BAB III
PELAKSANAAN KERJA PROFESI

3.1 Bidang Kerja

Selama menjalani Kerja Profesi di PT Alpha Solusi Teknologi, praktikan ditempatkan di divisi Programmer dengan tugas utama pada pengembangan dan integrasi sistem menggunakan Odoo. Divisi Programmer ini memiliki tanggung jawab utama dalam mengembangkan solusi berbasis Odoo yang disesuaikan dengan kebutuhan klien dan memperkenalkan inovasi yang menjadikan produk lebih kompetitif di pasar teknologi. Selain itu, divisi ini memainkan peran krusial dalam mendorong terciptanya produk yang inovatif dengan memanfaatkan teknologi terbaru sebagai upaya menghadapi persaingan yang akan datang.

Sebagai anggota dari divisi Programmer, praktikan ikut serta dalam proyek integrasi Sistem Informasi Sekolah berbasis Odoo. Proyek ini bertujuan untuk menyediakan akses data yang relevan bagi siswa dan orang tua melalui aplikasi mobile, meliputi informasi akademik, kehadiran, tagihan biaya, hingga aktifitas siswa lainnya. Fitur utama sistem ini memungkinkan siswa memperoleh informasi penting secara terstruktur dan real-time.

Dalam proses pengembangan Sistem Informasi Sekolah ini, praktikan bertugas mengintegrasikan data dari Odoo agar dapat ditampilkan pada aplikasi mobile, serta bekerja sama dengan tim backend dan frontend untuk memastikan bahwa data tersaji secara optimal. Selain itu, praktikan juga diinstruksikan untuk membuat diagram *Unified Modeling Language* (UML), termasuk *use case diagram* dan *activity diagram*, guna memodelkan sistem secara terstruktur dan memberikan gambaran alur kerja yang lebih jelas bagi tim yang terlibat.

3.2 Pelaksanaan Kerja

Praktikan, bersama pembimbing eksternal, menetapkan serangkaian langkah kerja yang sistematis dan terorganisir, termasuk menyusun linimasa proyek yang terstruktur. Langkah ini bertujuan agar pengembangan sistem berjalan sesuai dengan target yang ditetapkan. Berikut adalah tabel linimasa kerja yang disusun oleh praktikan untuk memudahkan pembaca dalam memahami langkah-langkah kerja.

Tabel 3.1 Linimasa Aktifitas Kerja Profesi

Aktivitas Kerja Profesi	Jadwal Kegiatan															
	Juni				Juli				Agustus				September			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Analisis Kebutuhan Fitur																
Pembutan Mockup & UI																
Perancangan Backend																
Perancangan Frontend																
Implementasi Integrasi Sistem																
Pengujian Sistem																
Evaluasi Kinerja Sistem																

Tabel di atas memberikan gambaran tentang tahapan pengembangan sistem selama periode 3 bulan dan akan berlanjut hingga sistem siap dirilis. Kemungkinan besar, sistem yang dikembangkan akan mengalami beberapa penyesuaian guna mendukung proses pengambilan keputusan oleh pihak terkait. Sepanjang linimasa ini, praktikan secara rutin berkoordinasi dengan pembimbing eksternal untuk memastikan sistem yang dihasilkan sesuai dengan kebutuhan pengguna.

Dalam pengembangan sistem ini, praktikan menerapkan beberapa tahap pengerjaan untuk memastikan sistem dapat berfungsi dengan baik tanpa kendala. Tahap-tahap tersebut meliputi fase analisis, desain, implementasi, integrasi, dan pengujian.

3.2.1 Deskripsi Sistem

Praktikan mengembangkan sebuah sistem informasi sekolah yang terintegrasi antara platform Odoo dan aplikasi mobile untuk memudahkan distribusi informasi penting kepada siswa dan staf sekolah. Sistem ini mencakup berbagai modul penting, seperti perpustakaan, penilaian akademik, tagihan, kehadiran, layanan jemputan, dan lainnya. Pada platform Odoo, modul-modul ini digunakan oleh staf pengajar dan

administrasi untuk memudahkan pengelolaan data, mulai dari pembaruan nilai siswa hingga pemantauan kehadiran dan pengelolaan layanan transportasi sekolah.

Aplikasi mobile didesain agar siswa dapat mengakses informasi pribadi yang berasal dari sistem Odoo secara cepat dan real-time. Siswa dapat melihat data nilai, status kehadiran, rincian tagihan, dan detail lainnya dengan mudah. Melalui integrasi ini, informasi yang ada di Odoo dapat diolah dan disajikan secara langsung pada aplikasi mobile, yang memungkinkan siswa untuk selalu mendapatkan informasi terkini mengenai perkembangan akademis dan berbagai layanan sekolah lainnya kapan pun dibutuhkan.

3.2.2 Spesifikasi Input Data Ke Sistem

Selama pengembangan sistem, praktikan bersama pembimbing eksternal (Dalam hal ini adalah pihak sekolah) melakukan analisis kebutuhan data yang akan digunakan dalam sistem. Tahap ini sangat penting karena data yang dikumpulkan akan menjadi dasar untuk informasi yang disajikan kepada pengguna. Oleh karena itu, analisis ini disusun secara sistematis sebagai panduan dalam proses pengembangan. Kebutuhan data input ke dalam sistem melalui Odoo.

1. Data Kehadiran
   Data ini mencakup catatan kehadiran siswa, seperti tanggal, waktu, serta status (hadir, izin, sakit, atau alpa), yang akan disinkronkan dari Odoo dan ditampilkan pada aplikasi mobile.

2. Data Siswa
   Informasi lengkap mengenai siswa, meliputi nama, kelas, nomor induk siswa (NIS), tanggal lahir, dan kontak orang tua. Data ini penting untuk identifikasi siswa di berbagai modul aplikasi mobile.

3. Data Nilai Siswa
   Data nilai untuk setiap mata pelajaran dan semester, termasuk nilai ujian dan tugas. Informasi ini akan diambil dari Odoo dan disajikan di aplikasi mobile sebagai gambaran perkembangan akademik siswa.

4. Data Tagihan Siswa

   Rincian tagihan sekolah, seperti SPP, biaya kegiatan, dan pembayaran lainnya. Data ini mencakup status pembayaran, jumlah tagihan, dan tanggal jatuh tempo, sehingga siswa dan orang tua dapat memantau status keuangan melalui aplikasi mobile.

5. Data Penggunaan Jasa Sekolah

   Informasi terkait layanan tambahan yang disediakan oleh sekolah, seperti transportasi (jemputan) dan ekstrakurikuler. Data ini mencakup jadwal, jenis layanan, serta rincian biaya untuk layanan yang digunakan siswa.

6. Data Informasi Jadwal Tugas dan Kegiatan Sekolah

   Data terkait pengumuman penting dari sekolah, seperti Tugas, lomba, dan acara-acara lainnya yang melibatkan siswa. Informasi ini akan disajikan di aplikasi mobile agar siswa selalu mendapatkan pembaruan mengenai aktivitas sekolah.

3.2.3 Spesifikasi Kebutuhan Output Data ke Sistem

Dalam pengembangan sistem ini, praktikan bersama pembimbing eksternal melakukan analisis mengenai data yang akan ditampilkan pada aplikasi mobile untuk siswa. Tahap ini menjadi penting karena data tersebut akan mendukung kebutuhan informasi yang relevan bagi siswa dan mempermudah akses terhadap perkembangan akademik dan layanan sekolah. Oleh karena itu, tahap analisis ini disusun secara sistematis agar dapat menjadi pedoman dalam pengembangan sistem. Output data yang akan ditampilkan pada aplikasi mobile untuk siswa adalah sebagai berikut.

1. Informasi Kehadiran Siswa

   Menyajikan data kehadiran siswa secara rinci, termasuk tanggal, waktu, dan status kehadiran, yang ditampilkan dalam format yang mudah dipahami.

2. Rincian Nilai Akademik

   Menampilkan nilai-nilai akademik siswa untuk setiap mata pelajaran dan semester, termasuk nilai tugas dan ujian, dalam bentuk grafik atau tabel agar siswa dapat memantau perkembangan akademis mereka.

3. Status dan Rincian Tagihan

   Memberikan informasi terkait tagihan sekolah, seperti SPP dan biaya kegiatan, beserta status pembayaran dan rincian lainnya, sehingga siswa dapat mengakses informasi keuangan secara praktis.

4. Status penggunaan Layanan Sekolah

   Menampilkan data mengenai layanan tambahan yang diakses siswa, seperti layanan transportasi dan ekstrakurikuler, dengan informasi jadwal dan rincian layanan.

5. Informasi Jadwal Tugas, Kegiatan dan Pengumuman Sekolah

   Menyajikan pengumuman penting dari sekolah mengenai tugas, kegiatan, lomba, dan acara lainnya yang melibatkan siswa, agar siswa dapat mengikuti dan berpartisipasi aktif dalam berbagai aktivitas sekolah.

3.2.4 Rekayasa Proses

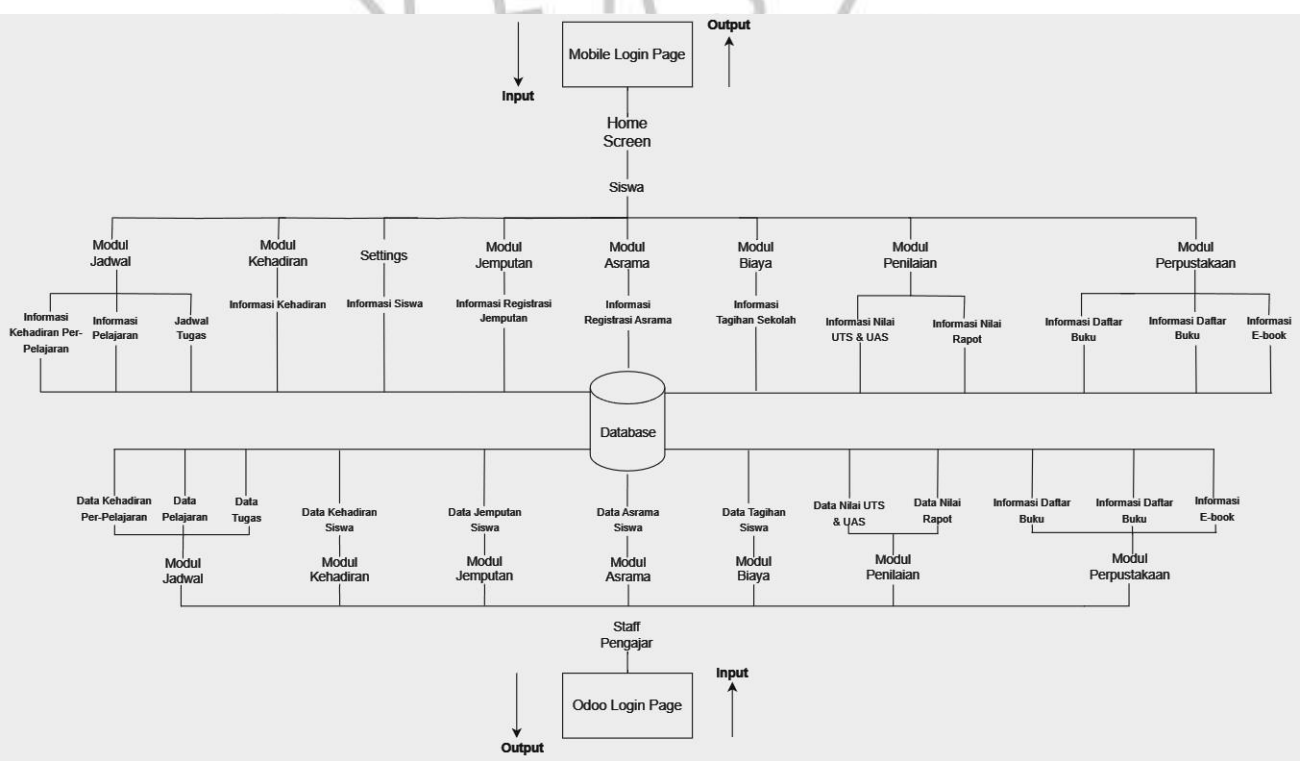
Dalam pengembangan sistem ini, praktikan melakukan analisis terhadap berbagai proses yang terjadi pada integrasi antara Odoo dan aplikasi mobile. Sistem ini dimulai dari proses input data yang dilakukan oleh staf sekolah melalui platform Odoo. Staf, seperti guru dan administrator, akan memasukkan berbagai data terkait siswa, termasuk catatan kehadiran dalam kegiatan absensi, jadwal tugas, penilaian tugas, informasi kegiatan sekolah, serta data lain yang relevan bagi siswa.

Setelah data dimasukkan ke dalam Odoo, informasi tersebut akan disimpan dalam database terpusat. Data-data yang telah tersimpan kemudian dapat diakses melalui endpoint yang telah dikonfigurasi, memungkinkan aplikasi mobile untuk menarik data secara langsung dari Odoo. Dengan menggunakan endpoint ini, aplikasi mobile dapat menampilkan informasi yang dibutuhkan oleh siswa, seperti status kehadiran, nilai tugas, jadwal, tagihan, dan berbagai pengumuman sekolah.

Sebelum siswa dapat melihat informasi tersebut di aplikasi mobile, mereka akan diminta untuk login menggunakan email dan password yang telah terdaftar dalam sistem. Dengan demikian, aplikasi mobile dapat menyajikan data secara personal dan aman sesuai dengan informasi masing-masing siswa.

3.2.5 Perancangan Sistem

Pada tahap perancangan ini, setelah analisis kebutuhan perangkat dan data yang diperlukan diselesaikan oleh praktikan, langkah-langkah strategis selanjutnya diambil secara hati-hati untuk memastikan bahwa sistem yang dibangun dapat berfungsi sesuai dengan ekspektasi. Analisis yang telah dilakukan dijadikan sebagai acuan utama dalam membentuk struktur sistem. sehingga diharapkan sistem ini mampu memenuhi kebutuhan sekolah dalam menyajikan informasi yang relevan, akurat, dan selalu diperbarui bagi siswa serta pihak terkait.

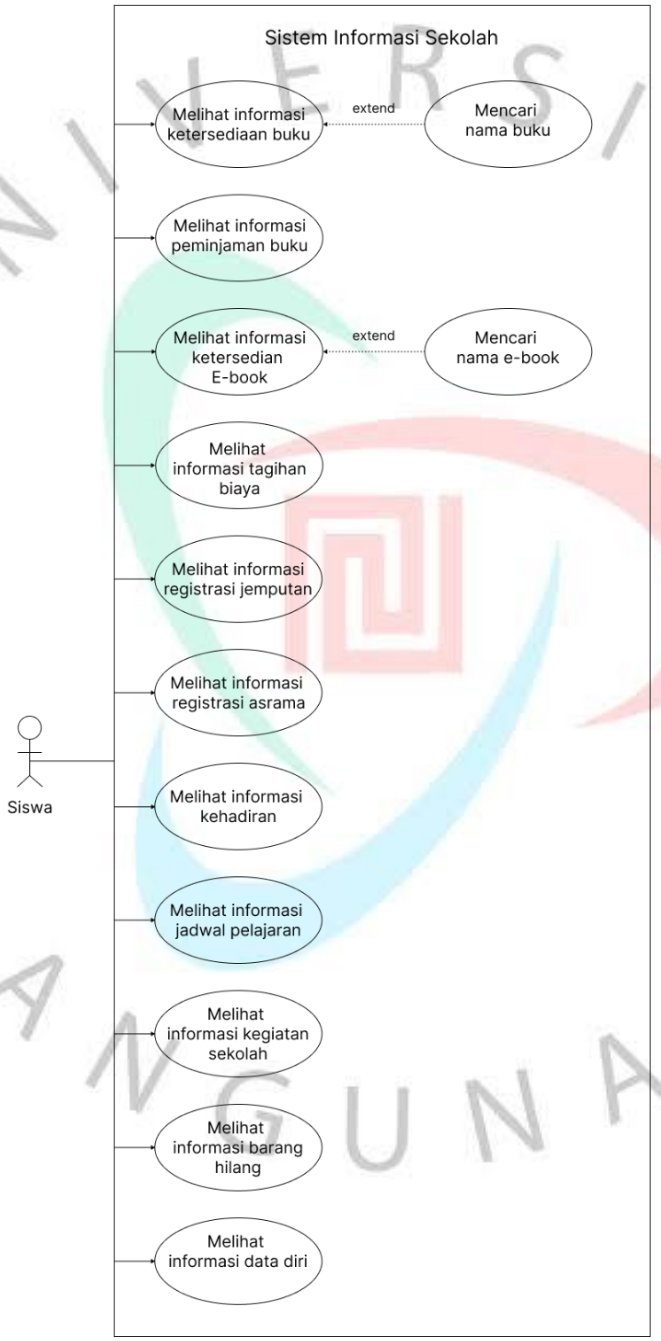


Gambar 3.1 Diagram Struktur Aplikasi

Gambar 3.1 memberikan pandangan menyeluruh mengenai struktur dasar dari sistem informasi sekolah yang akan dikembangkan, di mana alur data antara platform Odoo dan aplikasi mobile dijelaskan secara terperinci. Diagram ini memperlihatkan bagaimana modul-modul seperti Jadwal, Kehadiran, Jemputan, Asrama, Biaya, Penilaian, dan Perpustakaan dikelola oleh staf pengajar melalui Odoo dan kemudian disinkronkan ke aplikasi mobile yang diakses oleh siswa. Data yang dimasukkan oleh staf di Odoo disimpan dalam database terpusat, yang kemudian diakses oleh siswa melalui tampilan antarmuka mobile.

Struktur ini kemudian diuraikan lebih mendalam menggunakan berbagai alat perancangan seperti Unified Modeling Language (UML), diagram alir, MockUp, dan spesifikasi database. UML digunakan sebagai standar pemodelan untuk

menggambarkan kebutuhan dan arsitektur sistem berorientasi objek (Fu'adi & Prianggono, 2022), sementara MockUp berperan dalam visualisasi antarmuka dan alur kerja untuk memudahkan praktikan dalam memahami struktur sistem sebelum memasuki tahap implementasi. Selain itu, spesifikasi database dirancang agar data tersimpan dengan baik dan dapat diakses sesuai kebutuhan pengguna, memastikan sistem berjalan secara terstruktur dan efisien.

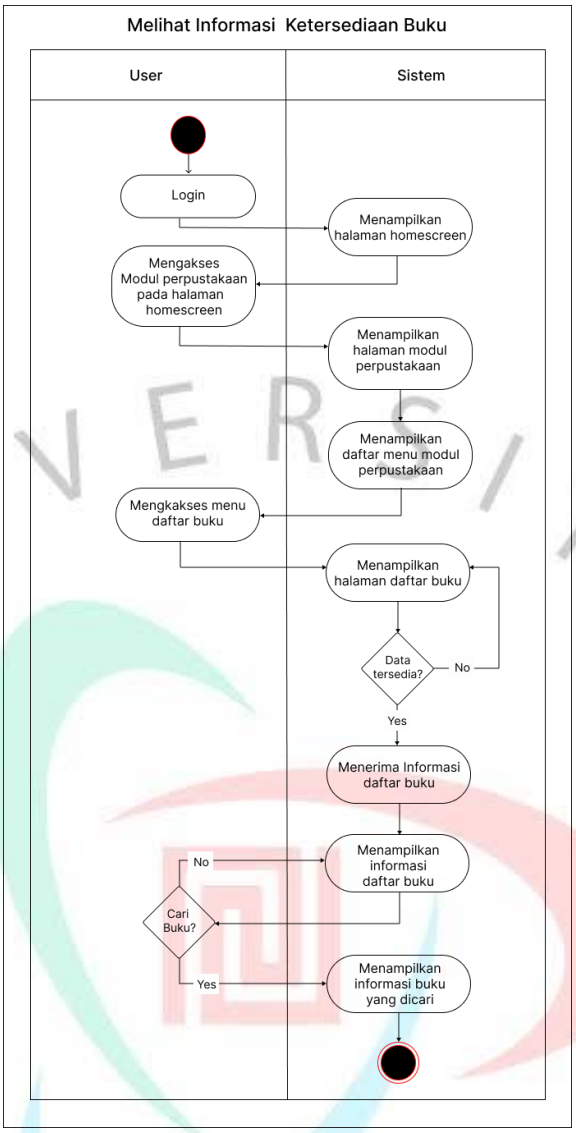


Gambar 3.2 *Use Case* Sistem untuk Siswa Sekolah

Gambar 3.2 menunjukkan use case diagram dari aplikasi Sistem Informasi Sekolah. Diagram ini dibuat dengan tujuan untuk menjelaskan berbagai fungsi yang dapat dilakukan oleh sistem dalam mendukung operasional sekolah. Dalam diagram tersebut, terdapat 11 use case yang masing-masing menggambarkan aktivitas atau layanan yang disediakan oleh sistem untuk para pengguna, baik siswa maupun staf sekolah. Setiap use case nantinya akan dijelaskan secara lebih terperinci melalui activity diagram untuk memberikan pemahaman yang lebih jelas tentang bagaimana alur proses berlangsung dan bagaimana interaksi antara pengguna dengan sistem terjadi. Diagram ini juga membantu mengidentifikasi peran dan tanggung jawab yang berbeda di dalam sistem.

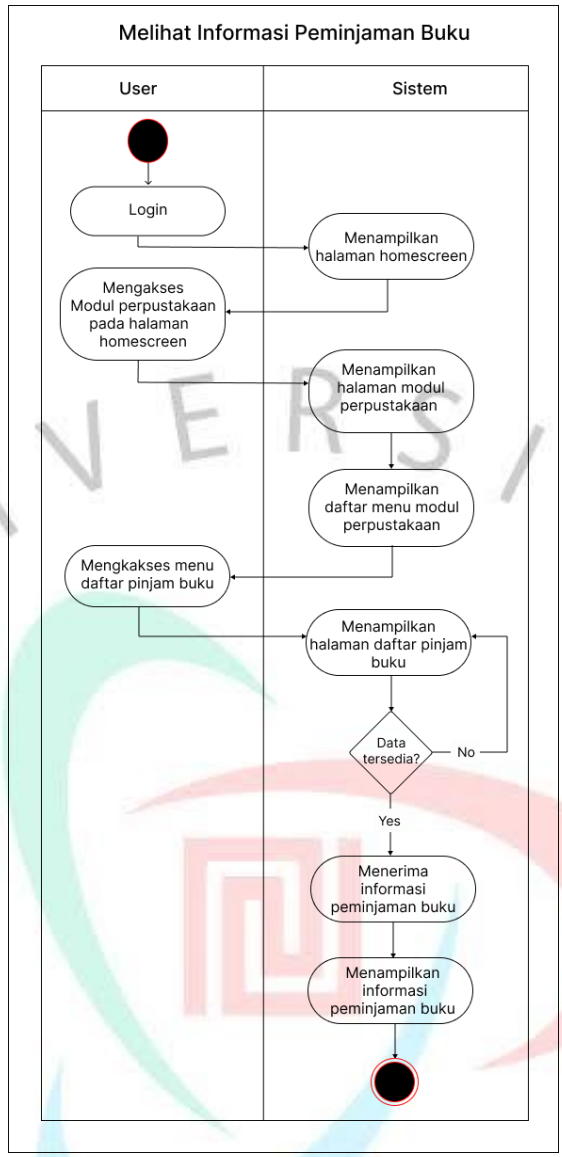
3.2.5.1Activity Diagram

Praktikan membuat Activity Diagram yang bertujuan untuk memetakan aktifitas dari sistem pada setiap halaman. Diagram tersebut membantu dalam memahami kondisi setiap komponen sistem berinteraksi serta proses yang terjadi di setiap langkah selama pengguna berinteraksi dengan aplikasi (Lnu, 2020).



