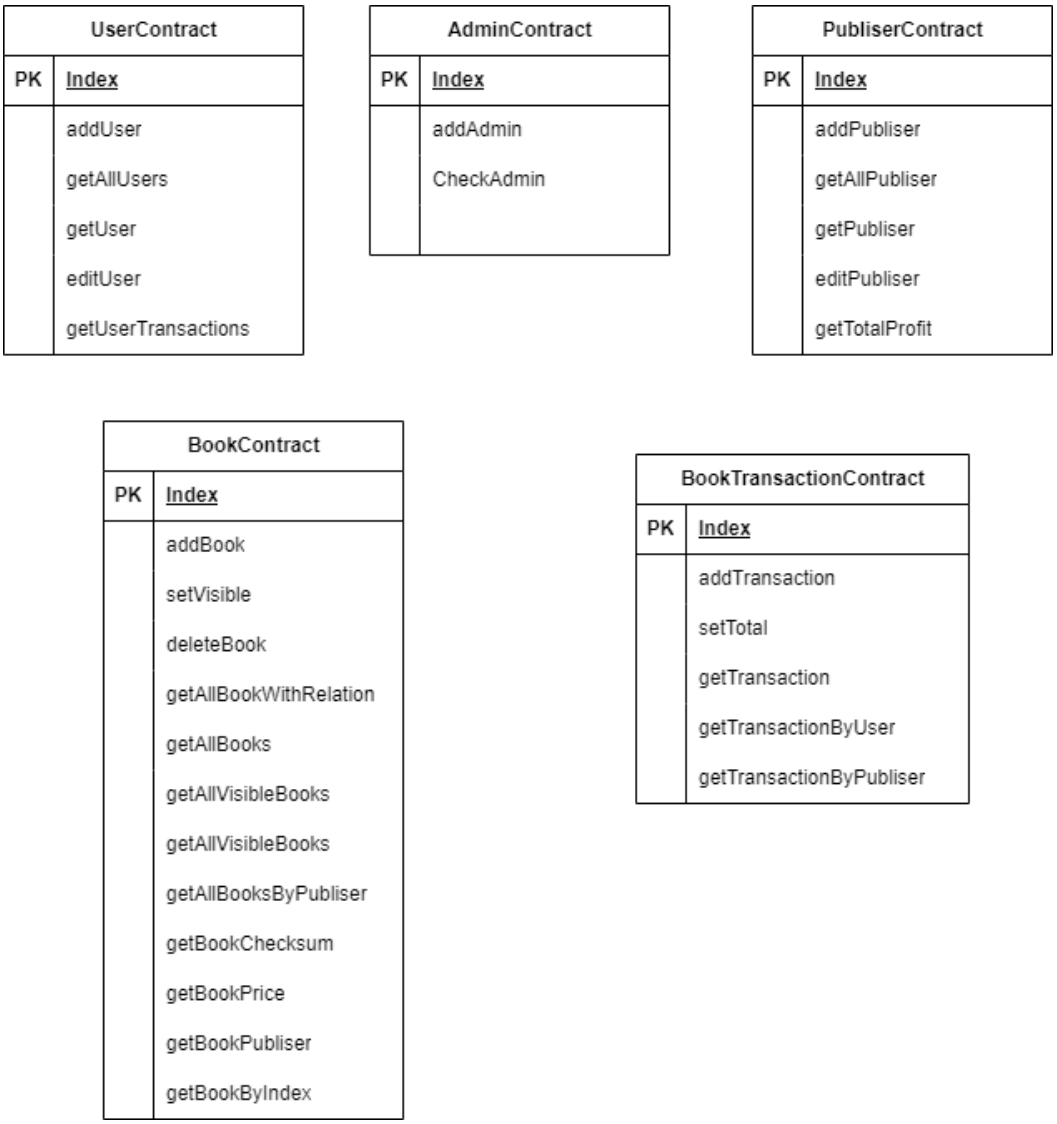
a. MySQL



Gambar 3.19 Design Database Web2

Peneliti telah mengembangkan sebuah rancangan database MySQL yang dinamai "Blockbook" terlihat pada gambar 3.19 sebagai design database Web 2 Server untuk menyokong sebuah sistem manajemen konten yang kompleks. Struktur database ini terdiri dari empat tabel utama yang masing-masing memegang peranan penting dalam sistem. Pertama, tabel blockbook_admin yang berisi informasi tentang administrator sistem, termasuk kolom untuk nama, email, password, dan kunci pribadi untuk keamanan data. Kedua, tabel blockbook_penerbit yang menyimpan data penerbit, mirip dengan tabel admin tetapi dengan penambahan kolom index untuk referensi atau urutan khusus. Ketiga,

tabel blockbook_*user* dikonfigurasikan untuk pengguna, termasuk kunci pribadi untuk menjamin keamanan autentikasi dan transaksi pengguna. Keempat, tabel blockbook_cart yang mencatat data keranjang belanja pengguna, terhubung dengan tabel pengguna melalui kunci asing id_*user*.



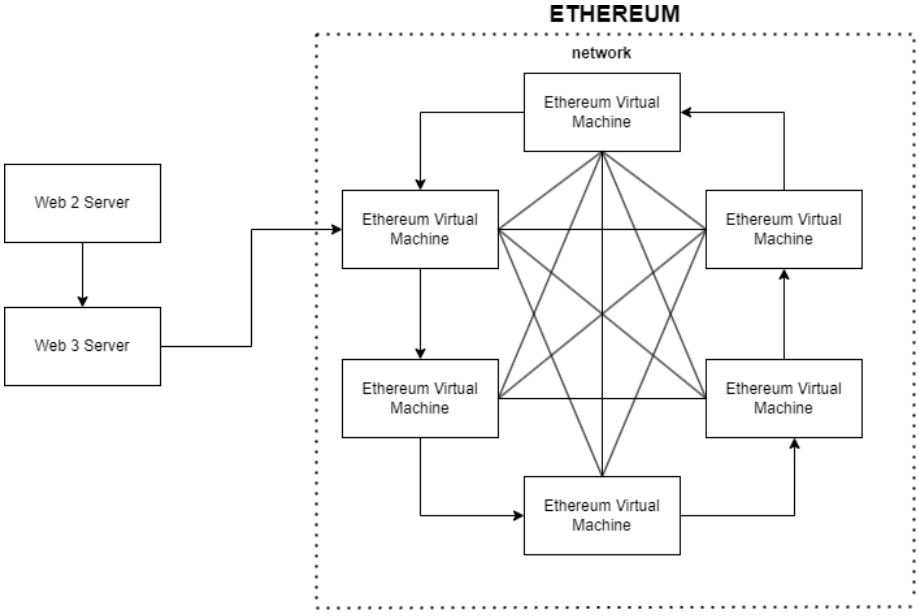
Gambar 3. 20 Design Table of content Smart contract

Gambar yang diberikan menampilkan rancangan dari lima *smart contract* yang berbeda, masing-masing dengan serangkaian fungsi spesifik untuk mengelola aspek tertentu dari sebuah platform digital yang berkaitan dengan buku atau konten serupa. Pertama, *UserContract* bertanggung jawab untuk mengelola pengguna, termasuk penambahan, pengambilan, dan pengeditan informasi pengguna serta

mengakses transaksi pengguna. Kedua, *AdminContract* difokuskan pada administrasi sistem dengan kemampuan untuk menambah admin baru dan memeriksa keberadaan admin dalam database. Ketiga, *PublisherContract* dirancang untuk menangani interaksi dengan penerbit, menyediakan fungsionalitas untuk menambah, mengedit, dan mengambil data penerbit, serta menghitung keuntungan yang diperoleh. Keempat, *BookContract* mengelola data buku, termasuk fungsi untuk menambah, menghapus, dan mengatur visibilitas buku, serta mengambil informasi terperinci buku berdasarkan berbagai kriteria seperti indeks, penerbit, atau status visibilitas. Terakhir, *BookTransactionContract* mengatur transaksi terkait buku, termasuk menambahkan transaksi baru dan mengambil data transaksi berdasarkan pengguna atau penerbit. Setiap kontrak ini memiliki Index sebagai kunci utama, menandakan bahwa indeks digunakan sebagai referensi utama untuk operasi dalam database. Fungsi dalam masing-masing *smart contract* dirancang untuk mendukung efisiensi dan keamanan dalam operasi data dan transaksi yang terkait dengan platform.

3.3.2.6 Pemilihan Platform *Blockchain*

Platform *Blockchain* yang digunakan yaitu Ethereum *Blockchain* tetapi dengan provider *localhost* sebagai node *blockchain* pada penelitian kali ini. Terlihat pada gambar 3.21 berikut,



Gambar 3.21 Tampilan Ethereum Blockchain Localhost

Pada gambar 3.21 menunjukan bahwa *ethereum blockchain* bersifat

desentralisasi, sehingga data yang terkirim pada jaringan *blockchain* tepatnya salah satu node jaringan (*Ethereum Virtual Machine*) akan disebarkan luaskan pada seluruh node pada jaringan *blockchain.*

3.3.3 Tahap *Development* (Pengembangan)

Pada tahap ini dilakukan proses pembuatan *website* yang meliputi pengembangan backend dan integrasi web 2 server dengan web 3 server, pengembangan sistem keamanan data pada server dengan metode encoder base64 shuffle, implementasi *smart contract* untuk manajemen hak cipta termasuk pencatatan kepemilikan dan pencatatan transaksi *e-book*, implementasi fungsionalitas hak cipta dan transaksi *e-book*.

Peneliti menggunakan perangkat laptop dalam pengembangan *website* untuk memastikan bahwa aplikasi yang dikembangkan dapat beroperasi secara efektif di berbagai platform dan perangkat keras. Dengan menggunakan laptop selama fase pengujian, peneliti dapat mengidentifikasi dan mengatasi isu-isu yang berkaitan dengan responsivitas dan fungsionalitas pada *website*. Selain itu, penggunaan server lokal dan server cloud dalam pengembangan memberikan wawasan tentang kinerja *website* di lingkungan yang berbeda, memastikan bahwa aplikasi web teroptimasi baik untuk kecepatan dan keamanan. Kombinasi penggunaan perangkat ini tidak hanya meningkatkan kualitas akhir dari *website* tetapi juga memastikan bahwa pengalaman pengguna yang konsisten dan efisien dijamin di semua platform.