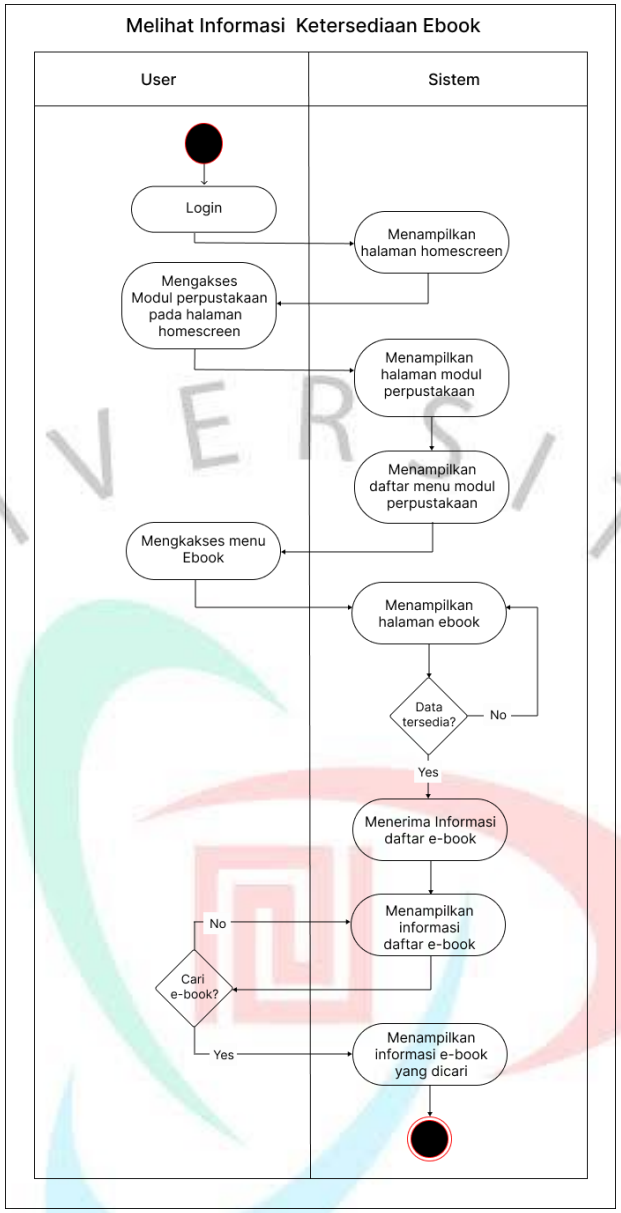
Gambar 3.3 *Activity Diagram* Informasi Ketersediaan Buku

Proses dimulai dengan pengguna yang melakukan login ke dalam sistem. Setelah berhasil masuk, pengguna mengakses modul perpustakaan melalui halaman homescreen. Sistem kemudian menampilkan daftar menu yang terkait. Pengguna memilih untuk mengakses menu daftar buku, dan sistem menampilkan halaman daftar buku yang tersedia. Jika data tersedia, pengguna menerima informasi terkait daftar buku yang ada. Selanjutnya, pengguna dapat memutuskan untuk mencari buku tertentu. Jika pengguna memilih untuk mencari buku, sistem akan menampilkan informasi buku yang dicari tersebut.



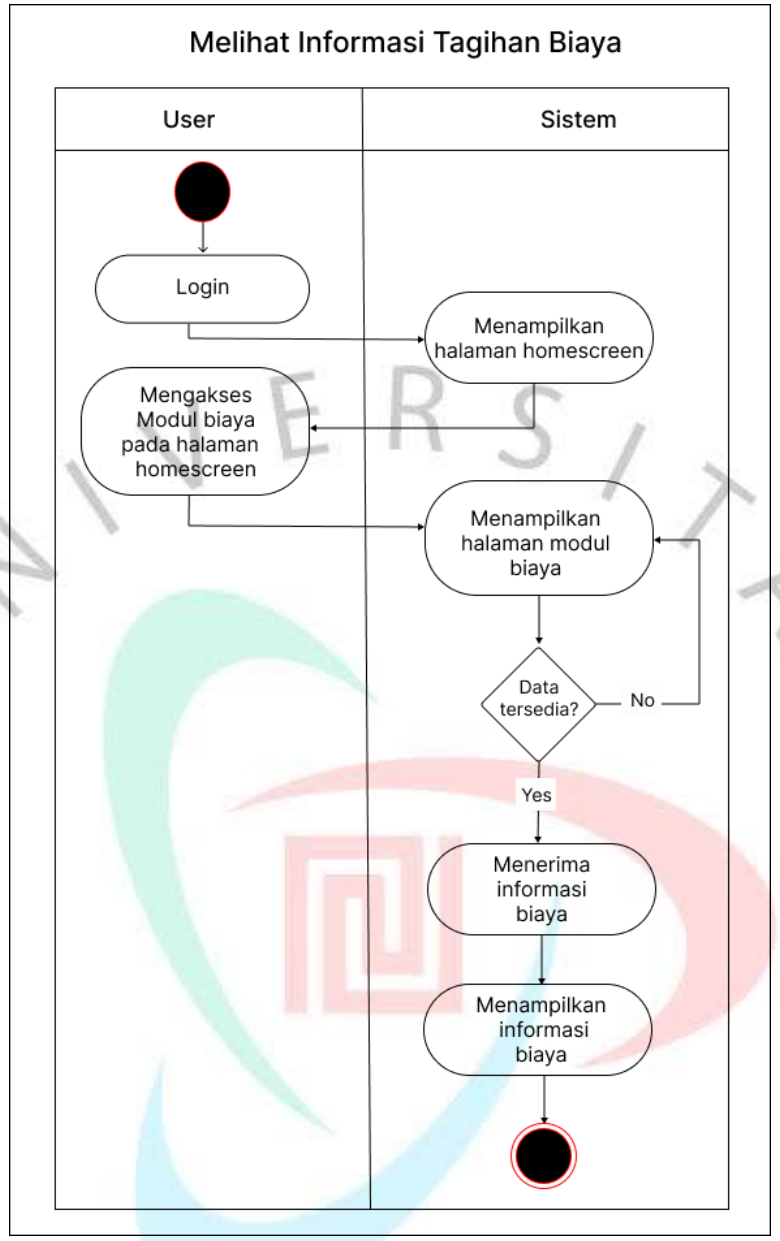
Gambar 3.4 *Activity Diagram* Informasi Peminjaman Buku

Activity diagram ini menjelaskan alur proses use case "Melihat Informasi Peminjaman Buku" dalam Sistem Informasi Sekolah. Proses dimulai ketika pengguna melakukan login ke dalam sistem. Setelah berhasil masuk, pengguna mengakses modul perpustakaan dari halaman homescreen. Sistem kemudian menampilkan halaman modul perpustakaan beserta daftar menu yang tersedia di dalam modul tersebut. Pengguna selanjutnya memilih untuk mengakses menu daftar pinjam buku, dan sistem menampilkan halaman daftar pinjam buku. Jika data tersedia, pengguna akan menerima informasi terkait peminjaman buku. Sistem kemudian menampilkan informasi detail peminjaman buku tersebut kepada pengguna.



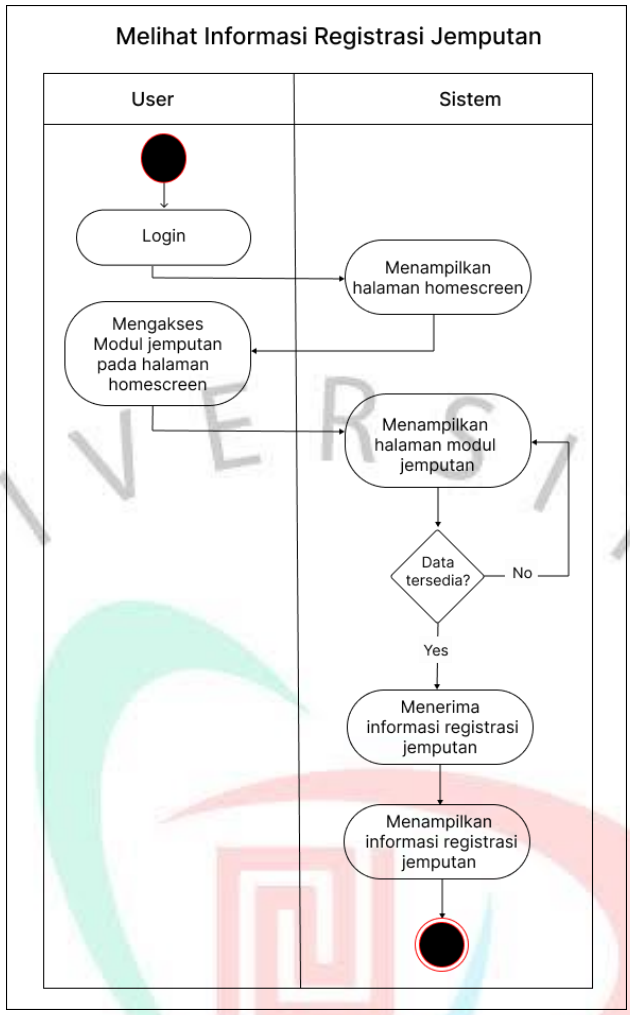
Gambar 3.5 *Activity Diagram* Informasi Ketersediaan E-book

Gambar 3.5 adalah activity diagram yang menggambarkan alur proses melihat ketersediaan e-book. Proses dimulai dengan pengguna melakukan login ke dalam sistem. Setelah login berhasil, pengguna mengakses modul perpustakaan dari halaman homescreen. Sistem kemudian menampilkan halaman modul perpustakaan beserta daftar menu yang tersedia di dalamnya. Pengguna selanjutnya memilih untuk mengakses menu ebook, dan sistem menampilkan halaman yang berisi daftar ebook. Sistem memeriksa apakah data ebook tersedia. Jika data tersedia, pengguna menerima informasi tentang daftar ebook. Pengguna kemudian memiliki opsi untuk mencari ebook tertentu. Jika pengguna memilih untuk mencari, sistem akan menampilkan informasi ebook yang dicari.



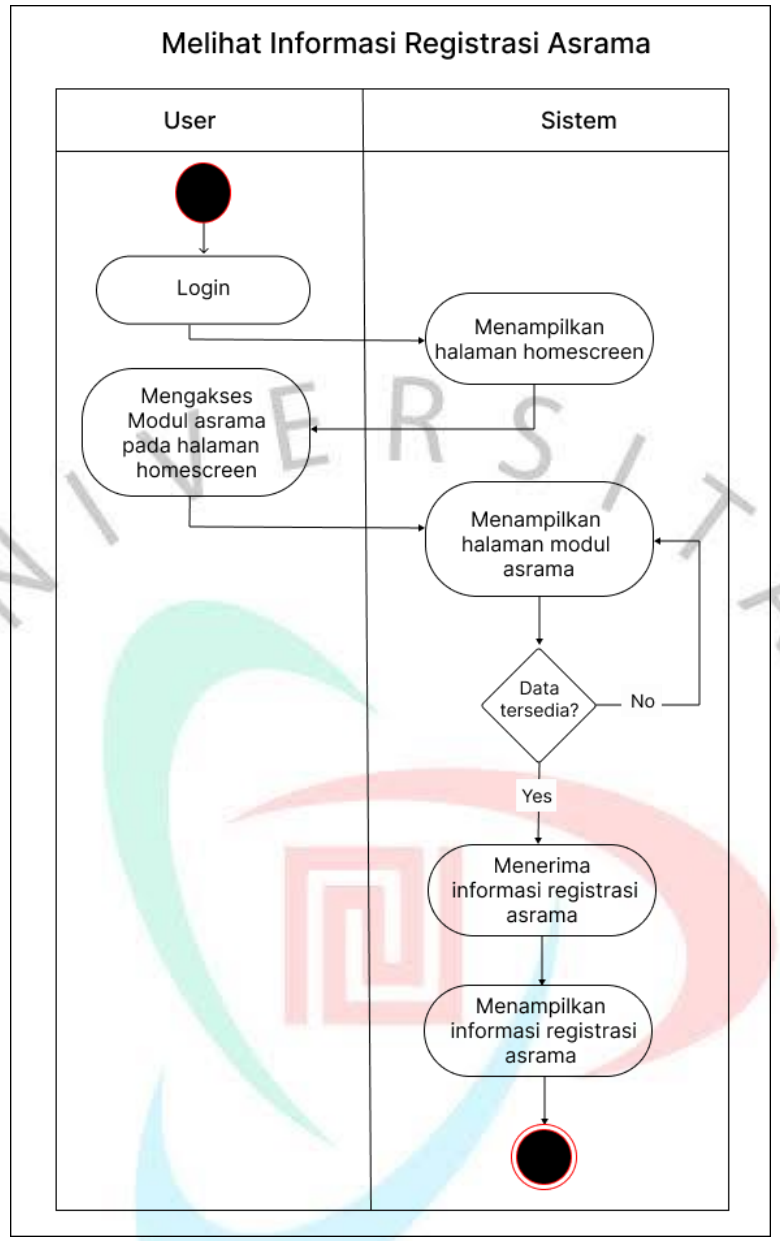
Gambar 3.6 *Activity Diagram* Informasi Tagihan Biaya

Gambar 3.6 menjelaskan tentang alur proses dari melihat halaman tagihan siswa. Proses dimulai ketika pengguna melakukan login ke sistem. Setelah berhasil masuk, pengguna mengakses modul biaya melalui halaman homescreen. Sistem kemudian menampilkan halaman modul biaya tersebut. Selanjutnya, sistem memeriksa ketersediaan data tagihan biaya. Jika data tidak tersedia, sistem tidak akan menampilkan informasi tambahan. Namun, jika data tersedia, pengguna akan menerima informasi mengenai tagihan biaya, dan sistem akan menampilkan detail informasi tersebut.



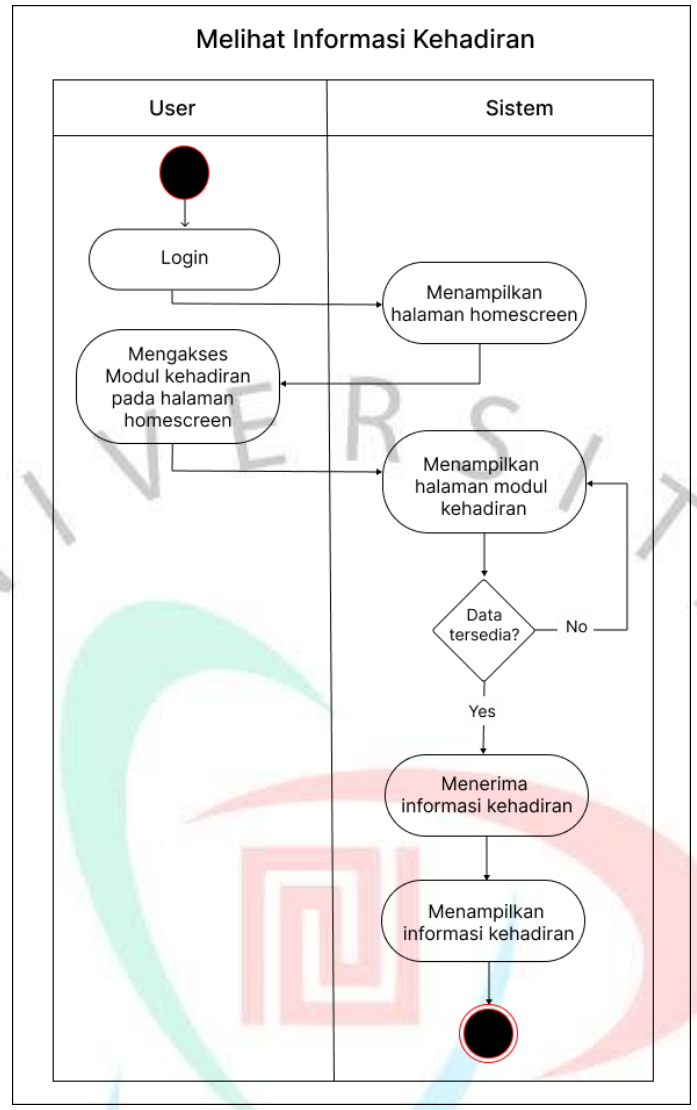
Gambar 3.7 *Activity Diagram* Informasi Registrasi Jemputan

Activity diagram ini menggambarkan alur proses use case "Melihat Informasi Registrasi Jemputan" dalam Sistem Informasi Sekolah. Alur ini dimulai dengan pengguna yang melakukan login ke dalam sistem untuk mengamankan akses ke fitur yang tersedia. Setelah proses login berhasil, pengguna diarahkan ke halaman homescreen. Pengguna kemudian memilih modul jemputan dari daftar menu yang ada pada halaman homescreen. Sistem merespons tindakan tersebut dengan menampilkan halaman modul jemputan, yang berisi opsi dan informasi terkait layanan jemputan sekolah. Di tahap ini, sistem secara otomatis memeriksa ketersediaan data registrasi jemputan. Jika data registrasi jemputan tidak ditemukan, proses berakhir di tahap ini tanpa menampilkan informasi lebih lanjut kepada pengguna. Namun, jika data tersedia, sistem akan memproses dan mengirimkan informasi registrasi jemputan.



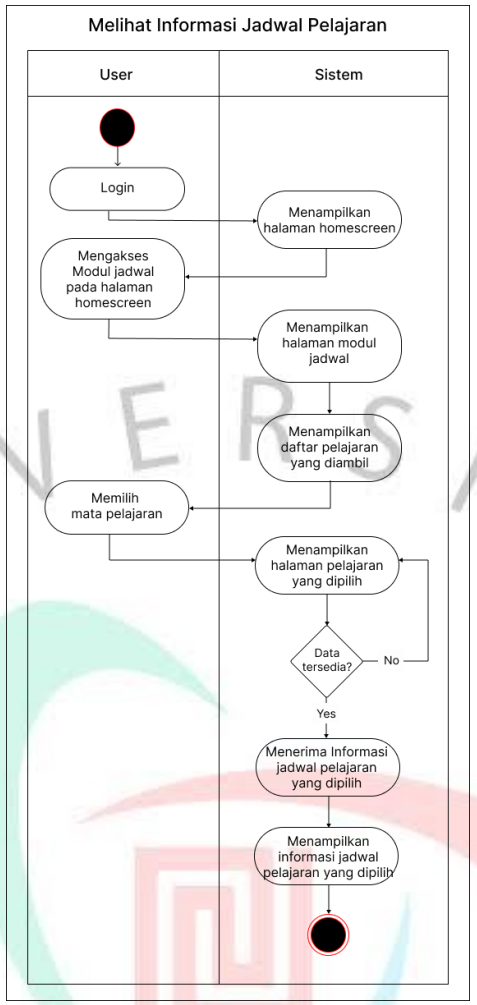
Gambar 3.8 *Activity Diagram* Informasi Registrasi Asrama

Proses ini dimulai dengan pengguna melakukan login ke dalam sistem untuk memastikan bahwa mereka memiliki akses yang sah ke fitur-fitur yang ada. Setelah berhasil login, pengguna diarahkan ke halaman homescreen, yang berisi berbagai modul yang dapat diakses. Pengguna kemudian memilih untuk mengakses modul asrama yang ada di halaman homescreen. Sistem merespons tindakan pengguna dengan menampilkan halaman modul asrama. Di tahap ini, sistem melakukan pengecekan terhadap ketersediaan data registrasi asrama yang relevan. Jika data yang dicari tidak tersedia, sistem tidak akan menampilkan informasi lebih lanjut, dan proses berakhir di sini. Namun, jika data registrasi asrama tersedia, sistem mengirimkan informasi tersebut kepada pengguna. Pengguna menerima informasi ini, dan sistem menampilkan detail lengkap registrasi asrama.



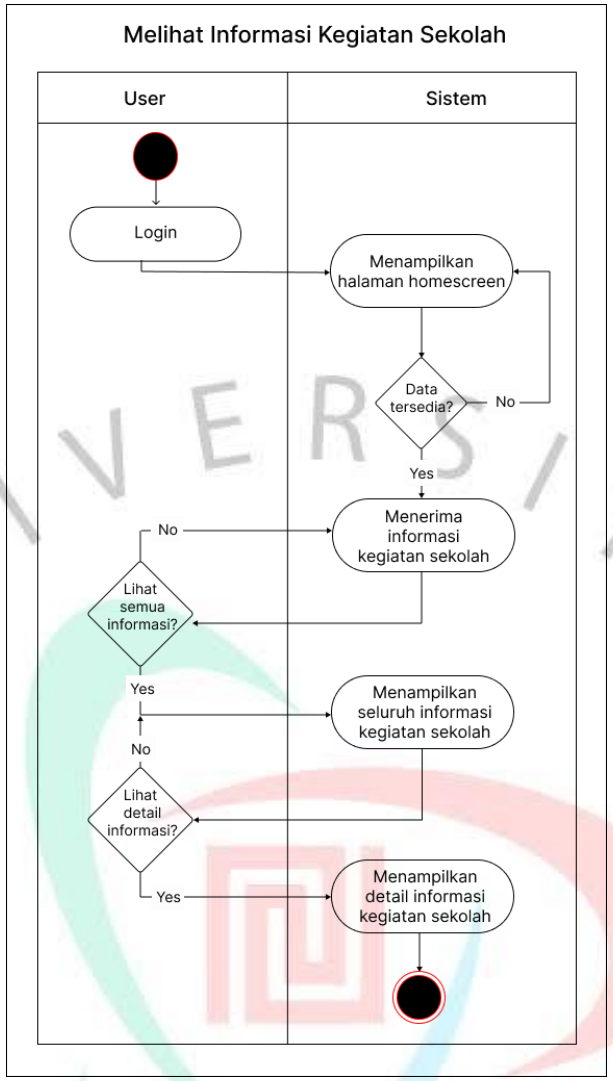
Gambar 3.9 *Activity Diagram* Informasi Kehadiran

Proses dimulai dengan pengguna melakukan login ke dalam sistem untuk mendapatkan akses ke fitur yang tersedia. Setelah proses login berhasil, pengguna diarahkan ke halaman homescreen, di mana berbagai modul dapat diakses. Pengguna kemudian memilih untuk mengakses modul kehadiran dari halaman homescreen. Sistem merespons dengan menampilkan halaman modul kehadiran yang berisi opsi terkait informasi kehadiran. Setelah halaman modul kehadiran terbuka, sistem memeriksa ketersediaan data kehadiran. Jika data kehadiran tidak tersedia, sistem tidak menampilkan informasi lebih lanjut, dan alur proses berhenti di sini. Namun, jika data tersedia, sistem mengirimkan informasi tersebut ke pengguna, dan pengguna menerima data terkait kehadiran. Sistem kemudian menampilkan detail informasi kehadiran tersebut kepada pengguna.