Perangkat yang digunakan oleh peneliti, meliputi :

Laptop : Asus X454YA

Processor : AMD A8 with Radeon R5 Graphic

RAM : 8GB

VRAM : 2GB

Selain itu terdapat aplikasi dan Alat yang digunakan, meliputi :

1. Visual Studio code versi 1.19.1;
2. *Node js* versi 20.9.0;
3. NPM versi 10.8.0;
4. Windows Subsytem Linux (WSL) Ubuntu versi 20.04.6 LTS;
5. Hardhat Solidity;

6. MySQL;
7. Firebase Storage;

Dengan menggunakan perangkat, aplikasi, dan alat yang tersedia, diharapkan peneliti dapat meningkatkan proses pengembangan dari desain aplikasi yang telah dijelaskan sebelumnya. Penggunaan alat-alat ini memungkinkan peneliti untuk menyempurnakan aplikasi sesuai dengan kebutuhan pengguna. Pendekatan ini membantu dalam mengidentifikasi dan mengatasi masalah lebih dini dalam proses pengembangan, sehingga memastikan kualitas dan keandalan aplikasi.

3.3.3.1 Pemilihan Keamanan Perlindungan Database Server

Pada penelitian kali ini perlindungan database server menggunakan penyimpanan Firebase Storage, dan diterapkan metode encoder yang melibatkan penggunaan Base64 Shuffle. Metode ini mengambil data asli dan mengubahnya menjadi format Base64, yang kemudian diacak susunannya melalui algoritma shuffle yang telah ditentukan.

Gambar 3.22 Alur Kerja Base64 shuffle

Proses ini tidak hanya menyamarkan konten data dari pandangan langsung tetapi juga menambah lapisan perlindungan terhadap upaya pemecahan kode secara langsung, terlihat pada gambar 3.22. Encoder seperti ini sangat berguna dalam melindungi informasi sensitif seperti dokumen *e-book* dan identitas penerbit yang disimpan dalam Firebase Storage, memastikan bahwa hanya pihak yang memiliki kunci dekripsi yang dapat mengakses dan memahami isi data secara benar.

Peneliti juga melakukan pengujian secara langsung melalui situs web bernama base64.guru, yang dapat melakukan dekode terhadap *file* PDF yang telah dikodekan menggunakan base64. Pengujian ini dilakukan semata-mata untuk menunjukkan bahwa pengkodean base64 dengan metode acak (shuffle) dapat menghalangi penguraian oleh dekoder base64 pada situs web tersebut atau dekoder lainnya.

3.3.3.2 Integrasi *Blockchain* dengan Web Interface

Integrasi antara server Web 2 yang berbasis Express.js dan teknologi *blockchain* Web 3 dapat dilakukan dengan efektif menggunakan API Ether.js. Ether.js merupakan pustaka yang ringan dan fleksibel, dirancang khusus untuk berinteraksi dengan Ethereum *blockchain*. Melalui integrasi ini, aplikasi Web 2 dapat memanfaatkan *smart contract* dan fitur keamanan *blockchain* lainnya secara langsung. Untuk memfasilitasi integrasi ini, telah dibuat sebuah API List yang mendokumentasikan setiap endpoint yang tersedia untuk interaksi antara interface Web dan komponen *blockchain*. Dengan mengimplementasikan Ether.js dan memanfaatkan API List yang rinci, penelitian ini memungkinkan pengiriman, penerimaan, dan pengelolaan transaksi *blockchain* serta interaksi yang lebih efisien dengan *smart contract* melalui kode JavaScript yang sederhana. Hal ini memungkinkan peneliti untuk mengembangkan aplikasi tradisional yang berjalan pada server Express.js agar dapat mengintegrasikan komponen desentralisasi, memperluas fungsionalitas mereka, dan meningkatkan keamanan serta transparansi operasional. Dibawah ini merupakan tabel List API untuk semua API yang digunakan pada *website* untuk integrasi.

Tabel 3.1 List API

No	Proses	Method	endpoint
1	Membuat akun baru *user*	POST	/registrasi*user*
2	Membuat akun baru penerbit	POST	/registrasipenerbit
3	Melakukan *login*	POST	/*login*
4	Melakukan *logout*	GET	/*logout*
5	Melakukan publikasi buku	POST	penerbit/publikasi
6	Melihat data buku	GET	admin/data buku
7	Melihat halaman shop	GET	*user*/shop
8	Mengambil data cart pada sesi	GET	/cart
9	Menyimpan data cart pada sesi	POST	/cart
10	Melakukan checkout	POST	/checkout
11	Melihat halaman about	GET	*user*/about
12	Melihat halaman contact	GET	*user*/contact
13	Melihat halaman publikasi buku	GET	penerbit/publikasi
14	Melihat halaman dashboard admin	GET	admin/dashboard admin
15	Melihat halaman dashboard tamu	GET	/dashboard
16	Melihat halaman dashboard *user*	GET	*user*/dashboard_*user*
17	Melihat halaman *login*	GET	/*login*
18	Melhat halaman registrasi *user*	GET	/regist*euser*
19	Melihat halaman registrasi penerbit	GET	/*Register*penerbit
20	Melihat halaman profil penerbit	GET	penerbit/profil
21	Melakukan download *file* ebook	GET	/download *file* ebook
22	Melakukan download *file* identitas	GET	/download *file* identitas
23	Mebuat buku terhapus	POST	/delete_buku
24	Membuat buku terlihat	POST	/updatevisible_buku
25	Melihat halaman keranjang	GET	*user*/keranjang

26	Melihat halaman transaksi admin	GET	admin/transaksi admin
27	Melihat halaman transaksi penerbit	GET	penerbit/transaksi penerbit
28	Melakukan pemayaran	POST	/payment
29	Melihat halaman transaksi *user*	GET	*user*/myorder

3.3.4 Tahap Implementation (Implementasi)

Tahap ini merupakan tahap uji coba sistem yang telah dikembangkan. Pengujian dilakukan menggunakan *Blackbox Testing*. Pengujian meliputi kesesuain input dan output pada *website* BlockBook, fungsionalitas *smart contract* untuk integrasi dan pengiriman data serta fungsionalitas *blockchain* sebagai penyimpan data buku dan data transaksi. Pengujian Blackbox Testing dilakukan dengan menerapkan skenario yang telah dibuat pada tabel berikut.

Tabel 3.2 Skenario Pengujian Fungsionalitas Halaman Log-In

Kode Pengujian	Skenario Pengujian	Hasil yang Diharapkan
LGN-001	*Login* dengan akun	*Login* berhasil
LGN-002	*Login* dengan akun tidak terdaftar	*Login* tidak berhasil
LGN-002	Terdapat kolom kosong	Validasi input untuk memasukan email dan password
LGN-002	*Login* email tanpa kredensial yang benar	Validasi input untuk memasukan @ sebagai kredensial yang benar

Tabel 3.3 Skenario Pengujian Fungsionalitas Halaman Register

Kode Pengujian	Skenario Pengujian	Hasil yang Diharapkan
RGT-001	Terdapat kolom kosong	Validasi input untuk memasukan input pada kolom kosong

RGT-002	*Login* email tanpa kredensial yang benar	Validasi input untuk memasukan @ sebagai kredensial yang benar
RGT-003	Tidak centang checkbox	Tidak dapat melakukan registrasi
RGT-004	Registrasi terisi sesuai	Registrasi berhasil

Tabel 3.4 Skenario Pengujian Fungsionalitas Halaman Log-Out

Kode Pengujian	**Skenario Pengujian**	**Hasil yang Diharapkan**
LGT-001	Melakukan klik pada *logout*	*Logout* berhasil

Tabel 3.5 Skenario Pengujian Fungsionalitas Halaman Dashboard Tamu

Kode Pengujian	**Skenario Pengujian**	**Hasil yang Diharapkan**
DBT-001	Masuk halaman dengan session akun	Tampilan *login*

Tabel 3.6 Skenario Pengujian Fungsionalitas Halaman Publikasi

Kode Pengujian	**Skenario Pengujian**	**Hasil yang Diharapkan**
PBL-001	Terdapat kolom kosong	Validasi input untuk memasukan input pada kolom kosong
PBL-002	Publikasi buku benar	Publikasi berhasil
PBL-003	Publikasi buku salah	Publikasi tidak berhasil

Tabel 3.7 Skenario Pengujian Fungsionalitas Halaman Data Buku

Kode Pengujian	**Skenario Pengujian**	**Hasil yang Diharapkan**
HDB-001	Upload rincian buku pada form publikasi	Rincian buku tampil
HDB-002	Konfigurasi izin buku tampil *true*	Buku tampil pada shop

HDB-003	Konfigurasi izin buku tampil *false*	Buku tidak tampil pada shop
HDB-004	Menghapus salah satu buku	Buku tidak tampil dan terhapus
HDB-005	Download *file* Identitas dan Buku	*File* terdownload

Tabel 3.8 Skenario Pengujian Fungsionalitas Halaman Keranjang

Kode Pengujian	Skenario Pengujian	Hasil yang Diharapkan
KJG-001	Memasukan buku kedalam keranjang	Buku tampil pada keranjang
KJG-002	Melakukan checkout pada keranjang	Data tampil pada halaman pembayaran
KJG-003	Hapus buku pada keranjang	Buku hilang dalam keranjang

Tabel 3.9 Skenario Pengujian Fungsionalitas Halaman Pembayaran

Kode Pengujian	Skenario Pengujian	Hasil yang Diharapkan
PBR-001	Pembayaran semua buku	Pembayaran terkonfirmasi

Tabel 3.10 Skenario Pengujian Fungsionalitas Blockchain

Kode Pengujian	Skenario Pengujian	Hasil yang Diharapkan
FBC-001	Publikasi buku	Berhasil tersimpan pada *blockchain*
FBC-002	Transaksi pembelian buku	Transaksi tersimpan pada *blockchain*

Selain pengujian Blackbox Testing untuk input validation dan keseuaian validasi data, peneliti melakukan pengujian pada fungsionalitas *smart contract* menggunakan Test Driven Development (TDD). TDD merupakan metodologi

pengembangan yang efektif untuk memastikan keandalan dan keamanan *smart contract* yang akan diterapkan pada platform Web3. Dalam konteks pengembangan *smart contract* menggunakan Solidity, TDD melibatkan penulisan rangkaian tes sebelum pengembangan *smart contract* itu sendiri untuk mendefinisikan perilaku yang diharapkan dari kontrak tersebut. Melalui penggunaan Hardhat, sebuah lingkungan pengembangan Ethereum yang populer, pengembang dapat menulis dan menjalankan tes secara lokal dengan kemudahan yang tinggi. Hardhat menyediakan fitur-fitur yang mendukung simulasi eksekusi *smart contract*, memungkinkan pengembang untuk menemukan dan memperbaiki kesalahan lebih dini dalam siklus pengembangan. Proses ini tidak hanya meningkatkan kualitas kode tetapi juga menjamin keamanan dan optimalisasi kontrak sebelum di-deploy ke jaringan Ethereum.