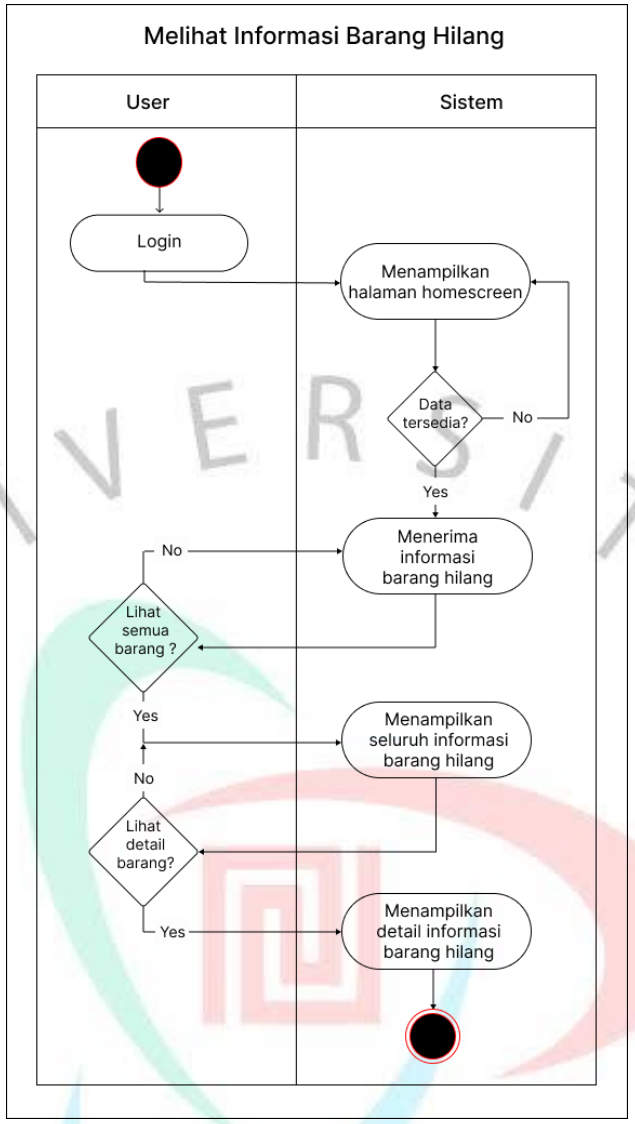
Gambar 3.10 *Activity Diagram* Melihat Jadwal Pelajaran

Proses pada gambar 3.10 dimulai dengan pengguna melakukan login ke dalam sistem untuk memastikan akses yang aman dan sah. Setelah login berhasil, pengguna diarahkan ke halaman homescreen, di mana berbagai modul yang tersedia dalam sistem dapat diakses. Pengguna kemudian memilih untuk mengakses modul jadwal pelajaran dari halaman homescreen. Sistem merespons tindakan ini dengan menampilkan halaman modul jadwal yang berisi daftar mata pelajaran yang telah diambil oleh pengguna. Selanjutnya, pengguna memilih mata pelajaran tertentu yang ingin dilihat jadwalnya. Setelah pengguna memilih mata pelajaran, sistem menampilkan halaman dengan detail jadwal untuk mata pelajaran yang dipilih. Di tahap ini, sistem memeriksa apakah data jadwal tersedia. Jika data tidak tersedia, sistem tidak akan menampilkan informasi lebih lanjut, dan proses berhenti di sini. Namun, jika data tersedia, sistem mengirimkan informasi jadwal kepada pengguna, dan pengguna menerima detail tersebut. Sistem kemudian menampilkan informasi lengkap mengenai jadwal pelajaran yang dipilih.



Gambar 3.11 *Activity Diagram* Informasi Kegiatan Sekolah

Proses dimulai dengan pengguna melakukan login untuk mendapatkan akses ke sistem. Setelah login berhasil, pengguna diarahkan ke halaman homescreen, di mana berbagai opsi modul tersedia. Sistem kemudian memeriksa ketersediaan data kegiatan sekolah. Jika data tidak tersedia, proses berhenti di sini, dan pengguna tidak dapat melanjutkan. Namun, jika data kegiatan sekolah tersedia, pengguna menerima informasi tersebut. Pengguna kemudian dapat memilih apakah ingin melihat seluruh informasi kegiatan sekolah atau tidak. Jika tidak, proses berakhir di sini. Jika pengguna memilih untuk melihat seluruh informasi, sistem akan menampilkan semua data terkait kegiatan sekolah. Selanjutnya, pengguna dapat memilih apakah ingin melihat detail informasi dari salah satu kegiatan tertentu. Jika pengguna memilih untuk tidak melihat detail, proses berhenti di sini. Jika pengguna ingin melihat lebih lanjut, sistem menampilkan detail informasi kegiatan sekolah yang diinginkan.



Gambar 3.12 *Activity Diagram* Informasi Barang Hilang

Gambar 3.12 dimulai dengan pengguna melakukan login ke dalam sistem untuk mengamankan akses ke fitur yang tersedia. Setelah login berhasil, pengguna diarahkan ke halaman homescreen. Sistem kemudian memeriksa apakah data terkait informasi barang hilang tersedia. Jika data tidak ditemukan, proses berakhir di sini. Jika data tersedia, sistem mengirimkan informasi tersebut kepada pengguna, dan pengguna menerima data barang hilang yang terdaftar. Selanjutnya, pengguna diberikan opsi untuk melihat semua data barang hilang. Jika pengguna memilih untuk tidak melihat semua data, proses berakhir di tahap ini. Jika pengguna memutuskan untuk melihat seluruh informasi barang hilang, sistem akan menampilkan data lengkap barang hilang. Pengguna kemudian memiliki opsi untuk melihat detail lebih lanjut tentang barang tertentu. Jika pengguna tidak ingin melihat detail, proses selesai di sini. Namun, jika pengguna ingin melihat rincian lebih lanjut, sistem menampilkan informasi detail tentang barang hilang tersebut.

3.2.5.2Wireframe

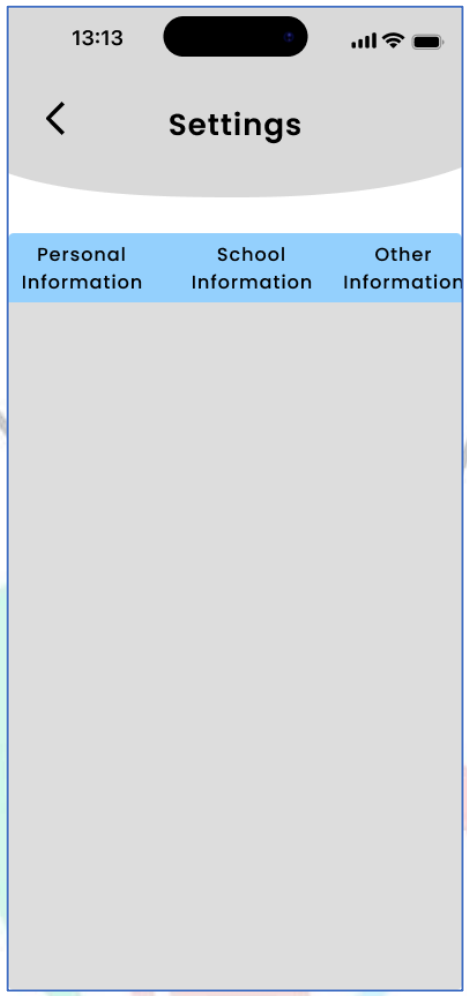
Setelah menyelesaikan perancangan use case diagram, praktikan melanjutkan ke tahap pengembangan sistem dengan membuat desain awal atau wireframe. Wireframe digunakan untuk memberikan kerangka awal gambaran visual mengenai tampilan dan alur kerja sistem yang akan dikembangkan (Hartawan, 2022), serta membantu praktikan memahami konsep sistem sebelum memasuki tahap implementasi dalam bentuk kode program.

Gambar 3.13 Wireframe Login

Wireframe ini menampilkan desain halaman login untuk Sistem Informasi Sekolah. Halaman login ini memiliki tampilan dengan logo sistem di bagian atas, diikuti dengan judul "Login" dan sambutan singkat "Selamat datang di Sistem Informasi Sekolah." Di bawahnya, terdapat dua field input untuk "Email" dan "Password," serta ikon mata pada field password untuk mengatur visibilitas kata sandi. Di bawah field password, ada opsi "Forgot password?" untuk pemulihan kata sandi. Tombol "SIGN IN" ditempatkan di bagian bawah untuk mengakses sistem setelah data login dimasukkan.

Gambar 3.14 Wireframe Homescreen

Wireframe ini menampilkan halaman homescreen aplikasi Sistem Informasi Sekolah. Di bagian atas terdapat profil pengguna dengan nama, foto, dan opsi keluar. Bagian utama berisi ikon-ikon modul utama seperti Perpustakaan, Siswa, Gedung, Jemputan, EduCore, Penilaian, Asrama, dan Kehadiran untuk akses cepat ke setiap fitur. Di bawahnya ada bagian "Informasi" dengan kartu-kartu informasi yang dapat dilihat lebih lengkap. Di bagian bawah, terdapat bilah navigasi dengan ikon untuk Beranda, Transaksi, Bantuan, dan Akun, memudahkan navigasi antar fitur utama.

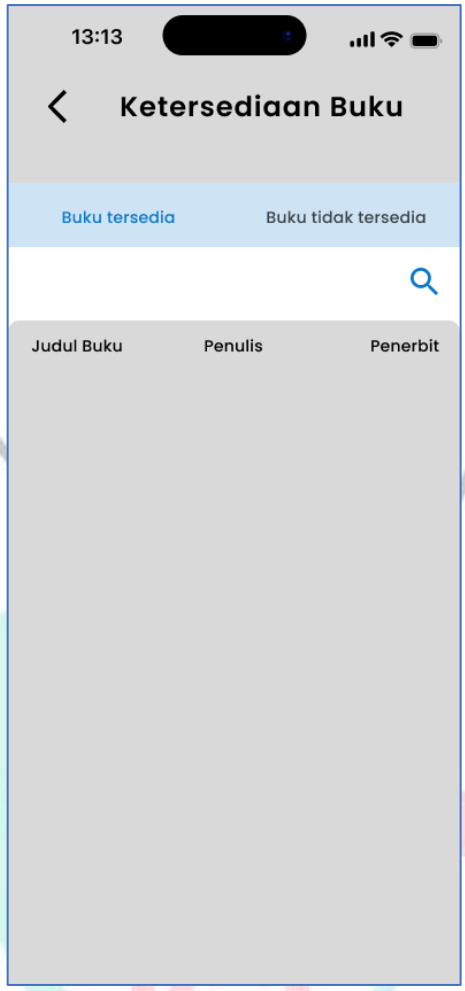


Gambar 3.15 Wireframe Settings

Wireframe ini menunjukkan halaman settings dalam aplikasi, dengan tampilan sederhana yang memuat tiga tab utama di bagian atas: Personal Information, School Information, dan Other Information. Setiap tab memungkinkan pengguna untuk mengakses informasi terkait sesuai kategorinya. Terdapat tombol "Back" di sudut kiri atas untuk kembali ke halaman sebelumnya. Desain ini mempermudah pengguna dalam menavigasi pengaturan informasi pribadi, sekolah, dan lainnya dalam satu halaman.

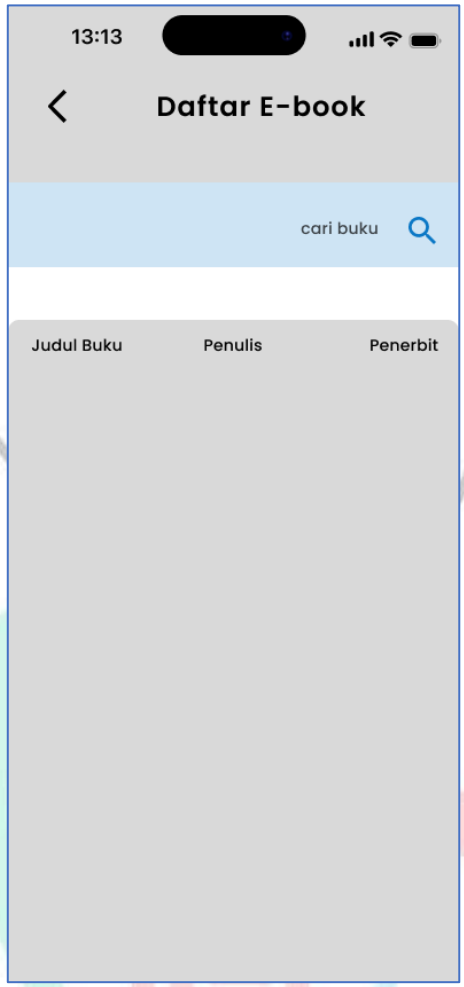
Gambar 3.16 Wireframe Modul Perpustakan

Wireframe ini menampilkan halaman modul perpustakaan dengan judul "Daftar Pinjam Buku." Terdapat tiga pilihan menu utama: Daftar Pinjam Buku untuk melihat peminjaman buku fisik, Daftar E-book untuk melihat koleksi ebook perpustakaan, dan Peminjaman Buku untuk melihat jadwal peminjaman buku. Setiap menu dilengkapi dengan ikon ilustratif dan deskripsi singkat yang memudahkan pengguna memahami fungsi masing-masing. Tombol "Back" di sudut kiri atas memungkinkan pengguna kembali ke halaman sebelumnya.



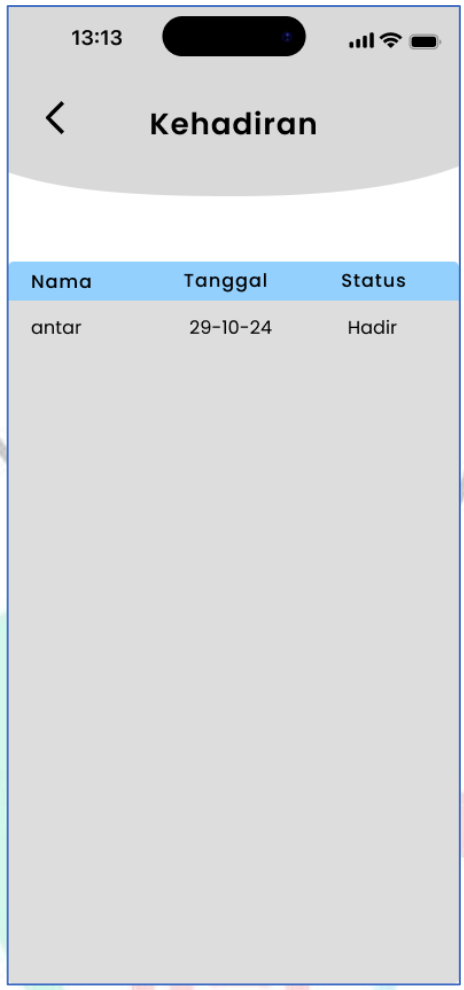
Gambar 3.17 Wireframe Daftar Buku

Wireframe ini menampilkan halaman Ketersediaan Buku yang memungkinkan pengguna melihat daftar buku berdasarkan ketersediaannya. Di bagian atas, terdapat dua tab: Buku tersedia dan Buku tidak tersedia, yang memudahkan pengguna dalam menyaring buku sesuai status ketersediaan. Ada ikon pencarian untuk memudahkan pencarian spesifik berdasarkan judul, penulis, atau penerbit. Tabel di bawahnya memiliki kolom Judul Buku, Penulis, dan Penerbit untuk menampilkan informasi detail dari setiap buku yang ada di perpustakaan. Tombol "Back" di kiri atas memungkinkan pengguna kembali ke halaman sebelumnya.



Gambar 3.18 Wireframe E-book

Wireframe ini menampilkan halaman Daftar E-book, yang dirancang untuk memudahkan pengguna dalam mencari dan melihat koleksi e-book. Di bagian atas terdapat bilah pencarian dengan placeholder cari buku serta ikon pencarian untuk mencari e-book berdasarkan kata kunci. Tabel di bawahnya memiliki kolom Judul Buku, Penulis, dan Penerbit yang akan menampilkan informasi detail dari e-book yang ada. Tombol "Back" di sudut kiri atas memungkinkan pengguna kembali ke halaman sebelumnya.

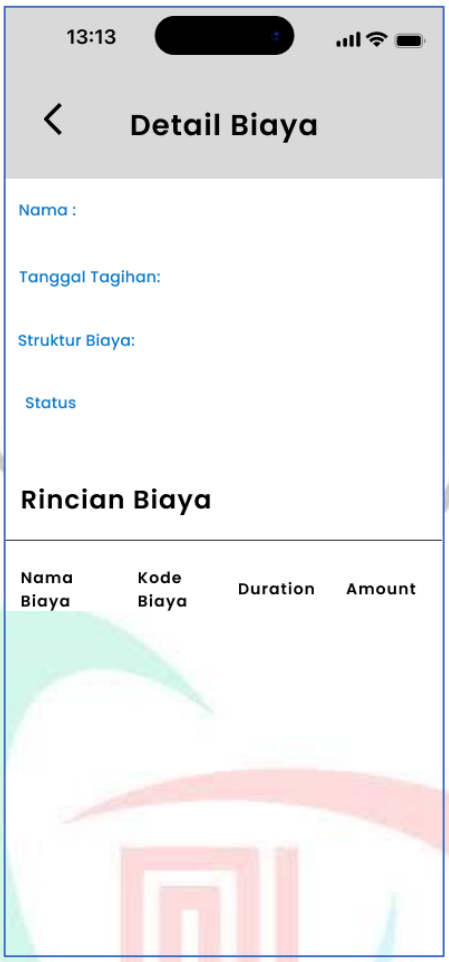


Gambar 3.19 Wireframe Modul Perpustakan

Wireframe ini menunjukkan halaman Kehadiran yang dirancang untuk menampilkan catatan kehadiran pengguna atau siswa. Di bagian atas terdapat judul "Kehadiran" dan tombol "Back" untuk kembali ke halaman sebelumnya. Tabel kehadiran menampilkan tiga kolom utama: Nama, Tanggal, dan Status, yang menunjukkan nama individu, tanggal kehadiran, dan status kehadirannya (contohnya: "Hadir").

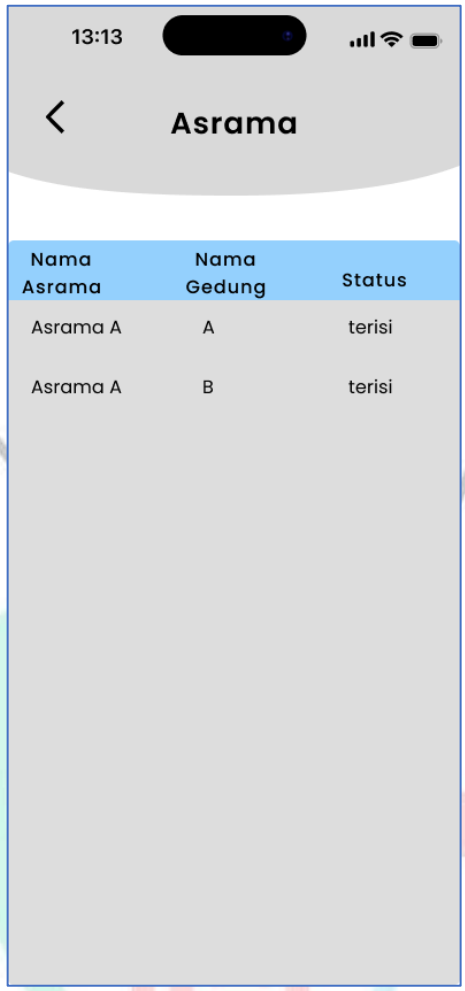
Gambar 3.20 Wireframe Modul Biaya

Wireframe ini menunjukkan halaman Biaya yang dirancang untuk menampilkan informasi tagihan pengguna. Di bagian atas terdapat judul "Biaya" dan tombol "Back" untuk kembali ke halaman sebelumnya. Tabel biaya memiliki tiga kolom utama: Nama Tagihan, Tanggal Tagihan, dan Status. Kolom ini menampilkan rincian seperti nama tagihan (contohnya: "Tagihan - Alifia"), tanggal tagihan, dan status pembayaran (contohnya: "Terbayar").



Gambar 3.21 Wireframe Modul Detail Biaya

Wireframe ini menunjukkan halaman Detail Biaya yang dirancang untuk menampilkan rincian lengkap terkait tagihan biaya. Di bagian atas terdapat informasi dasar seperti Nama, Tanggal Tagihan, Struktur Biaya, dan Status, yang memberikan gambaran umum tentang tagihan tersebut. Di bawahnya terdapat tabel Rincian Biaya dengan kolom Nama Biaya, Kode Biaya, Duration, dan Amount, yang menyajikan detail lebih spesifik mengenai biaya yang dikenakan. Tombol "Back" di kiri atas memungkinkan pengguna kembali ke halaman sebelumnya.

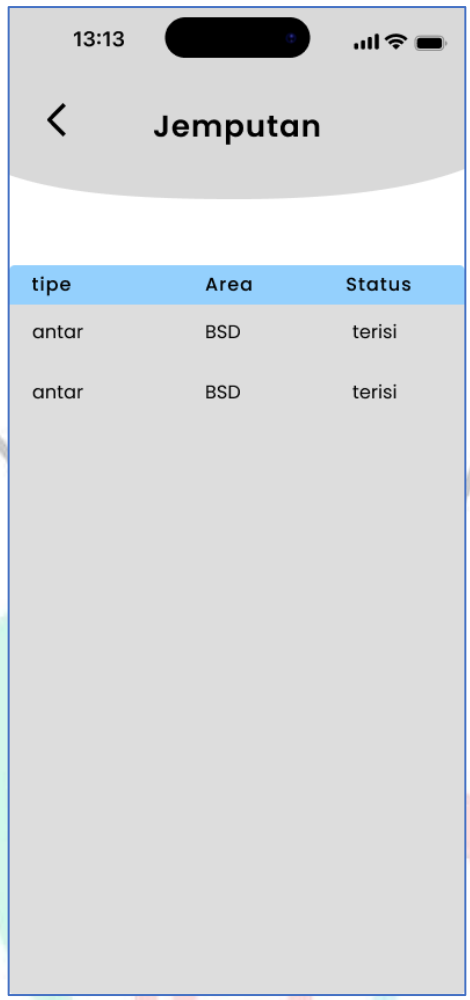


Gambar 3.22 Wireframe Modul Asrama

Wireframe ini menunjukkan halaman Asrama, yang dirancang untuk menampilkan informasi terkait asrama. Di bagian atas terdapat judul "Asrama" dan tombol "Back" untuk kembali ke halaman sebelumnya. Tabel utama memiliki tiga kolom: Nama Asrama, Nama Gedung, dan Status, yang menunjukkan nama asrama, gedung terkait, dan status keterisian (contohnya: "terisi").

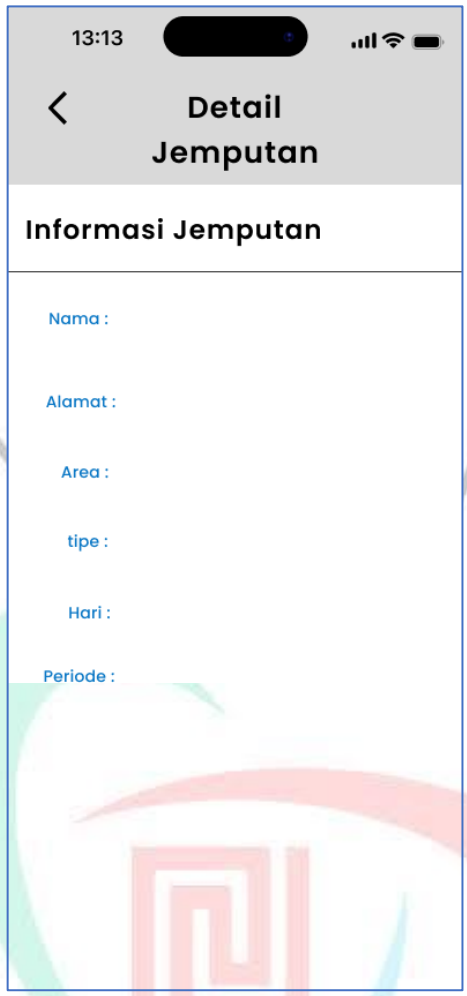
Gambar 3.23 Wireframe Modul Detail Asrama

Wireframe ini menampilkan halaman Detail Asrama, yang dirancang untuk memberikan informasi rinci mengenai asrama. Bagian Informasi Asrama mencakup berbagai detail seperti Nama, Asrama, Type, Gedung, Lantai, dan Fasilitas, yang membantu pengguna memahami deskripsi lengkap dari asrama tersebut. Di bawahnya, terdapat bagian Informasi Pendaftaran untuk menampilkan data terkait proses atau status pendaftaran asrama. Tombol "Back" di kiri atas memungkinkan pengguna kembali ke halaman sebelumnya.



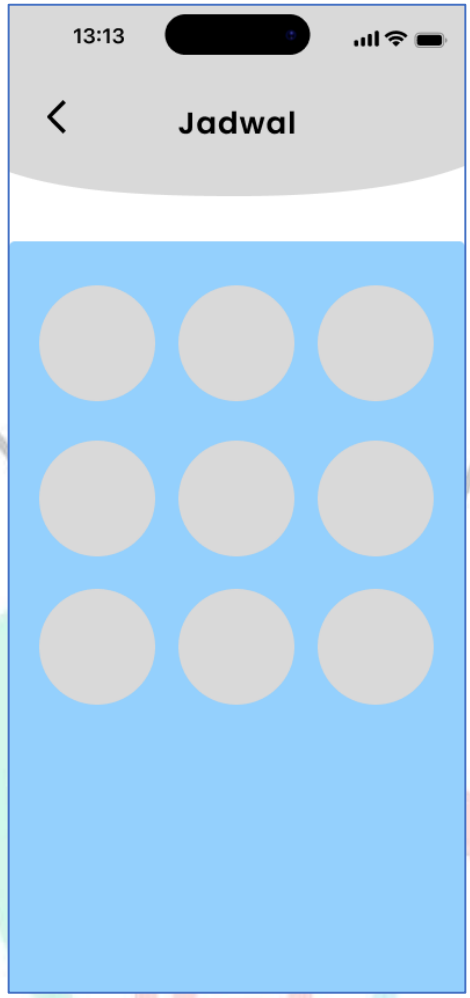
Gambar 3.24 Wireframe Modul Jemputan

Wireframe ini menampilkan halaman Jemputan, yang dirancang untuk memberikan informasi terkait layanan antar-jemput. Tabel di halaman ini memiliki tiga kolom utama: Tipe (jenis layanan seperti antar atau jemput), Area (lokasi layanan, seperti BSD), dan Status (contohnya: "terisi" untuk menunjukkan ketersediaan). Di bagian atas terdapat judul "Jemputan" dan tombol "Back" untuk kembali ke halaman sebelumnya.



Gambar 3.25 Wireframe Modul Detail Jemputan

Wireframe ini menampilkan halaman Detail Jemputan, yang dirancang untuk memberikan informasi rinci mengenai layanan antar-jemput. Di bagian Informasi Jemputan, terdapat berbagai detail seperti Nama, Alamat, Area (lokasi layanan), Tipe (jenis layanan seperti antar atau jemput), Hari (jadwal hari layanan), dan Periode (durasi atau waktu layanan). Di bagian atas, terdapat judul "Detail Jemputan" dan tombol "Back" untuk kembali ke halaman sebelumnya.



Gambar 3.26 Wireframe Jadwal Pelajaran

Wireframe ini menampilkan halaman Jadwal, yang dirancang untuk memberikan akses cepat ke berbagai jadwal. Di tengah halaman terdapat kumpulan ikon atau tombol berbentuk lingkaran yang kemungkinan mewakili kategori atau hari tertentu untuk jadwal, memudahkan pengguna dalam memilih jadwal yang ingin dilihat. Di bagian atas, terdapat judul "Jadwal" dan tombol "Back" untuk kembali ke halaman sebelumnya.

3.5.2.3Spesifikasi Database

Sistem yang praktikan kembangkan menggunakan Odoo sebagai platform utama dalam manajemen data dan proses bisnis. Odoo menyediakan struktur modular yang memungkinkan integrasi berbagai modul bisnis dengan efisien. Dalam penerapannya, sistem ini menggunakan PostgreSQL sebagai sistem manajemen basis data (DBMS) yang handal untuk mendukung penyimpanan, pemrosesan, dan manajemen data secara terstruktur. DBMS merupakah sebuah sistem yang dirancang untuk menangani proses manajemen basis data (Dirgantara et al,. 2023). PostgreSQL dipilih karena kemampuannya dalam menghandle pembuatan record data dalam skala besar (Praba & Safitri, 2020). PostgreSQL dipilih karena kompatibilitasnya dengan Odoo dan kemampuannya dalam menangani volume data yang besar, serta fitur-fitur keamanan dan performa yang mumpuni. Praktikan membuat tabel database disesuaikan dengan modul-modul yang dikembangkan pada sistem Odoo ini. Setiap modul memiliki tabel database yang dibangun berdasarkan struktur data dan kebutuhan spesifik modul tersebut. Struktur ini dirancang agar data pada setiap tabel dapat saling berhubungan dan mendukung proses bisnis yang berlangsung. Berikut adalah struktur database dari setiap tabel yang digunakan.

Table "public.asrama"

Column	Type	Collation	Nullable	Default
id	integer		not null	nextval('asrama_id_seq'::regclass)
kapasitas	integer			
gedung_id	integer			
lantai	integer			
total_terisi_siswa	integer			
total_reservasi_siswa	integer			
max_reservasi	integer			
create_uid	integer			
write_uid	integer			
name	character varying			
room_number	character varying			
tipe	character varying			
state	character varying			
create_date	timestamp without time zone			
write_date	timestamp without time zone			
panjang	double precision			
lebar	double precision			
luas	double precision			

Gambar 3.27 Tabel Basis Data Modul Asrama

Tabel asrama dalam database ini menyimpan data terkait informasi asrama dan registrasi asrama, seperti kapasitas maksimum penghuni (kapasitas), lokasi gedung (gedung_id), lantai (lantai), dan jumlah penghuni aktif serta reservasi (total_terisi_siswa dan total_reservasi_siswa). Kolom name dan room_number mencatat nama asrama dan nomor kamar, sedangkan tipe dan state menunjukkan kategori dan status asrama. Data terkait pembuatan dan perubahan tersimpan di

create_date dan write_date, serta dimensi fisik asrama tercatat pada kolom panjang, lebar, dan luas.

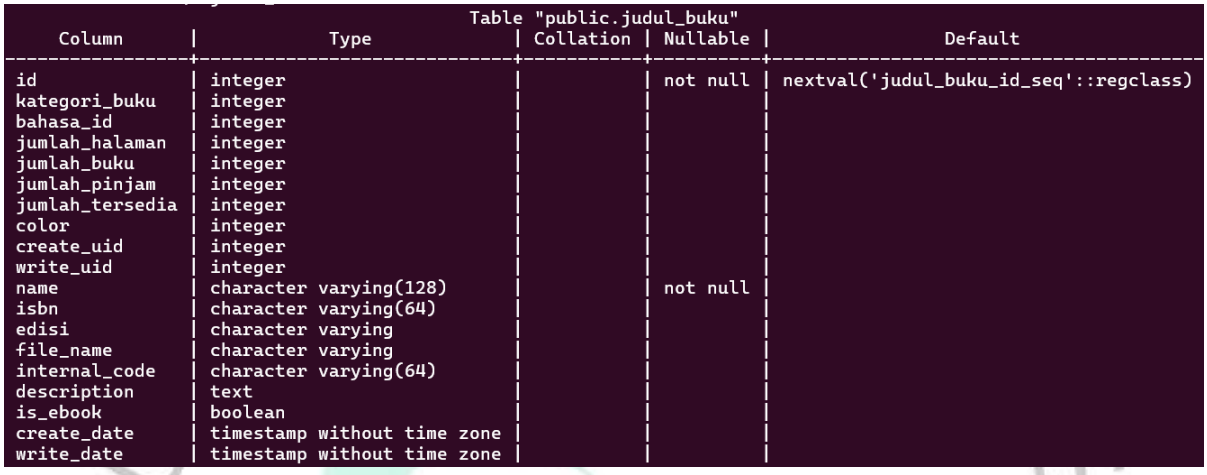


Table "public.judul_buku"

Column	Type	Collation	Nullable	Default
id	integer		not null	nextval('judul_buku_id_seq'::regclass)
kategori_buku	integer			
bahasa_id	integer			
jumlah_halaman	integer			
jumlah_buku	integer			
jumlah_pinjam	integer			
jumlah_tersedia	integer			
color	integer			
create_uid	integer			
write_uid	integer			
name	character varying(128)		not null	
isbn	character varying(64)			
edisi	character varying			
file_name	character varying			
internal_code	character varying(64)			
description	text			
is_ebook	boolean			
create_date	timestamp without time zone			
write_date	timestamp without time zone			

Gambar 3.28 Tabel Basis Data Ketersediaan Buku Perpustakaan

Tabel "judul_buku" dalam database ini menyimpan data terkait informasi buku, seperti kategori buku (kategori_buku), bahasa (bahasa_id), jumlah halaman (jumlah_halaman), dan stok buku yang terdiri dari jumlah total (jumlah_buku), jumlah yang dipinjam (jumlah_pinjam), serta jumlah yang tersedia (jumlah_tersedia). Kolom name mencatat judul buku, sedangkan isbn, edisi, dan file_name menyimpan informasi ISBN, edisi buku, dan nama file buku. Kolom internal_code dan description digunakan untuk kode internal serta deskripsi buku. Buku yang berupa ebook ditandai pada kolom boolean is_ebook. Data terkait pembuatan dan perubahan tercatat pada kolom create_uid, write_uid, create_date, dan write_date.

Table "public.pinjam_buku"

Column	Type	Collation	Nullable	Default
id	integer		not null	nextval('pinjam_buku_id_seq'::regclass)
judul_buku	integer			
nama_buku	integer			
siswa_id	integer			
pegawai_sekolah	integer			
kartu_perpustakaan	integer			
pinjam_id	integer			
penalty_days	integer			
create_uid	integer			
write_uid	integer			
type	character varying			
state	character varying			
issued_date	date			
return_date	date			
due_date	date			
tanggal_pengembalian	date			
create_date	timestamp without time zone			
write_date	timestamp without time zone			

Gambar 3.29 Tabel Basis Data Peminjaman Buku Perpustakaan

Tabel "pinjam_buku" dalam database ini menyimpan data terkait peminjaman buku di perpustakaan, dengan kolom-kolom yang mencakup identitas buku yang dipinjam (judul_buku, nama_buku), siswa peminjam (siswa_id), dan pegawai yang mengelola peminjaman (pegawai_sekolah). Informasi tambahan mencakup nomor kartu perpustakaan (kartu_perpustakaan), ID peminjaman (pinjam_id), serta jumlah hari denda (penalty_days) jika terdapat keterlambatan. Status peminjaman diatur melalui kolom type dan state, sementara kolom tanggal mencatat rincian waktu peminjaman, pengembalian, serta batas akhir pengembalian, seperti issued_date, return_date, due_date, dan tanggal_pengembalian. Informasi pembuatan dan perubahan data tercatat pada kolom create_uid, write_uid, create_date, dan write_date.

Table "public.kehadiran"

Column	Type	Collation	Nullable	Default
id	integer		not null	nextval('kehadiran_id_seq'::regclass)
akademik_id	integer			
color	integer			
walikelas	integer			
create_uid	integer			
write_uid	integer			
name	character varying			
state	character varying			
type	character varying			
days	character varying			
date_start	date			
start_datetime	date		not null	
end_datetime	timestamp without time zone			
create_date	timestamp without time zone			
write_date	timestamp without time zone			

Gambar 3.30 Tabel Basis Data Kehadiran

Tabel "kehadiran" dalam database ini menyimpan data terkait kehadiran akademik, yang mencakup informasi seperti ID akademik (akademik_id), warna label

untuk penanda kehadiran (color), dan wali kelas (walikelas). Kolom name mencatat nama aktivitas atau acara kehadiran, sementara state dan type merepresentasikan status dan tipe kehadiran. Kolom days menunjukkan jumlah hari kehadiran, dan ada juga data waktu spesifik seperti tanggal mulai (date_start), waktu mulai (start_datetime), dan waktu selesai (end_datetime). Kolom create_uid, write_uid, create_date, dan write_date mencatat informasi tentang siapa yang membuat dan mengubah data, serta waktu pencatatan dan perubahan data tersebut.

Table "public.jadwal"

Column	Type	Collation	Nullable	Default
id	integer		not null	nextval('jadwal_id_seq'::regclass)
kelas_id	integer			
walikelas	integer			
pelajaran_id	integer			
materi_id	integer			
guru	integer			
jam_ke_id	integer			
color	integer			
create_uid	integer			
write_uid	integer			
event_type	character varying		not null	
name	character varying			
kode	character varying			
type	character varying			
state	character varying		not null	
days	character varying			
tanggal	date			
date_start	date			
materi	text			
deskripsi	text			
start_datetime	timestamp without time zone			
end_datetime	timestamp without time zone			
create_date	timestamp without time zone			
write_date	timestamp without time zone			

Gambar 3.31 Tabel Basis Data Jadwal

Tabel "jadwal" dalam database ini menyimpan data terkait jadwal kegiatan atau pelajaran. Kolom utama mencakup informasi kelas (kelas_id), wali kelas (walikelas), mata pelajaran (pelajaran_id), materi (materi_id), dan guru (guru). Kolom jam_ke_id mencatat sesi jam pelajaran, sedangkan color digunakan untuk penanda warna. Kolom event_type, name, kode, type, dan state merepresentasikan jenis acara, nama kegiatan, kode unik, tipe kegiatan, dan status kegiatan. Tanggal dan waktu kegiatan dicatat dalam kolom tanggal, date_start, start_datetime, dan end_datetime. Deskripsi detail materi tersimpan di kolom materi dan deskripsi. Informasi terkait pencatatan dan perubahan data tersimpan pada kolom create_uid, write_uid, create_date, dan write_date.

Table "public.jemputan"

Column	Type	Collation	Nullable	Default
id	integer		not null	nextval('jemputan_id_seq'::regclass)
siswa_id	integer			
area_id	integer		not null	
pemilik_id	integer			
tipe	integer			
kendaraan	integer			
period_id	integer			
akademik_id	integer			
akademik_tingkat	integer			
walikelas	integer			
create_uid	integer			
write_uid	integer			
alamat	character varying		not null	
selected_days_display	character varying			
plat_nomor	character varying			
state	character varying			
create_date	timestamp without time zone			
write_date	timestamp without time zone			
base_price	double precision		not null	

Gambar 3.32 Tabel Basis Data Jemputan

Tabel "jemputan" dalam database ini menyimpan data terkait layanan penjemputan untuk siswa, dengan kolom-kolom yang mencakup identitas siswa (siswa_id), area penjemputan (area_id), dan pemilik layanan (pemilik_id). Kolom tipe dan kendaraan mencatat jenis dan kendaraan yang digunakan, sementara period_id dan akademik_id menunjukkan periode dan tahun akademik. Kolom walikelas menyimpan informasi wali kelas terkait. Informasi tambahan meliputi alamat (alamat), hari-hari penjemputan yang dipilih (selected_days_display), nomor plat kendaraan (plat_nomor), serta status layanan (state). Harga dasar layanan dicatat pada kolom base_price. Data pembuatan dan perubahan disimpan pada kolom create_uid, write_uid, create_date, dan write_date.

Table "public.tagihan_siswa"

Column	Type	Collation	Nullable	Default
id	integer		not null	nextval('tagihan_siswa_id_seq'::regclass)
struktur_biaya_id	integer			
siswa_id	integer			
akademik_id	integer			
akademik_tingkat	integer			
journal_id	integer			
batch_id	integer			
create_uid	integer			
write_uid	integer			
number	character varying			
name	character varying			
sequence_name	character varying			
name_tagihan	character varying			
state	character varying			
payment_state	character varying			
date	date			
create_date	timestamp without time zone			
write_date	timestamp without time zone			

Gambar 3.33 Tabel Basis Data Tagihan Siswa

Tabel "tagihan_siswa" dalam database ini menyimpan data terkait tagihan yang harus dibayar oleh siswa. Kolom-kolom utama mencakup informasi struktur biaya

(struktur_biaya_id), identitas siswa (siswa_id), tahun akademik (akademik_id dan akademik_tingkat), serta jurnal dan batch pembayaran (journal_id dan batch_id). Kolom number, name, sequence_name, dan name_tagihan digunakan untuk menyimpan informasi nomor tagihan, nama tagihan, urutan tagihan, dan nama spesifik tagihan. Status tagihan dicatat dalam kolom state, sedangkan status pembayaran disimpan dalam kolom payment_state. Tanggal tagihan ditentukan pada kolom date, sementara informasi pencatatan dan perubahan data tersimpan di kolom create_uid, write_uid, create_date, dan write_date.

Table "public.psb"

Column	Type	Collation	Nullable	Default
id	integer		not null	nextval('psb_id_seq'::regclass)
color	integer			
user_id	integer			
age_months	integer			
state_id	integer			
country_id	integer			
sekolah	integer			
psb_id	integer		not null	
akademik_tingkat	integer			
suku_id	integer			
struktur_biaya_id	integer			
stage_id	integer			
create_uid	integer			
write_uid	integer			
nama_panggilan	character varying			
gender	character varying			
kanban_state	character varying		not null	
sequence	character varying			
priority	character varying			
place_of_birth	character varying			
age_years	character varying			
street	character varying(256)			
street2	character varying(256)			
phone	character varying			
mobile	character varying(16)			
email	character varying			
city	character varying(64)			
zip	character varying(8)			
asal_sekolah	character varying			
religion	character varying			
kewarganegaraan	character varying			
status	character varying			
kondisi	character varying			
sumbangan_wajib	character varying			
sumbangan_skl	character varying			
blood_group	character varying			
admission_date	date			
date_of_birth	date		not null	
name	jsonb			
riwayat_penyakit	text			
is_siswa	boolean			
create_date	timestamp without time zone			
write_date	timestamp without time zone			
fees	double precision			
height	double precision			
weight	double precision			

Gambar 3.34 Tabel Basis Data Data Siswa

Tabel "psb" dalam database ini menyimpan data terkait pendaftaran siswa baru, dengan berbagai kolom yang mencakup informasi pribadi dan akademik siswa. Kolom-kolom utama meliputi identitas pengguna (user_id), usia dalam bulan

(age_months), asal sekolah (sekolah), dan tingkat akademik (akademik_tingkat). Kolom nama_panggilan, gender, dan place_of_birth mencatat nama panggilan, jenis kelamin, dan tempat lahir siswa. Alamat lengkap siswa dicatat dalam kolom street, street2, city, dan zip, serta kontak pada kolom phone, mobile, dan email. Informasi lainnya termasuk kewarganegaraan (kewarganegaraan), kondisi (kondisi), golongan darah (blood_group), dan riwayat penyakit (riwayat_penyakit). Status pendaftaran dan data akademik lainnya, seperti sumbangan wajib dan sumbangan sekolah, dicatat dalam kolom kanban_state, priority, sumbangan_wajib, dan sumbangan_skl. Tanggal pendaftaran (admission_date), tanggal lahir (date_of_birth), dan data fisik (berat dan tinggi) disimpan dalam kolom fees, height, dan weight. Kolom tambahan seperti create_uid, write_uid, create_date, dan write_date mencatat informasi pencatatan dan perubahan data.

Table "public.berita"

Column	Type	Collation	Nullable	Default
id	integer		not null	nextval('berita_id_seq'::regclass)
create_uid	integer			
write_uid	integer			
name	character varying			
start_date	date			
end_date	date			
deskripsi	text			
create_date	timestamp without time zone			
write_date	timestamp without time zone			

Gambar 3.35 Tabel Basis Data Data Berita

Tabel "berita" dalam database ini menyimpan data terkait informasi berita. Kolom-kolom utama mencakup ID berita (id), nama atau judul berita (name), deskripsi (deskripsi), serta periode berlakunya berita yang ditandai dengan kolom tanggal mulai (start_date) dan tanggal berakhir (end_date). Kolom create_uid dan write_uid mencatat ID pengguna yang membuat dan memperbarui data, sementara kolom create_date dan write_date menyimpan informasi waktu pencatatan dan perubahan data.