Pengujian meliputi pengujian fungsionalitas *smart contract*, diantaranya AdminContract.sol, PubliserContract.sol, *User*Contract.sol, BookContract.sol dan BookTransactionContract.sol.

1. Pengujian AdminContract.sol

Pengujian kali ini melibatkan *test code* yang dibuat pada *node js*, *test code* meliputi AdminContract.testfail.js untuk pengujian pertama dan diharuskan pengujian dengan hasil salah itu berhasil di test. Sedangkan *test code* AdminContract.testpass.js untuk pengujian setelah *contract* selesai dibuat, sehingga contract dapat diketahui terdapatnya bug atau hal lainnya. *Code* AdminContract.testfail.js dan *code* AdminContract.testpass.js terdapat pada Lampiran 1.

Tabel 3.11 Pengujian AdminContract.sol

Nama Pengujian	Skenario Pengujian	Hasil yang Diharapkan
Admin fail test	memvalidasi mekanisme keamanan dan manajemen admin dalam kontrak	Test berhasil dilakukan
Admin pass test	memvalidasi bahwa proses menambahkan dan memverifikasi	Test berhasil dilakukan

	admin dalam AdminContract bekerja sesuai harapan	

Pengujian yang dilakukan merupakan pengujian untuk memvalidasi mekanisme keamanan dan manajemen admin dalam kontrak, dengan memastikan bahwa hanya admin dengan kredensial yang valid yang dapat diverifikasi, sementara akses yang tidak sah (dengan password salah) harus ditolak.

Sedangkan pengujian Admin pass test bertujuan untuk memvalidasi bahwa proses menambahkan dan memverifikasi admin dalam `AdminContract` bekerja sesuai harapan dan bahwa sistem dapat secara akurat mengidentifikasi admin yang sah berdasarkan kredensial yang diberikan.

2. Pengujian PubliserContract.sol

Pengujian kali ini melibatkan *test code* yang dibuat pada *node js*, *test code* meliputi PubliserContract.testfail.js untuk pengujian pertama dan diharuskan pengujian dengan hasil salah itu berhasil di test. Sedangkan *test code* PubliserContract.testpass.js untuk pengujian setelah contract selesai dibuat, sehingga contract dapat diketahui terdapatnya bug atau hal lainnya. Code PubliserContract.testfail.js dan code PubliserContract.testpass.js terdapat pada Lampiran 2.

Tabel 3.12 Pengujian PubliserContract.sol

Nama Pengujian	Skenario Pengujian	Hasil yang Diharapkan
Publiser fail test	memverifikasi bahwa sistem memiliki tindakan perlindungan yang tepat terhadap permintaan yang tidak valid atau akses ke data yang tidak ada.	Test berhasil dilakukan
Publiser pass test	Menambahkan publisher dengan data seperti nama, email, identitas,	Test berhasil dilakukan

	bankId, dan accessKey, kemudian mengambil publisher tersebut dari kontrak menggunakan index yang diharapkan, yaitu index 0.	
	Memeriksa jumlah awal publiser, yang seharusnya 0, menambahkan sebuah publisher dan memeriksa jumlah publiser .	Test berhasil dilakukan

Pengujian yang dilakukan untuk memverifikasi bahwa sistem memiliki tindakan perlindungan yang tepat terhadap permintaan yang tidak valid atau akses ke data yang tidak ada, dengan mengembalikan kesalahan yang tepat. Ini penting untuk memastikan bahwa kontrak beroperasi dengan aman dan sesuai dengan kebutuhan bisnis yang diharapkan, menghindari kegagalan yang tidak terduga atau akses tidak sah ke data yang sensitif.

Sedangkan pengujian pass test untuk memverifikasi bahwa fungsi dasar `PubliserContract`, seperti menambahkan dan mengambil publiser, bekerja dengan benar. Ini termasuk memastikan bahwa data yang disimpan dapat diakses kembali dengan akurat dan bahwa sistem memelihara jumlah entri dengan benar, memberikan validasi kunci untuk operasi yang andal dan teratur dalam penggunaan sehari-hari.

3. Pengujian *User*Contract.sol

Pengujian kali ini melibatkan *test code* yang dibuat pada *node js*, *test code* meliputi *User*Contract.testfail.js untuk pengujian pertama dan diharuskan pengujian dengan hasil salah itu berhasil di test. Sedangkan *test code* *User*Contract.testpass.js untuk pengujian setelah contract selesai dibuat, sehingga contract dapat diketahui terdapatnya bug atau hal lainnya. Code *User*Contract.testfail.js dan code *User*Contract.testpass.js terdapat pada Lampiran 3.