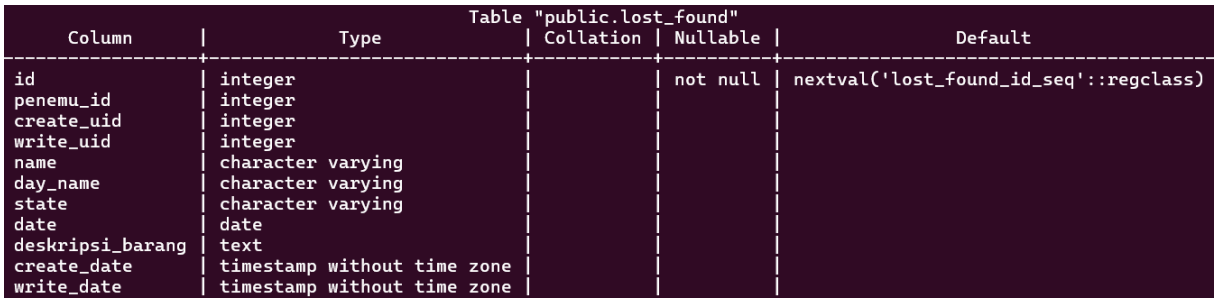


Table "public.lost_found"

Column	Type	Collation	Nullable	Default
id	integer		not null	nextval('lost_found_id_seq'::regclass)
penemu_id	integer			
create_uid	integer			
write_uid	integer			
name	character varying			
day_name	character varying			
state	character varying			
date	date			
deskripsi_barang	text			
create_date	timestamp without time zone			
write_date	timestamp without time zone			

Gambar 3.36 Tabel Basis Data Barang Hilang

Tabel "lost_found" dalam database ini menyimpan data terkait barang hilang dan ditemukan. Kolom-kolom utama mencakup ID barang (id), ID penemu barang (penemu_id), nama barang (name), hari ditemukannya barang (day_name), dan tanggal barang ditemukan (date). Status barang dicatat dalam kolom state, sementara deskripsi barang disimpan pada kolom deskripsi_barang. Kolom create_uid dan write_uid mencatat ID pengguna yang membuat dan memperbarui data, sedangkan kolom create_date dan write_date menyimpan informasi waktu pencatatan dan perubahan data tersebut.

3.2.6 Implementasi

Setelah tahap perancangan sistem selesai, proses dilanjutkan ke fase realisasi. Pada fase ini, desain konseptual atau kerangka kerja yang telah disusun akan dikembangkan menjadi produk yang dapat digunakan. Dalam tahap realisasi, praktikan mencakup pengembangan database, perealisasian wireframe, serta penentuan spesifikasi untuk perangkat keras dan perangkat lunak yang dibutuhkan.

3.2.6.1 Spesifikasi Kebutuhan Perangkat Lunak

Dalam proses implementasi, praktikan menggunakan beberapa perangkat lunak untuk memastikan bahwa sistem dapat dikembangkan dan berfungsi dengan baik. Adapun perangkat lunak yang digunakan adalah.

Tabel 3.2 Spesifikasi Kebutuhan Perangkat Lunak Mobile

No	Detail Perangkat Lunak	Kebutuhan Perangkat Lunak
1	*Operating System*	Windows 11
2	*Local Environment*	Node.js
3	*Front-End Framework (Mobile)*	React Native
4	*Backend*	ExpressJS (JavaScript)
5	*Database*	PostgreSQL (terkoneksi dengan Odoo)
6	*IDE*	Visual Studio Code
7	*Scripting Language*	JavaScript
8	*Version Control*	GitLab
9	*Emulator/Simulator*	ExpoGo
10	*CSS Styling*	CSS
11	*HTTP Client Library*	Axios

3.2.6.2Spesifikasi Kebutuhan Perangkat Keras

Dalam proses implementasi, praktikan menggunakan beberapa perangkat keras untuk memastikan bahwa sistem dapat dikembangkan dan berfungsi dengan baik serta sesuai dengan tujuannya. Adapun perangkat keras yang digunakan adalah.

Tabel 3.3 Spesifikasi Kebutuhan Perangkat Keras Mobile

No	Nama Perangkat Keras	Spesifikasi Perangkat Keras
1	Processor	AMD Ryzen 7 4700
2	VGA	Integrated AMD Radeon Graphics
3	Memory	SSD 512 GB
4	RAM	16 GB DDR4
5	Layar	15.6" FHD (1920x1080)
6	Baterai	45Wh, hingga 6 jam penggunaan

3.6.2.3Implementasi

Dalam tahap implementasi, praktikan menggunakan metode Model-View-ViewModel (MVVM) sebagai arsitektur utama. Metode MVVM merupakan pengembangand dari metode *Presentation Model* (PM) yang digunakan dalam mengkaji tampilan UI dan Logic (Fajri, 2022). Pendekatan MVVM dipilih karena kemampuannya dalam memisahkan logika bisnis dari antarmuka pengguna, sehingga setiap komponen memiliki tanggung jawabnya masing-masing, yang pada akhirnya dapat meningkatkan fleksibilitas aplikasi (Arif et al., 2019).

Pada bagian Model, sistem dirancang untuk menyimpan dan mengelola data serta logika bisnis utama. Model bertanggung jawab atas interaksi langsung dengan API Odoo untuk mendapatkan data yang relevan dari server, seperti data pengguna, produk, atau transaksi, dan menyimpan data tersebut secara efisien. Model juga memastikan bahwa data yang diterima dan dikirim ke server memenuhi kebutuhan aplikasi serta mematuhi logika bisnis yang diterapkan. Dengan mengisolasi logika bisnis dan pengelolaan data di dalam Model, aplikasi menjadi lebih mudah diperbarui jika ada perubahan pada logika atau struktur data.

ViewModel berperan sebagai jembatan penghubung antara Model dan View. Di sini, ViewModel mengelola alur data dari Model ke View dengan memastikan bahwa data yang diperlukan oleh antarmuka pengguna selalu up-to-date. ViewModel juga mengatur status tampilan aplikasi, seperti status loading, error, atau berhasil, sehingga tampilan aplikasi dapat berubah secara dinamis sesuai kondisi data.

View, yang terdiri dari komponen-komponen React Native, bertanggung jawab untuk merender antarmuka dan berinteraksi langsung dengan pengguna. View mengambil data dari ViewModel dan menampilkannya kepada pengguna dalam bentuk yang mudah dipahami. Berikut merupakan tampilan implementasi sistem ini.

Gambar 3.37 Kode Program & Implementasi Login

```
import { UserContext } from "./UserContext";

export default function LoginScreen() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const navigation = useNavigation();
  const { setUser } = useContext(UserContext);

  const handleLogin = async () => {
    try {
      const response = await axios.post(
        "http://18.136.196.171:3031/api/odoo/login",
        {
          email,
          password,
        }
      );

      if (response.data.success) {
        const uid = response.data.uid;
        const userDataResponse = await axios.post(
          "http://18.136.196.171:3031/api/odoo/getUserData",
          {
            email,
            uid,
            password,
          }
        );

        if (userDataResponse.data.error) {
          Alert.alert("Login Failed", userDataResponse.data.error);
        } else {
          const userData = userDataResponse.data;

          console.log("User Data:", userData); // Debugging

          // Set user data in context
          setUser({
            uid,
            password,
            email,
            name: userData.name,
            image: userData.image_128,
            alamat: userData.alamat,
            place_of_birth: userData.place_of_birth,
            date_of_birth: userData.date_of_birth,
            bahasa: userData.bahasa,
            gender: userData.gender,
            age: userData.age,
            kewarganegaraan: userData.kewarganegaraan,
            height: userData.height,
            weight: userData.weight,
```

(a) Kode Program

(b) Implementasi Login

Implementasi halaman login pada aplikasi Sistem Informasi Sekolah ini menampilkan logo, judul "Login," dan sambutan singkat. Pengguna dapat memasukkan email dan password, dengan opsi "Forgot password?" jika perlu mengatur ulang kata sandi. Tombol "SIGN IN" di bagian bawah digunakan untuk mengautentikasi pengguna dan mengarahkan mereka ke halaman utama setelah login berhasil. Source code pada gambar mengimplementasikan modul Login untuk aplikasi menggunakan API. Fungsi handleLogin memproses autentikasi dengan mengirimkan email dan password ke API login. Jika berhasil, sistem memanggil API tambahan untuk mendapatkan data pengguna, seperti nama, alamat, tanggal lahir, dan informasi pribadi lainnya. Data pengguna yang diterima kemudian disimpan dalam context untuk digunakan di aplikasi.

Gambar 3.38 Kode Program & Implementasi Homescreen

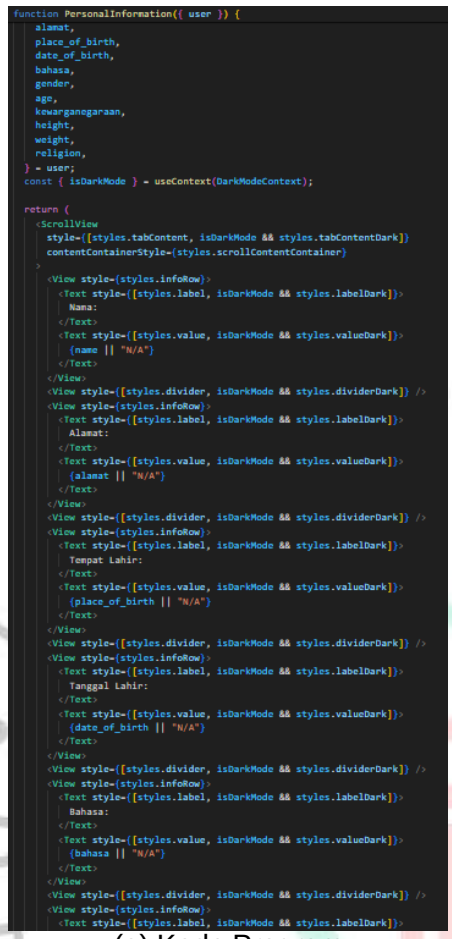
```
const libraryItems = [
  {
    name: "Perpustakaan",
    image: require("./assets/perpus_new.png"),
    screen: "Library",
  },
  {
    name: "Jemputan",
    image: require("./assets/jemputan_new.png"),
    screen: "JemputanScreen",
  },
  {
    name: "Penilaian",
    image: require("./assets/penilaian_new.png"),
    screen: "Evaluation",
  },
  {
    name: "Asrama",
    image: require("./assets/asrama_new.png"),
    screen: "AsramaScreen",
  },
  {
    name: "Kehadiran",
    image: require("./assets/kehadiran_new.png"),
    screen: "KehadiranScreen",
  },
  {
    name: "Biaya",
    image: require("./assets/biaya_new.png"),
    screen: "TagihanScreen",
  },
  {
    name: "Jadwal",
    image: require("./assets/jadwal_new.png"),
    screen: "DaftarPelajaranScreen",
  },
  {
    name: "UKS",
    image: require("./assets/uks_new.png"),
    screen: "UKS",
  },
];

export default function HomeScreen() {
  const { user } = useContext(UserContext);
  const navigation = useNavigation();
  const [berita, setBerita] = useState([]);
  const [barangHilang, setBarangHilang] = useState([]);
  const [loading, setLoading] = useState(true);
  const [refreshing, setRefreshing] = useState(false);
  const [currentScreen, setCurrentScreen] = useState(0);
  const [isFullScreen, setIsFullScreen] = useState(false);
  const [videoUrl, setVideoUrl] = useState("");
  const [attendancePercentage, setAttendancePercentage] = useState(null);
```

(a) Kode Program

(b) Implementasi Homescreen

Tiga kotak statistik utama menampilkan informasi ringkas tentang kehadiran, status tugas, dan semester aktif, memungkinkan pengguna memantau status akademik secara cepat. Di bawahnya, terdapat ikon-ikon modul utama seperti Perpustakaan, Jemputan, Penilaian, Asrama, Kehadiran, Biaya, Jadwal, dan UKS, serta modul Barang Hilang, yang memudahkan pengguna untuk mengakses fitur-fitur utama aplikasi. Bagian Informasi di bagian bawah menampilkan berita atau pengumuman terbaru dengan opsi "Lihat Semua" untuk informasi lebih lengkap. Source code pada gambar mengimplementasikan modul HomeScreen untuk aplikasi mobile dengan daftar menu utama. Data menu, seperti "Perpustakaan," "Jemputan," "Penilaian," "Asrama," dan lainnya, disimpan dalam array libraryItems, yang berisi nama, ikon, dan tujuan navigasi masing-masing. Fungsi HomeScreen menggunakan data ini untuk menampilkan ikon menu secara dinamis pada UI.

Gambar 3.39 Kode Program & Implementasi Setting



```
function PersonalInformation({ user }) {
  alamat,
  place_of_birth,
  date_of_birth,
  bahasa,
  gender,
  age,
  kewarganegaraan,
  height,
  weight,
  religion,
} = user;
const { isDarkMode } = useContext(DarkModeContext);

return (
  <ScrollView
    style={[styles.tabContent, isDarkMode && styles.tabContentDark]}
    contentContainerStyle={styles.scrollContentContainer}
  >
    <View style={styles.infoRow}>
      <Text style={[styles.label, isDarkMode && styles.labelDark]}>
        Nama:
      </Text>
      <Text style={[styles.value, isDarkMode && styles.valueDark]}>
        {name || "N/A"}
      </Text>
    </View>
    <View style={[styles.divider, isDarkMode && styles.dividerDark]} />
    <View style={styles.infoRow}>
      <Text style={[styles.label, isDarkMode && styles.labelDark]}>
        Alamat:
      </Text>
      <Text style={[styles.value, isDarkMode && styles.valueDark]}>
        {alamat || "N/A"}
      </Text>
    </View>
    <View style={[styles.divider, isDarkMode && styles.dividerDark]} />
    <View style={styles.infoRow}>
      <Text style={[styles.label, isDarkMode && styles.labelDark]}>
        Tempat Lahir:
      </Text>
      <Text style={[styles.value, isDarkMode && styles.valueDark]}>
        {place_of_birth || "N/A"}
      </Text>
    </View>
    <View style={[styles.divider, isDarkMode && styles.dividerDark]} />
    <View style={styles.infoRow}>
      <Text style={[styles.label, isDarkMode && styles.labelDark]}>
        Tanggal Lahir:
      </Text>
      <Text style={[styles.value, isDarkMode && styles.valueDark]}>
        {date_of_birth || "N/A"}
      </Text>
    </View>
    <View style={[styles.divider, isDarkMode && styles.dividerDark]} />
    <View style={styles.infoRow}>
      <Text style={[styles.label, isDarkMode && styles.labelDark]}>
        Bahasa:
      </Text>
      <Text style={[styles.value, isDarkMode && styles.valueDark]}>
        {bahasa || "N/A"}
      </Text>
    </View>
    <View style={[styles.divider, isDarkMode && styles.dividerDark]} />
    <View style={styles.infoRow}>
      <Text style={[styles.label, isDarkMode && styles.labelDark]}>
```

(a) Kode Program

(b) Implementasi Settings

Implementasi halaman Setting pada aplikasi Sistem Informasi Sekolah ini memberikan tampilan yang lebih kaya dan interaktif dibandingkan wireframe sebelumnya. Di bagian atas, terdapat fitur Dark Mode untuk mengubah tema tampilan dan tombol Keluar untuk logout. Foto profil pengguna, disertai dengan nama pengguna (Alifia), ditampilkan untuk memberikan sentuhan personal pada halaman ini. Halaman Setting ini dibagi menjadi beberapa tab, yaitu Personal Information, School, dan Other Information, yang memungkinkan pengguna untuk dengan mudah mengakses informasi yang relevan. Pada tab Personal Information, ditampilkan detail pengguna seperti Nama, Alamat, Tempat Lahir, Tanggal Lahir, dan Bahasa. Source code pada gambar menampilkan komponen PersonalInformationScreen untuk menampilkan informasi pribadi pengguna secara dinamis.

Gambar 3.40 Kode Program & Implementasi Modul Perpustakaan

```
const libraryItems = [
  {
    title: "Ketersediaan Buku",
    description: "Daftar buku yang tersedia.",
    image: require("./assets/book4.png"),
    screen: "KetersediaanBuku",
  },
  {
    title: "Daftar Pinjam Buku",
    description: "Daftar peminjaman buku saya.",
    image: require("./assets/book2.png"),
    screen: "DaftarPinjamBuku",
  },
  {
    title: "Riwayat Berkunjung",
    description: "Jadwal berkunjung ke perpustakaan.",
    image: require("./assets/book3.png"),
    screen: "RiwayatBerkunjung",
  },
  {
    title: "Electronic Book (E-Book)",
    description: "Lihat Buku Secara Online.",
    image: require("./assets/ebook.png"),
    screen: "EbookScreen",
  },
  {
    title: "Aturan Perpustakaan",
    description: "Aturan peminjaman dan pengembalian",
    image: require("./assets/book5.png"),
    screen: "AturanPerpustakaan",
  },
];

export default function LibraryScreen() {
  const navigation = useNavigation();
  const { user } = useContext(UserContext); // Get user from UserContext

  // Destructure necessary values from user object
  const { uid, email, password } = user || {};

  return (
    <View style={styles.container}>
      <View style={styles.header}>
        <TouchableOpacity onPress={() => navigation.goBack()}>
          <Text style={styles.backButton}>{"<"}</Text>
        </TouchableOpacity>
        <Text style={styles.headerText}>Perpustakaan</Text>
      </View>
      <ScrollView>
        {libraryItems.map((item, index) => (
          <TouchableOpacity
            key={index}
            style={styles.card}
            onPress={() =>
              navigation.navigate(item.screen, { uid, email, password })
            }
          >
            <Image source={item.image} style={styles.cardImage} />
            <View style={styles.cardContent}>
              <Text style={styles.cardTitle}>{item.title}</Text>
              <Text style={styles.cardDescription}>{item.description}</Text>
            </View>
          </TouchableOpacity>
        ))}
      </ScrollView>
    </View>
```

a) Kode Program

(b) Implementasi Modul Perpustakaan

Berikut implementasi halaman Modul Perpustakaan pada aplikasi Sistem Informasi Sekolah. Terdapat beberapa menu utama dengan ikon dan deskripsi singkat untuk setiap fungsi, yaitu Ketersediaan Buku untuk melihat daftar buku yang tersedia, Daftar Pinjam Buku untuk mengakses daftar buku yang sedang dipinjam oleh pengguna, Riwayat Berkunjung untuk melihat jadwal kunjungan perpustakaan, dan Aturan Perpustakaan yang memberikan informasi mengenai aturan peminjaman dan pengembalian buku. Setiap menu dirancang dengan tampilan yang jelas dan ikon yang intuitif, memudahkan pengguna dalam menavigasi modul perpustakaan. Source code pada gambar mengimplementasikan modul LibraryScreen untuk menampilkan menu fitur perpustakaan seperti "Ketersediaan Buku" dan "Daftar Pinjam Buku." Data menu disimpan dalam array libraryItems, yang berisi informasi judul, deskripsi, ikon, dan layar tujuan. Komponen menggunakan ScrollView untuk menampilkan daftar menu secara dinamis dengan memetakan setiap item menjadi kartu interaktif.

Gambar 3.41 Kode Program & Implementasi Daftar Ketersediaan Buku

```
function BukuTersedia({ route }) {
  const { uid, password, email } = route.params;
  const [data, setData] = useState([]);
  const [loading, setLoading] = useState(true);
  const [search, setSearch] = useState("");
  const navigation = useNavigation();

  useEffect(() => {
    if (email && password) {
      fetchData();
    } else {
      console.error("Email or password is missing.");
      setLoading(false);
    }
  }, []);

  const fetchData = async () => {
    try {
      const authResponse = await axios.post(
        "http://18.136.196.171:3031/api/odoo/login",
        { email, password }
      );
      const uid = authResponse.data.uid;

      const booksResponse = await axios.post(
        "http://18.136.196.171:3031/api/odoo/getAvailableBooks",
        { uid, password, email }
      );
      setData(booksResponse.data);
      setLoading(false);
    } catch (error) {
      console.error("Error fetching data:", error);
      setLoading(false);
    }
  };

  const filteredData = data.filter((item) =>
    item.name.toLowerCase().includes(search.toLowerCase())
  );

  return (
    <SafeAreaView style={styles.container}>
      <TextInput
        style={styles.searchBar}
        placeholder="Cari Buku..."
        value={search}
        onChangeText={(text) => setSearch(text)}
      />
      <View style={styles.tableHeader}>
        <Text style={styles.headerCell}>Judul Buku</Text>
        <Text style={styles.headerCell}>Jumlah Buku</Text>
        <Text style={styles.headerCell}>Jumlah Tersedia</Text>
      </View>
      {loading ? (
        <ActivityIndicator size="large" color="#007bff" />
      ) : (
        <ScrollView>
          {filteredData.map((item, index) => (
            <View key={index} style={styles.row}>
              <TouchableOpacity
                onPress={() =>
                  navigation.navigate("KetersediaanBukuDetail", { book: item })
                }
                style={styles.cell}
              >
                <Text style={styles.cellTextCore}>{item.name}</Text>
```

(a) Kode Program

(b) Implementasi Ketersediaan Buku

Implementasi halaman Ketersediaan Buku pada aplikasi Sistem Informasi Sekolah ini menyediakan informasi yang lengkap dan mudah diakses mengenai stok buku di perpustakaan. Bagian atas halaman dilengkapi dengan tab "BUKU TERSEDIA" dan "BUKU TIDAK TERSEDIA" yang memungkinkan pengguna untuk memfilter buku berdasarkan ketersediaannya. Fitur pencarian memudahkan pengguna untuk menemukan buku tertentu berdasarkan judul. Source code pada gambar menunjukkan implementasi modul BookAvailableScreen. Komponen ini menggunakan fungsi useEffect untuk menjalankan fetchData saat pertama kali dimuat, di mana API dipanggil menggunakan axios.post untuk autentikasi dan pengambilan data buku. Data yang diterima disimpan dalam state data dan difilter menggunakan fungsi filteredData berdasarkan teks input pencarian pengguna, yang dipantau melalui state search. Untuk menampilkan data, digunakan elemen ScrollView yang diisi dengan daftar buku menggunakan metode map.

Gambar 3.42 Kode Program & Implementasi Detail Ketersediaan Buku

```
export default function BookDetailScreen({ route }) {
  const { book } = route.params;
  const navigation = useNavigation();

  return (
    <SafeAreaView style={styles.container}>
      <View style={styles.header}>
        <TouchableOpacity onPress={() => navigation.goBack()}>
          <Text style={styles.backButton}>{"<"}</Text>
        </TouchableOpacity>
        <Text style={styles.headerText}>Detail Peminjaman</Text>
      </View>

      {/* Bungkus dengan ScrollView */}
      <ScrollView contentContainerStyle={styles.scrollViewContent}>
        <View style={styles.detailContainer}>
          <Text style={styles.sectionHeader}>Informasi Buku</Text>

          <View style={styles.row}>
            <Text style={styles.label}>Judul Buku:</Text>
            <Text style={styles.value}>{book.judul_buku[1]}</Text>
          </View>

          <View style={styles.row}>
            <Text style={styles.label}>Nama Buku:</Text>
            <Text style={styles.value}>{book.nama_buku}</Text>
          </View>

          <View style={styles.row}>
            <Text style={styles.label}>Dipinjam:</Text>
            <Text style={styles.value}>{book.issued_date}</Text>
          </View>

          <View style={styles.row}>
            <Text style={styles.label}>Dikembalikan:</Text>
            <Text style={styles.value}>{book.return_date}</Text>
          </View>

          <View style={styles.row}>
            <Text style={styles.label}>Status:</Text>
            <Text style={styles.value}>{book.state}</Text>
          </View>

          <View style={styles.divider}></View>

          <Text style={styles.sectionHeader2}>Penalty</Text>

          <View style={styles.row}>
            <Text style={styles.label}>Lewat Batas:</Text>
            <Text style={styles.value}>{book.lewat_batas}</Text>
          </View>

          <View style={styles.row}>
            <Text style={styles.label}>Penalty:</Text>
            <Text style={styles.value}>{book.penalty}</Text>
          </View>
        </View>
      </ScrollView>
    </SafeAreaView>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#fff",
```

(a) Kode Program

(b) Implementasi Detail Ketersediaan Buku

Implementasi halaman Detail Ketersediaan Buku pada aplikasi Sistem Informasi Sekolah ini memberikan informasi mendetail mengenai buku yang dipilih dari daftar Ketersediaan Buku. Setiap elemen informasi, seperti Judul Buku, Penulis, ISBN Code, Internal Code, Bahasa, dan Jumlah Halaman, ditampilkan dengan jelas untuk memudahkan pengguna mendapatkan informasi lengkap. Data ini diambil secara real-time dari database Odoo, memastikan bahwa setiap detail yang ditampilkan adalah informasi terkini yang tersimpan dalam sistem. Selain itu, terdapat bagian Penerbit yang menampilkan detail penerbit buku, menambah kelengkapan informasi bagi pengguna. Source code pada gambar mengimplementasikan modul BookDetailsScreen untuk menampilkan informasi detail buku. Data buku, seperti judul, penulis, ISBN, kode internal, bahasa, jumlah halaman, penerbit, dan informasi penalti, diterima melalui parameter route.params dan ditampilkan secara terstruktur dalam UI. Elemen ScrollView digunakan untuk memastikan seluruh informasi dapat di-scroll jika konten melebihi tinggi layar.

Gambar 3.43 Kode Program & Implementasi Informasi Sekolah

```
export default function DetailBarangHilangScreen({ route }) {
  const { name, date, day_name, penemu_id, deskripsi_image, deskripsi_barang } =
    route.params;
  const navigation = useNavigation();

  return (
    <View style={styles.container}>
      <View style={styles.header}>
        <TouchableOpacity onPress={() => navigation.goBack()}>
          <Text style={styles.backButton}>{"<"}</Text>
        </TouchableOpacity>
        <Text style={styles.headerText}>Detail Barang</Text>
      </View>
      <ScrollView contentContainerStyle={styles.scrollContainer}>
        {deskripsi_image ? (
          <Image
            style={styles.image}
            source={{ uri: deskripsi_image }} // Use directly as it's already in base64 format
          />
        ) : (
          <View style={styles.imagePlaceholder} />
        )}
        <View style={styles.detailsContainer}>
          <Text style={styles.label}>Nama Barang:</Text>
          <Text style={styles.value}>{name}</Text>

          <Text style={styles.label}>Tanggal Ditemukan:</Text>
          <Text style={styles.value}>{date}</Text>

          <Text style={styles.label}>Hari Ditemukan:</Text>
          <Text style={styles.value}>{day_name}</Text>

          <Text style={styles.label}>Penemu:</Text>
          <Text style={styles.value}>{penemu_id}</Text>

          <Text style={styles.label}>Deskripsi:</Text>
          <HTMLView value={deskripsi_barang} stylesheet={styles.description} />
        </View>
      </ScrollView>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#fff",
  },
  header: {
    flexDirection: "row",
    alignItems: "center",
    backgroundColor: "#445DA5",
    padding: 10,
  },
  backButton: {
    color: "white",
    fontSize: 25,
    marginRight: 5,
    marginLeft: 10,
    marginTop: 30,
  },
  headerText: {
    color: "white",
    fontSize: 18,
    textAlign: "center",
    flex: 1,
    marginTop: 25,
```

(a) Kode Program

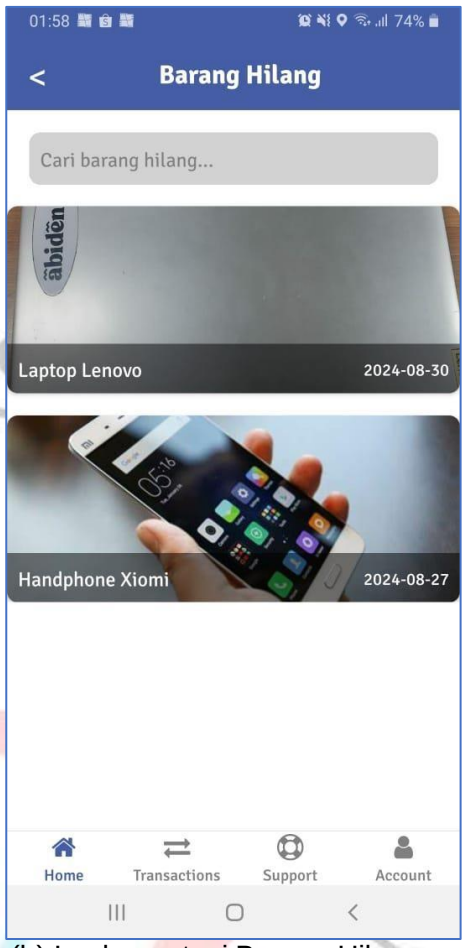
(b) Implementasi Informasi Sekolah

Implementasi halaman Seluruh Informasi pada aplikasi Sistem Informasi Sekolah ini menampilkan daftar informasi terbaru dari sekolah, seperti acara dan kegiatan, dalam format yang menarik secara visual. Di bagian atas halaman, terdapat fitur pencarian yang memungkinkan pengguna mencari berita atau informasi tertentu dengan cepat. Data ini diambil secara real-time dari database Odoo, memastikan bahwa konten yang disajikan selalu up-to-date dan relevan. Implementasi ini lebih kaya dibandingkan dengan wireframe awal karena menyajikan elemen visual penuh, integrasi gambar, serta pengelolaan data yang diambil langsung dari Odoo. Source code pada gambar mengimplementasikan modul untuk menampilkan detail informasi, seperti berita atau kegiatan tertentu, yang diterima melalui parameter route.params. Gambar ditampilkan menggunakan elemen <Image> dengan sumber yang diambil langsung dari data parameter. Selain itu, elemen TouchableOpacity digunakan untuk navigasi kembali ke halaman sebelumnya.

Gambar 3.44 Kode Program & Implementasi Barang Hilang

```
export default function DetailBarangHilangScreen({ route }) {
  const { name, date, day_name, penemu_id, deskripsi_image, deskripsi_barang } =
    route.params;
  const navigation = useNavigation();

  return (
    <View style={styles.container}>
      <View style={styles.header}>
        <TouchableOpacity onPress={() => navigation.goBack()}>
          <Text style={styles.backButton}>{"<"}</Text>
        </TouchableOpacity>
        <Text style={styles.headerText}>Detail Barang</Text>
      </View>
      <ScrollView contentContainerStyle={styles.scrollContainer}>
        {deskripsi_image ? (
          <Image
            style={styles.image}
            source={{ uri: deskripsi_image }} // Use directly as it's already in base64 format
          />
        ) : (
          <View style={styles.imagePlaceholder} />
        )}
        <View style={styles.detailsContainer}>
          <Text style={styles.label}>Nama Barang:</Text>
          <Text style={styles.value}>{name}</Text>

          <Text style={styles.label}>Tanggal Ditemukan:</Text>
          <Text style={styles.value}>{date}</Text>

          <Text style={styles.label}>Hari Ditemukan:</Text>
          <Text style={styles.value}>{day_name}</Text>

          <Text style={styles.label}>Penemu:</Text>
          <Text style={styles.value}>{penemu_id}</Text>

          <Text style={styles.label}>Deskripsi:</Text>
          <HTMLView value={deskripsi_barang} stylesheet={styles.description} />
        </View>
      </ScrollView>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#fff",
  },
  header: {
    flexDirection: "row",
    alignItems: "center",
    backgroundColor: "#445DA5",
    padding: 10,
  },
  backButton: {
    color: "white",
    fontSize: 25,
    marginRight: 5,
    marginLeft: 10,
    marginTop: 30,
  },
  headerText: {
    color: "white",
    fontSize: 18,
    textAlign: "center",
    flex: 1,
    marginTop: 25,
```

(a) Kode Program



(b) Implementasi Barang Hilang

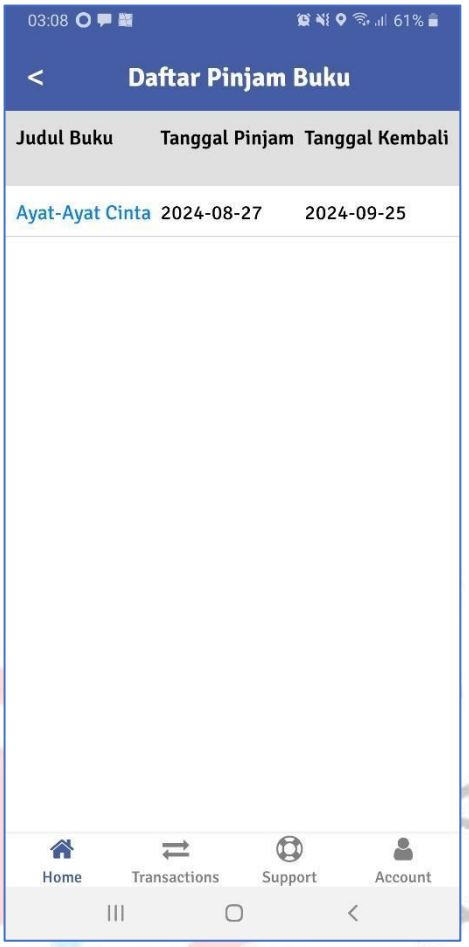
Implementasi halaman Barang Hilang pada aplikasi Sistem Informasi Sekolah ini menampilkan daftar barang hilang secara visual dengan gambar, nama barang, dan tanggal kehilangan, memudahkan pengguna untuk mengenali barang yang mungkin hilang. Di bagian atas terdapat kolom pencarian yang memungkinkan pengguna mencari barang hilang tertentu berdasarkan nama. Data ini diambil secara langsung dari database Odoo, memastikan bahwa informasi barang hilang yang ditampilkan selalu up-to-date. Implementasi ini berbeda dari wireframe awal karena menampilkan elemen visual penuh, termasuk gambar dan tanggal kehilangan, serta menyediakan fitur pencarian yang membantu pengguna menemukan informasi barang hilang dengan cepat. Source code pada gambar mengimplementasikan modul DetailBarangHilangScreen untuk menampilkan detail barang hilang berdasarkan data yang diterima melalui parameter route.params. Gambar barang hilang diambil dari properti deskripsi_image dan ditampilkan menggunakan elemen gambar

.

Gambar 3.45 Kode Program & Implementasi Daftar Pinjam Buku

```
export default function DaftarPinjamBukuScreen({ route }) {
  const navigation = useNavigation();

  useEffect(() => {
    if (email && password) {
      fetchData();
    } else {
      console.error("Email or password is missing.");
      setLoading(false);
    }
  }, []);

  const fetchData = async () => {
    try {
      console.log("Fetching data for email:", email);
      const authResponse = await axios.post(
        "http://18.136.196.171:3031/api/odoo/login",
        { email, password }
      );
      const uid = authResponse.data.uid;
      console.log("Authenticated, UID:", uid);

      const booksResponse = await axios.post(
        "http://18.136.196.171:3031/api/odoo/getBorrowedBooks",
        { uid, password, email }
      );
      setData(booksResponse.data);
      setLoading(false);
    } catch (error) {
      console.error("Error fetching data:", error);
      setLoading(false);
    }
  };

  const renderItem = ({ item }) => (
    <View style={styles.row}>
      <View style={styles.cellContainer}>
        <TouchableOpacity
          onPress={() => navigation.navigate("BookDetail", { book: item })}
        >
          <Text style={styles.cellTextCore}>{item.judul_buku[1]}</Text>
        </TouchableOpacity>
      </View>
      <View style={styles.cellContainer}>
        <Text style={styles.cellText}>{item.issued_date}</Text>
      </View>
      <View style={styles.cellContainer}>
        <Text style={styles.cellText}>{item.return_date}</Text>
      </View>
    </View>
  );

  if (loading) {
    return (
      <View style={styles.loader}>
        <ActivityIndicator size="large" color="#007bff" />
      </View>
    );
  }

  return (
    <View style={styles.container}>
      <View style={styles.header}>
        <TouchableOpacity onPress={() => navigation.goBack()}>
          <Text style={styles.backButton}>{"<"}</Text>
        </TouchableOpacity>
        <Text style={styles.headerText}>Daftar Pinjam Buku</Text>
```

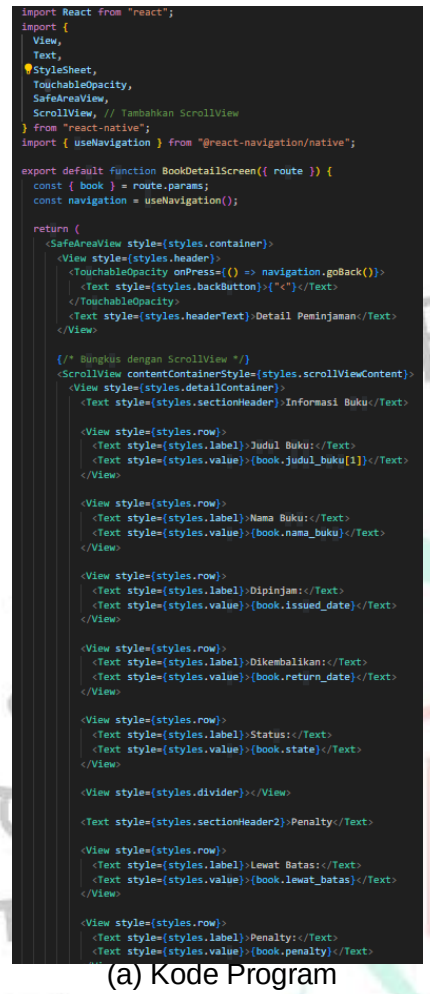
(a) Kode Program



(b) Implementasi Daftar PInjam Buku

Implementasi halaman Daftar Pinjam Buku pada aplikasi Sistem Informasi Sekolah ini menampilkan data peminjaman buku yang sedang aktif, mencakup Judul Buku, Tanggal Pinjam, dan Tanggal Kembali. Setiap baris mewakili satu transaksi peminjaman buku dengan tanggal pinjam dan tanggal kembali yang jelas, memudahkan pengguna untuk melacak buku yang dipinjam. Implementasi ini berbeda dari wireframe awal dengan adanya tampilan elemen visual yang lengkap dan konektivitas data langsung dari Odoo. Source code pada gambar mengimplementasikan modul DaftarPinjamBukuScreen untuk menampilkan daftar buku yang dipinjam oleh pengguna. Fungsi fetchData memanggil API melalui HTTP POST untuk autentikasi pengguna menggunakan email dan password, kemudian menggunakan UID yang diperoleh untuk mengambil data buku yang dipinjam melalui API lain.

Gambar 3.46 Kode Program & Implementasi Detail Peminjaman Buku



```
import React from "react";
import {
  View,
  Text,
  StyleSheet,
  TouchableOpacity,
  SafeAreaView,
  ScrollView, // Tambahkan ScrollView
} from "react-native";
import { useNavigation } from "@react-navigation/native";

export default function BookDetailScreen({ route }) {
  const { book } = route.params;
  const navigation = useNavigation();

  return (
    <SafeAreaView style={styles.container}>
      <View style={styles.header}>
        <TouchableOpacity onPress={() => navigation.goBack()}>
          <Text style={styles.backButton}>{"<"}</Text>
        </TouchableOpacity>
        <Text style={styles.headerText}>Detail Peminjaman</Text>
      </View>

      {/* Bungkus dengan ScrollView */}
      <ScrollView contentContainerStyle={styles.scrollViewContent}>
        <View style={styles.detailContainer}>
          <Text style={styles.sectionHeader}>Informasi Buku</Text>

          <View style={styles.row}>
            <Text style={styles.label}>Judul Buku:</Text>
            <Text style={styles.value}>{book.judul_buku[1]}</Text>
          </View>

          <View style={styles.row}>
            <Text style={styles.label}>Nama Buku:</Text>
            <Text style={styles.value}>{book.nama_buku}</Text>
          </View>

          <View style={styles.row}>
            <Text style={styles.label}>Dipinjam:</Text>
            <Text style={styles.value}>{book.issued_date}</Text>
          </View>

          <View style={styles.row}>
            <Text style={styles.label}>Dikembalikan:</Text>
            <Text style={styles.value}>{book.return_date}</Text>
          </View>

          <View style={styles.row}>
            <Text style={styles.label}>Status:</Text>
            <Text style={styles.value}>{book.state}</Text>
          </View>

          <View style={styles.divider}></View>

          <Text style={styles.sectionHeader2}>Penalty</Text>

          <View style={styles.row}>
            <Text style={styles.label}>Lewat Batas:</Text>
            <Text style={styles.value}>{book.lewat_batas}</Text>
          </View>

          <View style={styles.row}>
            <Text style={styles.label}>Penalty:</Text>
            <Text style={styles.value}>{book.penalty}</Text>
```

(a) Kode Program

(b) Implementasi Detail Peminjaman Buku

Implementasi halaman Detail Daftar Pinjam Buku pada aplikasi Sistem Informasi Sekolah ini memberikan informasi lengkap terkait transaksi peminjaman buku tertentu. Informasi yang ditampilkan meliputi Judul Buku, Nama Buku dengan kode identifikasi, tanggal Dipinjam, tanggal Dikembalikan, dan Status peminjaman. Selain itu, terdapat bagian Penalty yang menunjukkan apakah ada keterlambatan pengembalian serta jumlah denda jika berlaku. Data ini diambil langsung dari database Odoo, memastikan informasi yang ditampilkan selalu akurat dan terkini. Source code pada gambar mengimplementasikan modul BookDetailScreen untuk menampilkan detail peminjaman buku. Data buku, seperti judul, nama, tanggal dipinjam, tanggal pengembalian, status, batas waktu, dan penalti, diakses melalui parameter route.params. Komponen menggunakan ScrollView untuk memastikan semua informasi dapat dilihat dengan mudah meskipun layar terbatas.

Gambar 3.47 Kode Program & Implementasi Kunjungan Perpustakaan

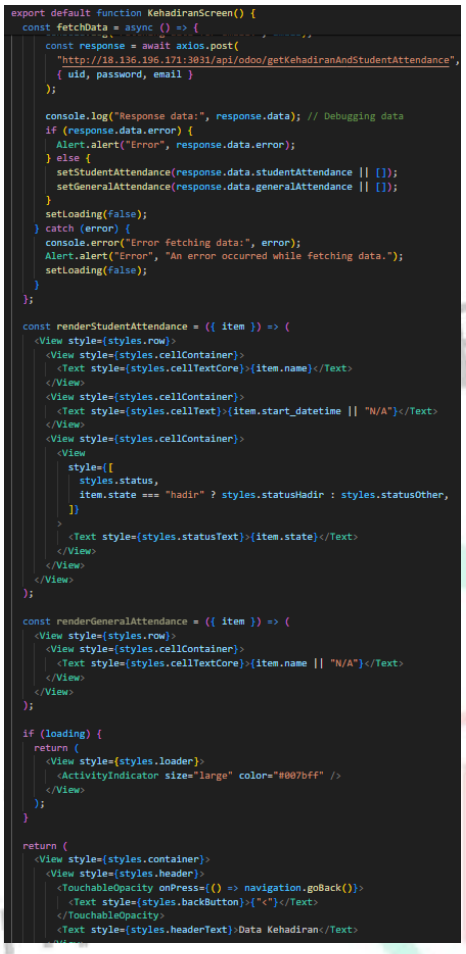
```
export default function RiwayatBerkunjungScreen({ route }) {
  useEffect(() => {
    console.log("Fetching data pengunjung with params:", {
      email,
      uid,
      password,
    }); // Debugging

    const fetchDataPengunjung = async () => {
      try {
        const response = await axios.post(
          "http://18.136.196.171:3031/api/odoo/getDataPengunjung",
          {
            email,
            uid,
            password,
          }
        );
        setDataPengunjung(response.data);
        setLoading(false);
      } catch (error) {
        console.error("Error fetching data pengunjung:", error);
        setLoading(false);
      }
    };

    fetchDataPengunjung();
  }, [email, uid, password]);

  const filteredData = dataPengunjung.filter(
    (item) => item.check_in && item.check_in.includes(search)
  );

  if (loading) {
    return (
      <SafeAreaView style={styles.container}>
        <ActivityIndicator size="large" color="#007bff" />
      </SafeAreaView>
    );
  }

  return (
    <SafeAreaView style={styles.container}>
      <View style={styles.header}>
        <TouchableOpacity onPress={() => navigation.goBack()}>
          <Text style={styles.backButton}>{"<"}</Text>
        </TouchableOpacity>
        <Text style={styles.headerText}>Data Berkunjung</Text>
      </View>
      <TextInput
        style={styles.searchInput}
        placeholder="Cari berdasarkan tanggal (YYYY-MM-DD)"
        value={search}
        onChangeText={setSearch}
      />
      {filteredData.length === 0 ? (
        <View style={styles.noVisitsContainer}>
          <Text style={styles.noVisitsText}>
            Anda belum pernah berkunjung ke perpustakaan.
          </Text>
        </View>
      ) : (
        <FlatList
          data={filteredData}
          keyExtractor={(item) => item.id.toString()}
          renderItem={({ item }) => (
            <View style={styles.card}>
```

(a) Kode Program

(b) Implementasi Kunjungan Perpustakaan

Implementasi halaman Data Berkunjung pada aplikasi Sistem Informasi Sekolah ini menampilkan informasi kunjungan perpustakaan yang dilakukan oleh pengguna. Setiap entri menampilkan Nama, Tanggal dan Waktu kunjungan, serta Nomor Telepon pengguna yang berkunjung. Di bagian atas halaman, terdapat fitur pencarian berdasarkan tanggal (format YYYY-MM-DD) yang memudahkan pengguna untuk menemukan data kunjungan tertentu. Data kunjungan ini diambil langsung dari database Odoo, memastikan bahwa informasi yang ditampilkan selalu akurat dan diperbarui secara real-time. Implementasi ini menambahkan elemen visual penuh, fitur pencarian, dan data terperinci yang langsung terhubung dengan Odoo. Fungsi fetchDataPengunjung memanggil API melalui HTTP POST untuk mengambil data kunjungan, seperti nama, tanggal dan waktu kunjungan, serta nomor telepon.