Tabel 3.13 Pengujian UserContract.sol

Nama Pengujian	Skenario Pengujian	Hasil yang Diharapkan
User fail test	Mencoba mengambil pengguna menggunakan indeks yang dianggap tidak ada.	Test berhasil dilakukan
	Mencoba mengedit pengguna dengan indeks yang tidak ada dengan menyediakan data baru (nama, email, dan access key).	Test berhasil dilakukan
User pass test	Menambahkan pengguna baru dengan data nama, email, dan access key yang spesifik dan engambil data pengguna tersebut untuk memverifikasi keakuratan informasi yang disimpan.	Test berhasil dilakukan
	Mengedit pengguna yang sudah ada dengan menyediakan data baru (nama baru, email baru, dan access key baru) dan mengambil data pengguna yang telah diedit untuk memverifikasi bahwa perubahan telah diterapkan dengan benar.	Test berhasil dilakukan

Pengujian yang dilakukan untuk untuk memverifikasi bahwa kontrak *User*Contract dapat secara konsisten mengenali dan menangani situasi ketika pengguna yang tidak terdaftar dicoba untuk diakses atau dimodifikasi. Hal ini penting untuk memastikan integritas data dan operasi dalam kontrak, serta untuk mencegah aksi-aksi yang tidak sah atau kesalahan dalam manajemen pengguna. Pengujian ini juga membantu memastikan bahwa sistem dapat memberikan pesan error yang jelas dan tepat untuk situasi-situasi error yang didefinisikan, sehingga memudahkan debugging dan pemeliharaan kode.

Sedangkan pengujian pass test untuk memverifikasi bahwa fungsi dasar *User*Contract, seperti menambahkan, mengedit, dan mengambil data pengguna, beroperasi dengan efektif dan akurat. Pengujian ini memastikan bahwa sistem dapat menangani penambahan dan modifikasi pengguna dengan benar, serta menyimpan dan mengembalikan data dengan tepat, yang merupakan fungsi penting untuk manajemen pengguna yang efisien dalam aplikasi.

4. Pengujian BookContract.sol

Pengujian kali ini melibatkan *test code* yang dibuat pada *node js*, *test code* meliputi BookContract.testfail.js untuk pengujian pertama dan diharuskan pengujian dengan hasil salah itu berhasil di test. Sedangkan *test code* BookContract.testpass.js untuk pengujian setelah contract selesai dibuat, sehingga contract dapat diketahui terdapatnya bug atau hal lainnya. Code BookContract.testfail.js dan code BookContract.testpass.js terdapat pada Lampiran 4.

Tabel 3.14 Pengujian BookContract.sol

Nama Pengujian	Skenario Pengujian	Hasil yang Diharapkan
Book fail test	Mencoba mengambil buku menggunakan indeks yang dianggap tidak ada.	Test berhasil dilakukan
	Mencoba menghapus buku yang ada menggunakan password yang salah.	Test berhasil dilakukan

Book pass test	Menambahkan buku baru dan kemudian mengambil buku tersebut untuk memverifikasi keakuratan data.	Test berhasil dilakukan
	Mengatur visibilitas buku menggunakan password yang valid dan mengambil buku tersebut untuk memverifikasi status visibilitasnya.	Test tidak berhasil dilakukan
	Menghapus buku menggunakan password yang valid dan mengambil buku tersebut untuk memverifikasi status penghapusannya.	Test tidak berhasil dilakukan

Pengujian yang dilakukan untuk memverifikasi bahwa kontrak BookContract dapat secara konsisten mengenali dan menangani situasi ketika buku yang dicoba untuk diakses atau dihapus tidak memenuhi kondisi yang dibutuhkan. Hal ini penting untuk memastikan integritas data dan operasi dalam kontrak, serta untuk mencegah aksi-aksi yang tidak sah atau kesalahan dalam manajemen buku. Pengujian ini juga membantu memastikan bahwa sistem dapat memberikan feedback error yang jelas dan tepat untuk situasi-situasi error yang didefinisikan, sehingga memudahkan debugging dan pemeliharaan kode.

Sedangkan pengujian pass test untuk memverifikasi bahwa kontrak BookContract berfungsi efektif dalam mengelola data buku, termasuk menambahkan, mengatur visibilitas, dan menghapus entri buku. Fungsi-fungsi ini

penting untuk manajemen buku yang efisien dan dinamis dalam sistem, serta menjamin bahwa operasi-operasi tersebut dapat dilakukan dengan benar, memastikan integritas data dan keandalan aplikasi.

5. BookTransactionContract.sol

Pengujian kali ini melibatkan *test code* yang dibuat pada *node js*, *test code* meliputi BookTransactionContract.testfail.js untuk pengujian pertama dan diharuskan pengujian dengan hasil salah itu berhasil di test sedangkan *test code* BookTransactionContract.testpass.js untuk pengujian setelah contract selesai dibuat, sehingga contract dapat diketahui terdapatnya bug atau hal lainnya. Code BookTransactionContract.testfail.js dan code BookTransactionContract.testpass.js terdapat pada Lampiran 5.