Gambar 3.48 Kode Program & Implementasi Data Kehadiran



```
export default function KehadiranScreen() {
  const fetchData = async () => {
    const response = await axios.post(
      "http://18.136.196.171:3031/api/odoo/getKehadiranAndStudentAttendance",
      { uid, password, email }
    );

    console.log("Response data:", response.data); // Debugging data
    if (response.data.error) {
      Alert.alert("Error", response.data.error);
    } else {
      setStudentAttendance(response.data.studentAttendance || []);
      setGeneralAttendance(response.data.generalAttendance || []);
    }
    setLoading(false);
  } catch (error) {
    console.error("Error fetching data:", error);
    Alert.alert("Error", "An error occurred while fetching data.");
    setLoading(false);
  }
};

const renderStudentAttendance = ({ item }) => (
  <View style={styles.row}>
    <View style={styles.cellContainer}>
      <Text style={styles.cellTextCore}>{item.name}</Text>
    </View>
    <View style={styles.cellContainer}>
      <Text style={styles.cellText}>{item.start_datetime || "N/A"}</Text>
    </View>
    <View style={styles.cellContainer}>
      <View
        style={[
          styles.status,
          item.state === "hadir" ? styles.statusHadir : styles.statusOther,
        ]}
      >
        <Text style={styles.statusText}>{item.state}</Text>
      </View>
    </View>
  </View>
);

const renderGeneralAttendance = ({ item }) => (
  <View style={styles.row}>
    <View style={styles.cellContainer}>
      <Text style={styles.cellTextCore}>{item.name || "N/A"}</Text>
    </View>
  </View>
);

if (loading) {
  return (
    <View style={styles.loader}>
      <ActivityIndicator size="large" color="#007bff" />
    </View>
  );
}

return (
  <View style={styles.container}>
    <View style={styles.header}>
      <TouchableOpacity onPress={() => navigation.goBack()}>
        <Text style={styles.backButton}>{"<"}</Text>
      </TouchableOpacity>
      <Text style={styles.headerText}>Data Kehadiran</Text>
```

(a) Kode Program

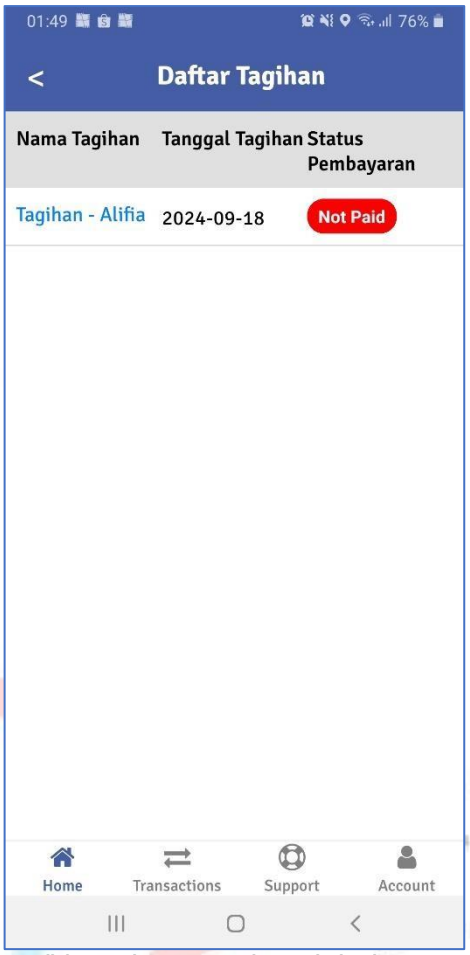
(b) Implementasi Data Kehadiran

Implementasi halaman Data Kehadiran pada aplikasi Sistem Informasi Sekolah ini menampilkan daftar kehadiran siswa secara rinci dengan kolom Nama Siswa, Tanggal Absensi, dan Status. Setiap status kehadiran ditampilkan dengan warna untuk memudahkan identifikasi: hijau untuk hadir dan merah untuk alpha (tidak hadir), serta status draft untuk kehadiran yang belum dikonfirmasi. Data ini diambil langsung dari database Odoo, sehingga pengguna selalu mendapatkan informasi kehadiran yang up-to-date dan akurat. Source code pada gambar mengimplementasikan modul DataKehadiranScreen yang menampilkan daftar kehadiran siswa. Fungsi fetchData memanggil API menggunakan HTTP POST untuk mengambil data kehadiran, termasuk nama siswa, tanggal absensi, dan status kehadiran. Data yang diterima disimpan dalam state dan dirender menggunakan fungsi renderStudentAttendance, yang menampilkan nama, tanggal, dan status dalam bentuk daftar.

Gambar 3.49 Kode Program & Implementasi Modul Biaya

```
export default function TagihanScreen() {
  const fetchData = async () => {
    const response = await axios.post(
      "http://18.136.196.171:3031/api/odoo/getBillDetails",
      { uid, password, email }
    );

    console.log("Bill data:", response.data); // Debugging data
    if (response.data.error) {
      Alert.alert("Error", response.data.error);
    } else {
      const billDataArray = Array.isArray(response.data)
        ? response.data
        : [response.data];
      setData(billDataArray);
    }
    setLoading(false);
  } catch (error) {
    setLoading(false);
  }
};

const renderItem = ({ item }) => (
  <View style={styles.row}>
    <View style={styles.cellContainer}>
      <TouchableOpacity
        onPress={() =>
          navigation.navigate("DetailTagihanScreen", {
            bill: item,
          })
        }
      >
        <Text style={styles.cellTextCore}>{item.name_tagihan}</Text>
      </TouchableOpacity>
    </View>
    <View style={styles.dateContainer}>
      <Text style={styles.cellText}>{item.date}</Text>
    </View>
    <View style={styles.cellContainer}>
      <View
        style={[
          styles.paymentStatus,
          item.payment_state === "paid"
            ? styles.paidBackground
            : styles.notPaidBackground,
        ]}
      >
        <Text style={[styles.paymentStatusText, styles.boldText]}>
          {item.payment_state === "paid" ? "Paid" : "Not Paid"}
        </Text>
      </View>
    </View>
  </View>
);

if (loading) {
  return (
    <View style={styles.loader}>
      <ActivityIndicator size="large" color="#007bff" />
    </View>
  );
}

return (
  <View style={styles.container}>
    <View style={styles.header}>
      <TouchableOpacity onPress={() => navigation.goBack()}>
```

(a) Kode Program



(b) Implementasi Modul Biaya

Implementasi halaman Daftar Tagihan pada aplikasi Sistem Informasi Sekolah ini menampilkan rincian tagihan siswa, yang mencakup kolom Nama Tagihan, Tanggal Tagihan, dan Status Pembayaran. Setiap status pembayaran ditampilkan dengan label berwarna, seperti merah untuk Not Paid, memudahkan pengguna dalam memantau status pembayaran secara sekilas. Data tagihan ini diambil langsung dari database Odoo, memastikan bahwa informasi yang ditampilkan selalu mutakhir dan akurat. Implementasi ini lebih lengkap dibandingkan dengan wireframe awal karena menyediakan elemen visual yang lebih informatif dan integrasi data real-time dari Odoo, membantu pengguna untuk mengelola dan melacak pembayaran tagihan dengan mudah.

Gambar 3.50 Kode Program & Implementasi Modul Detail Biaya

```
export default function DetailTagihanScreen({ route }) {
  useEffect(() => {
    const fetchBillLines = async () => {
      try {
        const response = await axios.post(
          "http://18.136.196.171:3031/api/odoo/getTagihanSiswaLine",
          {
            uid,
            password,
            slip_id: bill.id,
          }
        );

        if (response.data.error) {
          Alert.alert("Error", response.data.error);
        } else {
          setBillLines(response.data);
          console.log("Bill Lines:", response.data); // Add this log
        }
      } catch (error) {
        console.error("Error fetching bill lines:", error);
        Alert.alert("Error", "An error occurred while fetching bill lines.");
      } finally {
        setLoading(false);
      }
    };

    fetchBillLines();
  }, [bill.id, uid, password]);

  if (loading) {
    return (
      <SafeAreaView style={styles.container}>
        <ActivityIndicator size="large" color="#007bff" />
      </SafeAreaView>
    );
  }

  if (!billDetails) {
    return (
      <SafeAreaView style={styles.container}>
        <Text>No bill details found.</Text>
      </SafeAreaView>
    );
  }

  return (
    <SafeAreaView style={styles.container}>
      <View style={styles.header}>
        <TouchableOpacity onPress={() => navigation.goBack()}>
          <Text style={styles.backButton}>{"<"}</Text>
        </TouchableOpacity>
        <Text style={styles.headerText}>Detail Tagihan</Text>
      </View>
      <ScrollView contentContainerStyle={styles.content}>
        <View style={styles.detailRow}>
          <Text style={styles.label}>Nama Tagihan:</Text>
          <Text style={styles.value}>{billDetails.name_tagihan}</Text>
        </View>
        <View style={styles.detailRow}>
          <Text style={styles.label}>Tanggal Tagihan:</Text>
          <Text style={styles.value}>{billDetails.date}</Text>
        </View>
        <View style={styles.detailRow}>
          <Text style={styles.label}>Struktur Biaya:</Text>
          <Text style={styles.value}>{billDetails.struktur_biaya_id[1]}</Text>
```

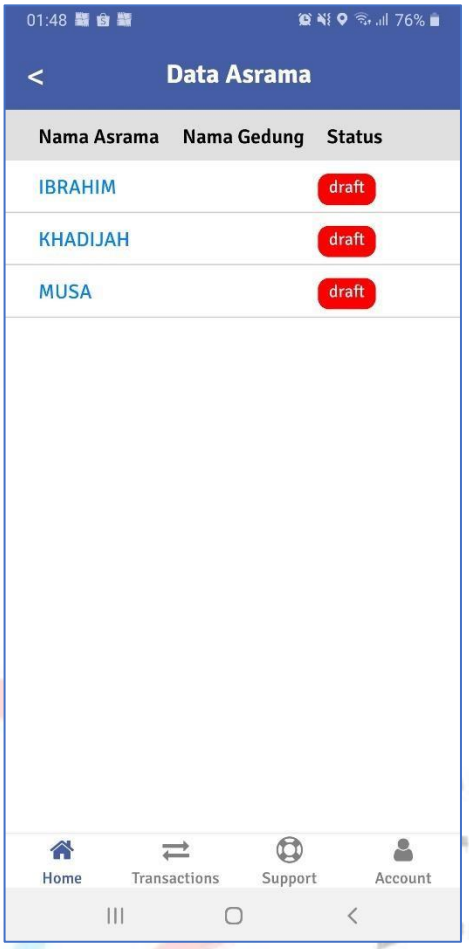
(a) Kode Program

(b) Implementasi Modul Detail Biaya

Implementasi halaman Detail Tagihan pada aplikasi Sistem Informasi Sekolah ini menampilkan rincian lengkap terkait tagihan tertentu. Informasi yang ditampilkan meliputi Nama Tagihan, Tanggal Tagihan, Struktur Biaya (seperti SPP dan Uang Gedung), Status tagihan, dan Status Pembayaran, yang disertai label berwarna untuk memudahkan identifikasi (misalnya Not Paid dalam warna merah). Di bagian bawah, terdapat tabel Rincian Biaya yang menunjukkan detail biaya seperti nama, kode biaya, durasi, dan jumlah yang harus dibayarkan. Data ini diambil langsung dari database Odoo, memastikan informasi yang ditampilkan selalu akurat dan terkini. Implementasi ini memberikan pengalaman pengguna yang lebih kaya dibandingkan wireframe awal dengan menampilkan data terperinci dan elemen visual yang membantu dalam memantau status keuangan secara efisien. Source code pada gambar mengimplementasikan modul Detail Tagihan menggunakan API untuk mengambil data tagihan seperti nama tagihan, tanggal, struktur biaya, status, dan rincian.

Gambar 3.51 Kode Program & Implementasi Modul Asrama

(a) Kode Program



(b) Implementasi Modul Asrama

Implementasi halaman Data Asrama pada aplikasi Sistem Informasi Sekolah ini menampilkan daftar asrama siswa beserta statusnya. Setiap baris dalam daftar menunjukkan Nama Asrama, Nama Gedung, dan Status, dengan status ditampilkan dalam label berwarna, seperti merah untuk draft, yang memudahkan pemantauan. Data ini terintegrasi langsung dengan database Odoo, sehingga informasi yang ditampilkan selalu terkini dan akurat. Implementasi ini lebih fungsional dibandingkan wireframe awal dengan tambahan elemen visual yang memperjelas status setiap asrama dan integrasi data secara real-time, yang membantu pengguna untuk melihat status pengelolaan asrama dengan lebih efisien. Source code pada gambar mengimplementasikan modul AsramaScreen untuk menampilkan daftar asrama. Fungsi fetchData memanggil API melalui HTTP POST untuk mengambil data asrama, seperti nama asrama, nama gedung, dan status.

Gambar 3.52 Kode Program & Implementasi Detail Modul Asrama

```
export default function AsramaDetailScreen({ route }) {
  useEffect(() => {
    const fetchAsramaDetails = async () => {
      const response = await axios.post(
        "http://18.136.196.171:3031/api/odoo/getAsramaDetails",
        {
          uid,
          password,
          email,
        }
      );

      console.log("Response data:", response.data);

      if (response.data.error) {
        Alert.alert("Error", response.data.error);
      } else {
        // Assuming response.data is an array
        const details = response.data.find((item) => item.id === asrama.id);
        setAsramaDetails(details);
      }
    } catch (error) {
      console.error("Error fetching asrama details:", error);
      Alert.alert(
        "Error",
        "An error occurred while fetching asrama details."
      );
    } finally {
      setLoading(false);
    }
  };

  fetchAsramaDetails();
}, [asrama.id, uid, password, email]);

if (loading) {
  return (
    <SafeAreaView style={styles.container}>
      <ActivityIndicator size="large" color="#007bff" />
    </SafeAreaView>
  );
}

if (!asramaDetails) {
  return (
    <SafeAreaView style={styles.container}>
      <Text>No asrama details found.</Text>
    </SafeAreaView>
  );
}

return (
  <SafeAreaView style={styles.container}>
    <View style={styles.header}>
      <TouchableOpacity onPress={() => navigation.goBack()}>
        <Text style={styles.backButton}>{"<"}</Text>
      </TouchableOpacity>
      <Text style={styles.headerText}>Detail Asrama</Text>
    </View>
    <ScrollView contentContainerStyle={styles.content}>
      <View style={styles.section}>
        <Text style={styles.sectionTitle}>Informasi Asrama</Text>
        <View style={styles.row}>
          <Text style={styles.label}>Nama:</Text>
          <Text style={styles.value}>
            {asramaDetails.siswa_id?.[1] || "N/A"}
          </Text>
```

(a) Kode Program

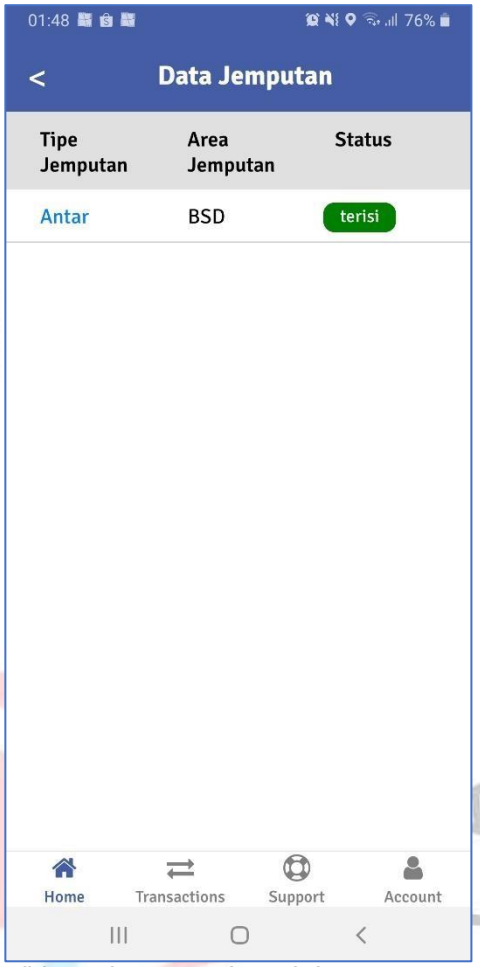
(b) Implementasi Modul Asrama

Implementasi halaman Detail Asrama pada aplikasi ini memberikan informasi lengkap mengenai detail asrama siswa, termasuk nama siswa, nama asrama, jenis asrama, gedung, lantai, fasilitas, dan status pendaftaran. Status ditampilkan dengan label berwarna untuk memudahkan identifikasi, contohnya status draft ditandai dengan warna merah. Selain itu, informasi pendaftaran seperti tanggal registrasi dan tanggal selesai juga tersedia untuk memudahkan pemantauan. Data ini langsung diambil dari database Odoo, memastikan bahwa informasi yang disajikan selalu akurat dan up-to-date. Implementasi ini lebih fungsional dibandingkan dengan wireframe awal. Source code pada gambar mengimplementasikan modul Detail Asrama dalam aplikasi mobile, menggunakan API untuk pengambilan data asrama secara dinamis. Fungsi fetchAsramaDetails bertugas memanggil API melalui HTTP POST dengan parameter autentikasi seperti email dan password, kemudian mengambil informasi penting terkait asrama.

Gambar 3.53 Kode Program & Implementasi Modul Jemputan

```
export default function JemputanScreen() {
  const fetchData = async () => {
    try {
      console.log("Fetching data for email:", email);
      const response = await axios.post(
        "http://18.136.196.171:3031/api/odoo/getJemputanDetails",
        { uid, password, email }
      );

      console.log("Asrama data:", response.data); // Debugging data
      if (response.data.error) {
        Alert.alert("Error", response.data.error);
      } else {
        setData(response.data);
      }
      setLoading(false);
    } catch (error) {
      setLoading(false);
    }
  };

  const renderItem = ({ item }) => (
    <View style={styles.row}>
      <View style={styles.cellContainer}>
        <TouchableOpacity
          onPress={() =>
            navigation.navigate("JemputanDetailScreen", {
              jemputan: item,
            })
          }
        >
          <Text style={styles.cellTextCore}>{item.tipe[1]}</Text>
        </TouchableOpacity>
      </View>
      <View style={styles.cellContainer}>
        <Text style={styles.cellText}>{item.area_id[1]}</Text>
      </View>
      <View style={styles.cellContainer}>
        <View
          style={[
            styles.status,
            item.state === "terisi" ? styles.statusFilled : styles.statusOther,
          ]}
        >
          <Text style={styles.statusText}>{item.state}</Text>
        </View>
      </View>
    </View>
  );

  if (loading) {
    return (
      <View style={styles.loader}>
        <ActivityIndicator size="large" color="#007bff" />
      </View>
    );
  }

  return (
    <View style={styles.container}>
      <View style={styles.header}>
        <TouchableOpacity onPress={() => navigation.goBack()}>
          <Text style={styles.backButton}>{"<"}</Text>
        </TouchableOpacity>
        <Text style={styles.headerText}>Data Jemputan</Text>
      </View>
```

(a) Kode Program



(b) Implementasi Modul Jemputan

Implementasi halaman Data Jemputan ini menampilkan informasi terkait layanan jemputan siswa, termasuk tipe jemputan, area jemputan, dan status ketersediaannya. Pada contoh ini, data menunjukkan tipe jemputan "Antar" di area "BSD" dengan status "terisi," yang ditandai dengan label hijau untuk menunjukkan ketersediaan layanan. Data ini diambil langsung dari database Odoo, memastikan agar informasi selalu terkini dan akurat. Implementasi ini mempermudah pengguna dalam memahami status dan detail layanan jemputan yang tersedia, dengan tampilan yang lebih jelas dan terstruktur dibandingkan dengan desain wireframe awal. Fungsi fetchData memanggil API menggunakan HTTP POST untuk mengambil informasi jemputan, termasuk tipe, area, dan status. Data yang diterima disimpan dalam state dan dirender dalam daftar menggunakan fungsi renderItem.

Gambar 3.54 Kode Program & Implementasi Detail Modul Jemputan

```
export default function JemputanDetailScreen({ route }) {
  useEffect(() => {
    const fetchJemputanDetails = async () => {
      const response = await axios.post(
        "http://18.136.196.171:3031/api/odoo/getJemputanDetails",
        {
          uid,
          password,
          email,
        }
      );

      console.log("Response data:", response.data);

      if (response.data.error) {
        Alert.alert("Error", response.data.error);
      } else {
        // Assuming response.data is an array
        const details = response.data.find((item) => item.id === jemputan.id);
        setJemputanDetails(details);
      }
    } catch (error) {
      console.error("Error fetching jemputan details:", error);
      Alert.alert(
        "Error",
        "An error occurred while fetching jemputan details."
      );
    } finally {
      setLoading(false);
    }
  };

  fetchJemputanDetails();
}, [jemputan.id, uid, password, email]);

if (loading) {
  return (
    <View style={styles.loader}>
      <ActivityIndicator size="large" color="#007bff" />
    </View>
  );
}

if (!jemputanDetails) {
  return (
    <View style={styles.container}>
      <Text>No Jemputan details found.</Text>
    </View>
  );
}

return (
  <View style={styles.container}>
    <View style={styles.header}>
      <TouchableOpacity onPress={() => navigation.goBack()}>
        <Text style={styles.backButton}>{"<"}</Text>
      </TouchableOpacity>
      <Text style={styles.headerText}>Detail Jemputan</Text>
    </View>
    <ScrollView contentContainerStyle={styles.content}>
      <View style={styles.section}>
        <Text style={styles.sectionTitle}>Informasi Jemputan</Text>
        <View style={styles.row}>
          <Text style={styles.label}>Nama:</Text>
          <Text
            style={styles.valueAlamat}
            numberOfLines={0} // Allow multiple lines
```

(a) Kode Program

(b) Implementasi Detail Modul Jemputan

Implementasi halaman Detail Jemputan ini memberikan informasi rinci tentang layanan jemputan untuk siswa, yang mencakup nama siswa, alamat, area, tipe layanan (Antar), hari operasional (Senin, Selasa, Rabu), periode (Weekly), harga layanan, serta base price. Status layanan ditampilkan dengan label hijau "terisi" untuk menunjukkan ketersediaan. Data ini diambil secara langsung dari database Odoo, yang memungkinkan pengguna mendapatkan informasi akurat dan real-time terkait layanan jemputan. Implementasi ini mengembangkan konsep wireframe dengan menambahkan detail penting yang memudahkan pengguna dalam mengakses informasi lengkap mengenai layanan jemputan. Source code pada gambar mengimplementasikan modul Detail Jemputan dalam aplikasi mobile dengan penggunaan API untuk pengambilan data. Fungsi fetchJemputanDetails memanggil API melalui HTTP POST untuk mengambil informasi jemputan berdasarkan ID.