Tabel 3. 15 Pengujian BookTransactionContract.sol

Nama Pengujian	Skenario Pengujian	Hasil yang Diharapkan
Book Transaction fail test	Mencoba mengambil transaksi menggunakan ID yang dianggap tidak ada.	Test berhasil dilakukan
	Mencoba mengatur total untuk transaksi yang tidak ada dengan menyediakan ID yang tidak valid dan total baru.	Test berhasil dilakukan
Book Transaction pass test	Mengambil transaksi yang telah ditambahkan untuk memverifikasi keakuratan data.	Test berhasil dilakukan
	Mengatur ulang total transaksi menggunakan nilai total baru dan mengambil transaksi tersebut untuk	Test berhasil dilakukan

memverifikasi bahwa totalnya telah diupdate.

Pengujian yang dilakukan untuk memverifikasi bahwa kontrak BookTransactionContract.sol dapat secara konsisten mengenali dan menangani situasi ketika transaksi yang dicoba untuk diakses atau dimodifikasi tidak memenuhi kondisi yang dibutuhkan. Hal ini penting untuk memastikan integritas data dan operasi dalam kontrak, serta untuk mencegah aksi-aksi yang tidak sah atau kesalahan dalam manajemen transaksi buku. Pengujian ini juga membantu memastikan bahwa sistem dapat memberikan *feedback error* yang jelas dan tepat untuk situasi-situasi *error* yang didefinisikan, sehingga memudahkan *debugging* dan pemeliharaan kode.

Sedangkan pengujian *pass test* untuk memverifikasi bahwa kontrak BookTransactionContract berfungsi efektif dalam mengelola data transaksi, termasuk menambahkan transaksi dan mengatur ulang total. Fungsi-fungsi ini penting untuk manajemen transaksi yang efisien dan dinamis dalam sistem, serta menjamin bahwa operasi-operasi tersebut dapat dilakukan dengan benar, memastikan integritas data dan keandalan aplikasi.

Perancangan pengujian *smart contract* sangat penting dalam pengembangan dan implementasi teknologi *blockchain* untuk menjamin keandalan dan keamanan sistem. Hasil pengujian yang menyeluruh diharapkan akan menunjukkan apakah kontrak pintar yang dirancang dapat berfungsi sesuai dengan spesifikasi yang telah ditetapkan sehingga dapat diandalkan dalam operasi sehari-hari. Hasil pengujian ini sangat penting karena akan menentukan apakah kontrak pintar tersebut siap untuk diproduksi atau tidak. Tujuan dari keseluruhan proses adalah untuk menghilangkan kerentanan dan memastikan bahwa semua operasi dan transaksi yang dilakukan melalui *smart contract* dilakukan dengan akurat dan efisien. Ini akan meningkatkan kepercayaan pengguna terhadap sistem *blockchain* yang digunakan.

3.3.5 Tahap *Evaluate* (Evaluasi)

Pada tahap ini dilakukan evaluasi berdasarkan hasil uji coba yang dilakukan. Evaluasi ini meliputi evaluasi sistem kerja, evaluasi keamanan dan evaluasi fungsionalitas.

3.4 Operasional Variabel

Variabel operasional merupakan ukuran yang telah ditetapkan oleh peneliti dengan tujuan mempelajari informasi yang diperoleh dan pada akhirnya mengambil kesimpulan. Ada dua jenis variabel operasional, yaitu variabel independen yang dapat mempengaruhi variabel dependen. Variabel - variabel operasional dalam penelitian ini terdapat pada tabel berikut.

Tabel 3.16 Operasional Variabel

No	Variabel	Definisi	Indikator	Cara Ukur	Alat Ukur
1.	Keberhasilan implementasi *blockchain*	keberhasilan penerapan teknologi *blockchain* pada sistem manajemen hak cipta dan transaksi *e-book*	- Koneksi Web2 dan Web3 - Integrasi data Web2 dan Web3	Evaluasi fungsi dan integrasi antara platform Web2 dan Web3 yang digunakan dalam sistem.	Log Server
2.	Transparansi transaksi *e-book*	Transaksi tidak melibatkan pihak ketiga	- transaksi transparan - transaksi tidak dapat diubah	Keterbukaan dan kemudahan akses informasi transaksi untuk semua pihak yang terlibat.	Audit Log
3	Keamanan data dalam transaksi *e-book*	Tingkat perlindungan terhadap akses dan modifikasi data tidak sah	- Data tersimpan dalam *blockchain* - Desentralisasi	Tingkat keamanan data melawan akses tidak sah dan modifikasi	*Blockchain* Integrity Checks

			blockchain	data.	
4	Efektivitas manajemen hak cipta *e-book*	Mencocokan data pada *blockchain* dengan *e-book* aseli	- Penggunaan checksum sha256 - Kekuatan enkripsi	Akurasi dan efisiensi dalam mencocokkan dan mengelola hak cipta menggunakan *blockchain*.	Checksum Validation