Gambar 3.55 Kode Program & Implementasi Modul Jadwal

```
export default function JadwalPelajaranScreen() {

  const fetchJadwal = async () => {
    try {
      const response = await axios.post(
        "http://18.136.196.171:3031/api/odoo/getJadwalKelas",
        { uid, password, email }
      );
      setJadwal(response.data);
      setOriginalJadwal(response.data); // Store the original data

      // Mengambil semua pelajaran dan menghilangkan duplikat
      const subjects = [
        ...new Set(response.data.map((item) => item.pelajaran_id[1])),
      ];
      setPelajaranList(subjects);
    } catch (error) {
      console.error("Error fetching jadwal data:", error);
    }
  };

  const sortByDate = () => {
    const sortedJadwal = [...jadwal].sort(
      (a, b) => new Date(a.tanggal) - new Date(b.tanggal)
    );
    setJadwal(sortedJadwal);
    setFilter("tanggal");
  };

  const filterByPelajaran = (pelajaran) => {
    const filteredJadwal = originalJadwal.filter(
      // Use originalJadwal here
      (item) => item.pelajaran_id[1] === pelajaran
    );
    setJadwal(filteredJadwal);
    setSelectedPelajaran(pelajaran);
    setFilter("pelajaran");
    setModalVisible(false);
  };

  const resetFilter = () => {
    setJadwal(originalJadwal); // Reset to the original jadwal data
    setFilter("semua");
    setSelectedPelajaran(null);
  };

  const renderItem = (item, index) => {
    const subjectName =
      item.pelajaran_id && item.pelajaran_id[1]
        ? capitalizeFirstLetter(item.pelajaran_id[1])
        : "Unknown Subject";

    const normalizedSubjectName = removeSpaces(subjectName);
    const subjectIcon =
      subjectIcons[normalizedSubjectName] ||
      require("./assets/icons/default.png");

    return (
      <View key={index} style={styles.itemWrapper}>
        <TouchableOpacity
          onPress={() =>
            navigation.navigate("DetailPelajaranScreen", {
              pelajaran: item.pelajaran_id[1],
            })
          }
        >
          <View style={styles.cellContainer}>
```

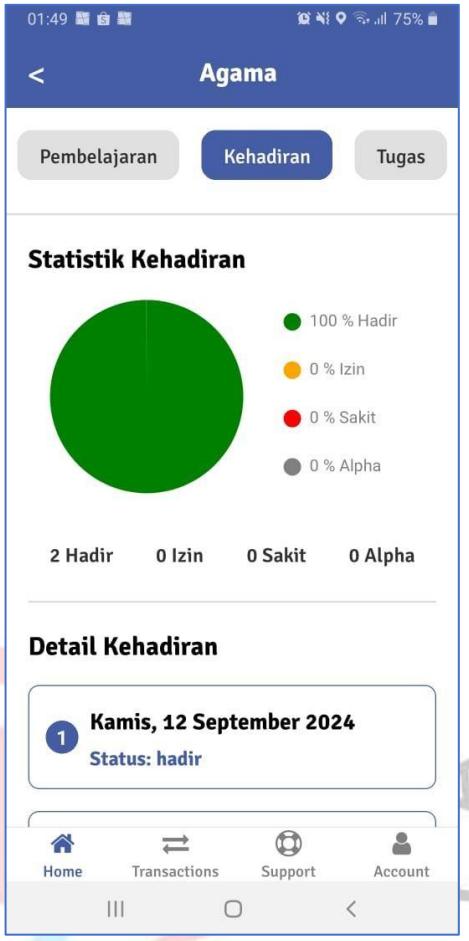
(a) Kode Program

(b) Implementasi Modul Jadwal

Implementasi halaman Jadwal Pelajaran ini menampilkan daftar mata pelajaran yang dijadwalkan untuk siswa, seperti Agama, Matematika, Fisika, Biologi, Kimia, dan Seni, masing-masing dilengkapi dengan ikon yang mewakili setiap pelajaran. Pengguna juga memiliki opsi untuk melihat jadwal dalam Tampilan Kalender, yang memungkinkan visualisasi jadwal dalam bentuk kalender yang lebih komprehensif. Data ini terhubung langsung dengan database Odoo untuk memastikan informasi yang akurat dan terstruktur terkait mata pelajaran yang diambil oleh siswa. Implementasi ini memperluas wireframe dengan menambahkan ikon dan akses ke tampilan kalender, memberikan pengalaman pengguna yang lebih interaktif dan informatif. Source code pada gambar mengimplementasikan modul Jadwal Pelajaran untuk aplikasi mobile. Fungsi fetchData memanggil API melalui HTTP POST untuk mengambil data jadwal dari server, yang kemudian diurutkan dengan sortData dan difilter menggunakan filterPelajaran.

Gambar 3.56 Kode Program & Implementasi Detail Modul Jadwal

```
export default function DetailPelajaranScreen({ route }) {
  const calculateAttendancePercentage = (attendanceData) => {
    setAttendanceStats({
      hadir: (hadirCount / totalSessions) * 100 || 0,
      izin: (izinCount / totalSessions) * 100 || 0,
      sakit: (sakitCount / totalSessions) * 100 || 0,
      alpha: (alphaCount / totalSessions) * 100 || 0,
      hadirCount,
      izinCount,
      sakitCount,
      alphaCount,
      totalSessions,
    });
  };

  const renderAttendanceChart = () => {
    const data = [
      {
        name: "% Hadir",
        population: parseFloat(attendanceStats.hadir.toFixed(1)),
        color: "green",
        legendFontColor: "#7F7F7F",
        legendFontSize: 15,
      },
      {
        name: "% Izin",
        population: parseFloat(attendanceStats.izin.toFixed(1)),
        color: "orange",
        legendFontColor: "#7F7F7F",
        legendFontSize: 15,
      },
      {
        name: "% Sakit",
        population: parseFloat(attendanceStats.sakit.toFixed(1)),
        color: "red",
        legendFontColor: "#7F7F7F",
        legendFontSize: 15,
      },
      {
        name: "% Alpha",
        population: parseFloat(attendanceStats.alpha.toFixed(1)),
        color: "gray",
        legendFontColor: "#7F7F7F",
        legendFontSize: 15,
      },
    ];

    return (
      <PieChart
        data={data}
        width={screenWidth - 40}
        height={220}
        chartConfig={{
          backgroundColor: "#e26a00",
          backgroundGradientFrom: "#fb8c00",
          backgroundGradientTo: "#ffa726",
          color: (opacity = 1) => `rgba(255, 255, 255, ${opacity})`,
        }}
        accessor={"population"}
        backgroundColor={"transparent"}
        paddingLeft={"15"}
        absolute
      />
    );
  };
```



(a) Kode Program

(b) Implementasi Detail Modul Jadwal

Implementasi halaman Detail Modul Jadwal untuk mata pelajaran Agama ini menyajikan statistik kehadiran siswa dengan visualisasi berupa grafik lingkaran yang menunjukkan persentase kehadiran, izin, sakit, dan alpha. Di bawah grafik, terdapat ringkasan jumlah masing-masing status kehadiran. Pengguna juga dapat melihat detail spesifik dari setiap kehadiran dalam daftar harian, seperti status kehadiran pada Kamis, 12 September 2024, yang tercatat sebagai "hadir." Halaman ini terintegrasi dengan data kehadiran dari database Odoo, sehingga memastikan akurasi dan kemudahan pelacakan kehadiran untuk setiap mata pelajaran yang diikuti siswa. Source code pada gambar mengimplementasikan modul Statistik Kehadiran dalam aplikasi mobile dengan penggunaan API untuk pengambilan data kehadiran. Fungsi calculateAttendancePercentage menghitung persentase kehadiran berdasarkan data yang diambil dari API, termasuk status hadir, izin, sakit, dan alfa.

3.6.7 Tahap Pengujian Sistem

Tahapan Pengujian dalam pengembangan perangkat lunak digunakan untuk mengindentifikasi *bug* atau kesalahan kerja sistem secara fungsional atau non-fungsional (Subagia et al., 2020). Dalam pengujian sistem yang praktikan lakukan pada sistem ini, praktikan menggunakan metode black box testing. Black box merupakan metode pengujian sistem yang menguji struktur logika sistem. Pengujian ini dilakukan berdasarkan sudut pandang pengguna yang nantinya akan menggunakan sistem yang dikembangkan. Salah satu keuntungan melakukan black box testing yaitu memudahkan pencarian aspek sistem seperti logika dan alur kerja yang tidak sesuai dengan kebutuhan awal (Praniffa et al., 2023). Pengujian pertama yang dilakukan yaitu menguji skema login pada sistem. Berikut adalah table black box yang praktikan lakukan.

Tabel 3.4 Skenario Pengujian Black Box Login

No.	Skenario Pengujian	Hasil yang Diharapkan	Validasi
1.	Mengisi email tetapi password dikosongkan	Muncul pesan “Email atau Password tidak sesuai”	Valid
2.	Mengisi email yang tidak terdaftar	Muncul pesan “Email atau Password tidak sesuai”	Valid
3.	Mengisi email namun password yang dimasukkan tidak sesuai	Muncul pesan “Email atau Password tidak sesuai”	Valid
4.	Mengisi email dan password dengan benar	Masuk ke halaman utama	Valid

Tabel 3.4 merupakan skenario pengujian black box login pada sistem yang praktikan buat. Hasil pengujian sistem pada gambar 3.4 sepenuhnya sudah sesuai dengan harapan praktikan.

Tabel 3.5 Skenario Pengujian Black Box Modul Perpustakaan

No.	Skenario Pengujian	Hasil yang Diharapkan	Validasi
1.	Mengakses menu daftar ketersediaan buku	Masuk ke dalam halaman daftar ketersediaan buku	Valid
2.	Mengkases menu daftar ketersediaan buku	Muncul informasi daftar ketersediaan buku	Valid
3.	Berhasil mencari daftar buku pada halaman ketersediaan buku	Muncul informasi daftar buku yang dicari	Valid
4.	Gagal mencari daftar buku pada halaman ketersediaan buku	Muncul pesan “Daftar buku tidak tersedia”	Valid
5	Mengakses menu daftar peminjaman buku	Masuk ke dalam halaman daftar peminjaman buku	Valid
6	Mengakses menu daftar peminjaman buku	Muncul informasi daftar peminjaman buku	Valid
7	Mengakses menu e-book	Masuk ke dalam halaman e-book	Valid
8	Mengakses menu daftar e-book	Muncul informasi daftar e-book	Valid
9	Berhasil mencari daftar e-book pada halaman	Muncul informasi daftar e-book yang dicari	Valid
10	Gagal mencari daftar e-book pada halaman e-book	Muncul pesan “e-book tidak tersedia”	Valid
11	Menekan tombol back pada modul perpustakaan	Kembali ke halaman homescreen	Valid

Tabel 3.5 merupakan skenario pengujian black box mengakses modul perpustkaan pada sistem yang praktikan buat. Hasil pengujian sistem pada gambar 3.5 sepenuhnya sudah sesuai dengan harapan praktikan.

Tabel 3.6 Skenario Pengujian Black Box Modul Biaya

No.	Skenario Pengujian	Hasil yang Diharapkan	Validasi
1.	Mengakses modul biaya	Masuk ke dalam halaman daftar tagihan	Valid
2.	Mengkases menu daftar tagihan siswa	Muncul daftar informasi tagihan siswa	Valid
3.	Menekan tombol “lihat detail” pada menu daftar tagihan siswa	Muncul detail informasi daftar tagihan siswa	Valid
4.	Menekan tombol back pada modul biaya	Kembali ke halaman homescreen	Valid

Tabel 3.6 merupakan skenario pengujian black box mengakses modul biaya pada sistem yang praktikan buat. Hasil pengujian sistem pada gambar 3.6 sepenuhnya sudah sesuai dengan harapan praktikan.

Tabel 3.7 Skenario Pengujian Black Box Modul Asrama

No.	Skenario Pengujian	Hasil yang Diharapkan	Validasi
1.	Mengakses modul Asrama	Masuk ke dalam halaman daftar registrasi asrama	Valid
2.	Mengkases menu daftar registrasi asrama	Muncul daftar informasi registrasi asrama	Valid
3.	Menekan tombol “lihat detail” pada menu registrasi asrama	Muncul detail informasi daftar registrasi asrama	Valid
4.	Menekan tombol back pada modul asrama	Kembali ke halaman homescreen	Valid

Tabel 3.7 merupakan skenario pengujian black box mengakses modul asrama pada sistem yang praktikan buat. Hasil pengujian sistem pada gambar 3.7 sepenuhnya sudah sesuai dengan harapan praktikan.

Tabel 3.8 Skenario Pengujian Black Box Modul Jemputan

No.	Skenario Pengujian	Hasil yang Diharapkan	Validasi
1.	Mengakses modul jemputan	Masuk ke dalam halaman daftar registrasi jemputan	Valid
2.	Mengkases menu daftar registrasi jemputan	Muncul daftar informasi registrasi jemputan	Valid
3.	Menekan tombol “lihat detail” pada menu registrasi jemputan	Muncul detail informasi daftar registrasi jemputan	Valid
4.	Menekan tombol back pada modul jemputan	Kembali ke halaman homescreen	Valid

Tabel 3.8 merupakan skenario pengujian black box mengakses modul jemputan pada sistem yang praktikan buat. Hasil pengujian sistem pada gambar 3.8 sepenuhnya sudah sesuai dengan harapan praktikan.

Tabel 3.9 Skenario Pengujian Black Box Modul Kehadiran

No.	Skenario Pengujian	Hasil yang Diharapkan	Validasi
1.	Mengakses modul kehadiran	Masuk ke dalam halaman daftar kehadiran	Valid
2.	Mengkases menu daftar kehadiran	Muncul daftar informasi kehadiran	Valid
3.	Menekan tombol back pada modul kehadiran	Kembali ke halaman homescreen	Valid

Tabel 3.9 merupakan skenario pengujian black box mengakses modul kehadiran pada sistem yang praktikan buat. Hasil pengujian sistem pada gambar 3.9 sepenuhnya sudah sesuai dengan harapan praktikan.

Tabel 3.10 Skenario Pengujian Black Box Modul Jadwal

No.	Skenario Pengujian	Hasil yang Diharapkan	Validasi
1.	Mengakses modul jadwal	Masuk ke dalam halaman jadwal	Valid
2.	Mengakses modul jadwal	Muncul daftar informasi jadwal pelajaran	Valid
3.	Memilih pelajaran pada daftar pelajaran	Muncul daftar informasi kehadiran, tugas, dan materi satu semester	Valid
4.	Menekan tombol back pada modul jadwal	Kembali ke halaman homescreen	Valid

Tabel 3.10 merupakan skenario pengujian black box mengakses modul jadwal pada sistem yang praktikan buat. Hasil pengujian sistem pada gambar 3.10 sepenuhnya sudah sesuai dengan harapan praktikan.

Tabel 3.11 Skenario Pengujian Black Box Informasi Barang Hilang

No.	Skenario Pengujian	Hasil yang Diharapkan	Validasi
1.	Mengakses informasi barang hilang pada homescreen	Muncul daftar informasi barang hilang	Valid
2.	Mengakses informasi barang hilang	Muncul detail informasi barang hilang	Valid
3.	Menekan tombol back pada halaman informasi barang hilang	Kembali ke halaman homescreen	Valid

Tabel 3.11 merupakan skenario pengujian black box mengakses daftar informasi barang hilang pada sistem yang praktikan buat. Hasil pengujian sistem pada gambar 3.11 sepenuhnya sudah sesuai dengan harapan praktikan.

Tabel 3.12 Skenario Pengujian Black Box Informasi Sekolah

No.	Skenario Pengujian	Hasil yang Diharapkan	Validasi
1.	Mengakses informasi sekolah pada homescreen	Muncul daftar informasi sekolah	Valid
2.	Mengakses informasi sekolah	Muncul detail informasi sekolah	Valid
4.	Menekan tombol back pada halaman informasi sekolah	Kembali ke halaman homescreen	Valid

Tabel 3.12 merupakan skenario pengujian black box mengakses daftar informasi sekolah pada sistem yang praktikan buat. Hasil pengujian sistem pada gambar 3.12 sepenuhnya sudah sesuai dengan harapan praktikan.

Tabel 3.13 Skenario Pengujian Black Box Informasi Data Diri

No.	Skenario Pengujian	Hasil yang Diharapkan	Validasi
1.	Berhasil mengakses menu settings	Muncul informasi data diri siswa	Valid
2	Gagal mengakses menu settings	Muncul pesan “Data diri siswa tidak tersedia”	Valid
3.	Menekan tombol menu homescreen	Kembali ke halaman homescreen	Valid

Tabel 3.13 merupakan skenario pengujian black box mengakses informasi data diri siswa pada sistem yang praktikan buat. Hasil pengujian sistem pada gambar 3.13 sepenuhnya sudah sesuai dengan harapan praktikan.

3.3 Kendala yang Dihadapi

Dalam pelaksanakaan kerja profesi yang dilaksanakan di PT. ALPHASOFT, praktikan mengalamai beberapa kendala selama proses berlangsung, diantaranya yaitu.

1. **Kurangnya literasi terkait integrasi odoo dengan react native**

   Dalam mengembangkan aplikasi mobile yang terintegrasi dengan Odoo menggunakan React Native, proses integrasi ini cukup kompleks karena melibatkan berbagai aspek teknis seperti cara kerja API, pengelolaan data, dan sinkronisasi antara backend (Odoo) dan frontend (React Native). Odoo sendiri adalah platform ERP dengan modul dan fitur yang mendalam, dan API yang disediakan membutuhkan pemahaman khusus agar data dapat ditarik dan ditampilkan secara akurat di aplikasi mobile. Selain itu, pengelolaan state dalam React Native juga memerlukan pengetahuan yang baik untuk memastikan bahwa data yang ditampilkan selalu up-to-date dan sinkron dengan data di Odoo. Keterbatasan pengetahuan mengenai cara melakukan integrasi ini menjadi tantangan tersendiri, karena sering kali praktikan harus mencari referensi dan mempelajari berbagai teknik baru yang belum pernah praktikan hadapi sebelumnya. Hal ini menyebabkan waktu pengerjaan menjadi lebih lama dan membutuhkan upaya lebih dalam mempelajari, mengimplementasikan, dan menguji fitur-fitur yang terintegrasi dengan baik antara kedua platform ini.

2. **Kurangnya litersasi cara penggunaan odoo versi 17**

   Odoo versi 17 memiliki beberapa fitur baru yang berbeda dari versi sebelumnya, baik dari sisi tampilan maupun alur kerja. Perubahan ini mempengaruhi berbagai aspek pengembangan, mulai dari modul yang tersedia hingga pengaturan konfigurasi dan API yang disediakan. Karena praktikan baru pertama kali bekerja dengan Odoo versi 17, praktikan seringkali harus mempelajari alur kerja yang berbeda dari versi sebelumnya, yang belum sepenuhnya praktikan kuasai. Fitur-fitur baru ini juga membutuhkan waktu untuk dipahami, terutama ketika harus dikombinasikan dengan teknologi lain seperti React Native. Dengan

demikian, keterbatasan pemahaman ini cukup menghambat proses pengembangan, karena praktikan perlu mempelajari banyak aspek baru yang tidak hanya melibatkan penggunaan Odoo versi 17 secara umum, tetapi juga teknik-teknik khusus dalam memanfaatkan fitur API untuk mendukung integrasi dengan aplikasi mobile yang sedang praktikan kembangkan.

3.4 Cara Menghadapi Kendala

Praktikan melakukan beberapa upaya untuk mengahadapi kendala dalam pengembangan sistem ini. Beberapa cara yang dilakukan praktikan yaitu.

1. **Mencari referensi dari dokumentasi resmi odoo dan Rect Native**

   Untuk mengatasi keterbatasan literasi terkait integrasi, praktikan mengandalkan dokumentasi resmi yang disediakan oleh Odoo dan React Native. Dokumentasi resmi ini memberikan informasi mendalam mengenai penggunaan API, baik dari cara melakukan request hingga format response yang akan diterima. Di sisi Odoo, dokumentasi membantu praktika memahami modul dan API khusus yang dapat diakses dan bagaimana cara melakukan pengaturan pada backend agar data yang diperlukan dapat diambil dengan tepat. Di sisi React Native, dokumentasi memberikan panduan teknis tentang pengelolaan data dari API dan sinkronisasi data di frontend. Selain itu, contoh-contoh kode yang diberikan dalam dokumentasi sangat membantu praktikan untuk memahami bagaimana data dapat dikelola dan ditampilkan di aplikasi mobile, sehingga praktikan dapat mengikuti praktik terbaik dalam pengembangan aplikasi.

2. **Berdiskusi dengan rekan-rekan di Perusahaan**

   Diskusi dengan rekan-rekan di perusahaan, terutama yang sudah berpengalaman dengan Odoo dan React Native, sangat membantu praktikan dalam menyelesaikan kendala yang dihadapi. Melalui diskusi ini, praktikan mendapatkan wawasan praktis dari pengalaman langsung mereka, termasuk cara mengatasi bug yang mungkin muncul selama proses integrasi. Diskusi ini juga menjadi sarana untuk berbagi pengetahuan dan solusi terbaik dalam melakukan sinkronisasi data, cara menangani error pada API, dan teknik-teknik optimasi lainnya yang

mungkin tidak praktikan temukan dalam dokumentasi atau pelatihan online. Dengan berbagi pengalaman, praktikan dapat memperoleh solusi secara efektif dan cepat, serta meningkatkan kualitas integrasi antara Odoo dan aplikasi mobile yang sedang dikembangkan.