3.5 Teknik Pengumpulan Data

a. Pengumpulan data transaksi *website* 2

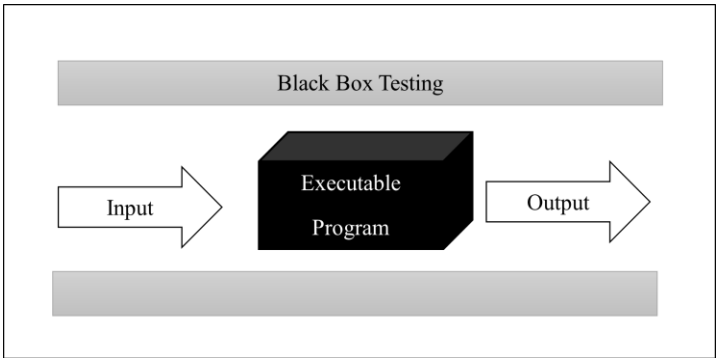
Pada *website* 2 transaksi umumnya dicatat dalam database server yang dikelola oleh *website*. API Logging: Gunakan Application Programming, Interface (API) untuk log transaksi. API bisa digunakan untuk mengekstrak data transaksi dari database backend, Data Integration Tools: Gunakan alat integrasi data untuk menarik data dari berbagai sumber.

b. Pengumpulan data transaksi *blockchain*

Pada *blockchain* semua data dikonfigurasi dan disimpan dengan bentuk dan konfigurasi yang berbeda, penyimpanan dilakukan dengan melakukan konfigurasi *smart contract* untuk berbagai function, seperti getAllBooks, getAllTransaction, getAllnode seperti *user*, admin dan penerbit.

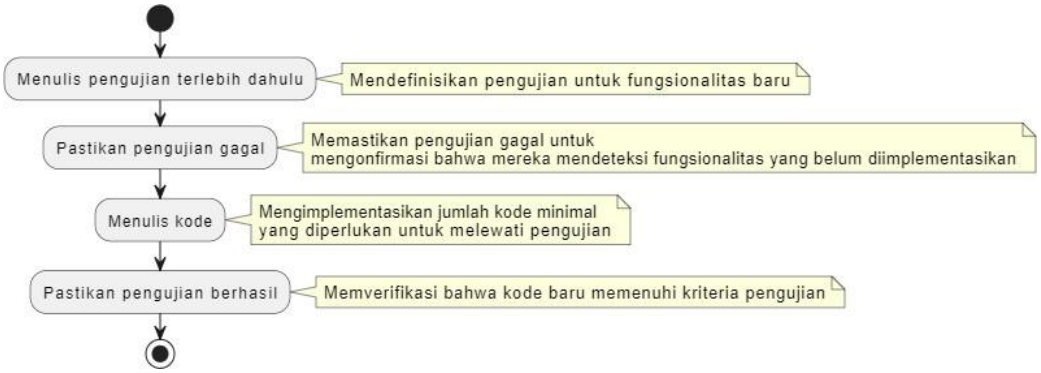
3.6 Pengujian Menggunakan Blackbox Testing dan Test Driven Development

Black Box testing merupakan sebuah cara pengujian yang berfokus kepada fungsionalitas suatu *software*. Uji coba *black box* ini berusaha menemukan kesalahan seperti fungsi yang salah, kesalahan *interface*, kesalahan dalam struktur data, kesalahan performa, dan kesalahan inisialisasi dan terminasi (Sabur & Atmia, 2019).



Gambar 3.23 Gambar Black Box Testing

Test Driven Development (TDD) adalah proses pengembangan perangkat lunak yang bergantung pada pengulangan siklus pengembangan yang sangat singkat. fungsionalitas dari perangkat lunak itu sendiri (Jamalludin et al, 2018). Pendekatan ini mendukung pengembangan perangkat lunak yang lebih bersih dan bug yang lebih sedikit. Metode TDD dalam pengujian ini digunakan untuk menguji fungsionalitas *smart contract* yang telah dibuat dengan pengujian unit testing. Prosesnya terbagi menjadi beberapa langkah utama, yang dapat dijelaskan melalui diagram alur kerja berikut:



Gambar 3.24 Test Driven Development

1. Write tests first

   Langkah pertama dalam TDD adalah menulis tes untuk fungsi atau fitur baru yang belum diimplementasikan. Tes ini dirancang untuk gagal pada awalnya karena kode yang diuji belum ada. Tes harus spesifik dan jelas menggambarkan apa yang diharapkan dari kode yang akan dibuat. Tes ini berfungsi sebagai definisi kebutuhan atau spesifikasi fungsional dalam bentuk kode.

2. Make sure tests fail

   Setelah tes ditulis, langkah selanjutnya adalah menjalankan tes tersebut

untuk memastikan bahwa mereka gagal. Kegagalan ini penting karena memverifikasi bahwa tes benar-benar mencerminkan ketiadaan fungsionalitas yang diharapkan dan valid dalam menangkap kekurangan dalam kode. Ini menegaskan bahwa tes telah ditetapkan dengan benar dan siap untuk digunakan sebagai dasar dalam pengembangan kode.

3. Write code

Dengan tes yang telah ditetapkan dan dikonfirmasi gagal, langkah berikutnya adalah menulis kode yang dibutuhkan untuk membuat tes tersebut lulus. Kode ini harus seminimal mungkin dan hanya cukup untuk memenuhi tes. Ini mendorong penulisan kode yang efisien dan menghindari penambahan fitur yang tidak perlu, yang sering menjadi sumber bug.

4. Make sure tests pass

Setelah kode ditulis, tes dijalankan kembali untuk memastikan bahwa mereka sekarang berhasil. Suksesnya tes ini menunjukkan bahwa kode baru memenuhi spesifikasi yang dinyatakan oleh tes dan berfungsi seperti yang diharapkan. Ini juga membantu memvalidasi keakuratan implementasi kode.