PEMBANGUNAN SISTEM INFORMASI *BOOKING* RUANG RAPAT DAN TRANSPORTASI BERBASIS *MOBILE* PADA PT SISI (SINERGI INFORMATIKA SEMEN INDONESIA)

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat Untuk Menyelesaikan Program Strata-I
Pada Departemen Sistem Informasi Fakultas Teknologi Informasi
Universitas Andalas

Oleh :

Fikran Shadiq Elyafit

2111521024

Pembimbing :

Husnil Kamil, M.T.

198201182008121002

DEPARTEMEN SISTEM INFORMASI

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

PADANG

2025

HALAMAN PENGESAHAN

TUGAS AKHIR

PEMBANGUNAN SISTEM INFORMASI BOOKING RUANG RAPAT DAN TRANSPORTASI BERBASIS MOBILE PADA PT SINERGI INFORMATIKA SEMEN INDONESIA

Oleh:

Fikran Shafiq Elyafi

2111521024

LULUS SIDANG TUGAS AKHIR

24 Oktober 2025

Padang, 24 Oktober 2025

Telah diperiksa dan disetujui oleh

Pembimbing Tugas Akhir

Hanif Ikhwan, M.T.

NIP. 196201252009121002

Mengetahui

Kepala Departemen Sistem Informasi

Ricky Akbar, M. Kom

NIP. 198410062012121003

PERNYATAAN

Saya menyatakan bahwa laporan tugas akhir yang berjudul "Pembangunan Sistem Informasi Booking Ruang Rapat dan Transportasi Berbasis Mobile pada PT Sinergi Informatika Semen Indonesia" ini tidak terdapat karya yang pernah diajukan sebagai syarat menyelesaikan mata kuliah tugas akhir di suatu perguruan tinggi dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat saya yang pernah ditulis atau diterbitkan oleh orang lain kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka

Padang, 05 November 2025

Penulis,

Fikran Shadiq Elyafit

KATA PENGANTAR

Puji dan syukur senantiasa penulis panjatkan kehadirat Allah Swt. atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan Laporan Tugas Akhir yang berjudul "Pembangunan Sistem Informasi *Booking* Ruang Rapat dan Transportasi Berbasis *Mobile* Pada PT SISI (Sinergi Informatika Semen Indonesia)".

Dalam penyusunan tugas akhir ini penulis banyak mendapatkan bantuan dari berbagai pihak, oleh karena itu penulis ingin mengucapkan terima kasih kepada :

1. Bapak Ricky Akbar, M.Kom., selaku Ketua Departemen Sistem Informasi Universitas Andalas.
2. Bapak Husnil Kamil, M.T., selaku dosen pembimbing yang telah membimbing dan memberi pengarahan kepada penulis.
3. Orang tua dan keluarga yang senantiasa memberikan doa, dukungan, dan semangat yang tiada henti.
4. Semua pihak yang tidak dapat disebutkan satu per satu yang turut berkontribusi dalam memberikan saran, bantuan, dan dukungan dalam penyusunan Laporan Tugas Akhir ini.

Penulis menyadari dalam penulisan Laporan Tugas Akhir ini masih terdapat banyak kekurangan dan kesalahan. Oleh karena itu, penulis mengharapkan saran dan kritikan yang membangun dari pembaca yang dapat dikirimkan melalui email: fikranelyafit@gmail.com demi menyempurnakan Laporan Tugas Akhir ini. Semoga Laporan Tugas Akhir ini bermanfaat, baik bagi pembaca maupun bagi penulis.

Padang, 05 November 2025

Penulis,

Fikran Shadiq Elyafit

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

PT Sinergi Informatika Semen Indonesia (SISI) merupakan anak perusahaan dari PT Semen Indonesia (Persero) Tbk yang berdiri pada 9 Juni 2014 dan bergerak di bidang teknologi informasi. Dalam mendukung aktivitas internal, perusahaan memanfaatkan sistem informasi untuk proses booking ruang rapat dan transportasi. Namun, sistem yang digunakan saat ini berbasis web dan memiliki sejumlah keterbatasan, seperti tidak menyediakan sistem notifikasi atau pemberitahuan melalui email, serta tampilan antarmuka yang kurang responsif. Berdasarkan hasil wawancara dengan admin sistem booking pada 31 Oktober 2024, ditemukan kebutuhan akan aplikasi mobile yang dapat memenuhi kebutuhan aktual pengguna, termasuk pengingat serta kemampuan monitoring bagi manajemen. Untuk menjawab kebutuhan tersebut, dilakukan pengembangan sistem informasi baru berbasis mobile untuk mengatasi keterbatasan sistem web saat ini dengan implementasi teknologi mobile yang sesuai dengan kebutuhan perusahaan. Penelitian ini bertujuan untuk membangun Sistem Informasi Booking Ruang Rapat dan Transportasi Berbasis Mobile pada PT SISI yang memungkinkan pengguna melakukan pemesanan secara real-time melalui perangkat mobile dengan pemberitahuan melalui email dan reminder otomatis. Pengembangan sistem dilakukan menggunakan metode waterfall yang terdiri dari beberapa tahapan, yaitu identifikasi masalah, pengumpulan data, analisis kebutuhan, perancangan sistem, implementasi, dan pengujian. Pengujian sistem menggunakan metode black box testing untuk memastikan setiap fungsi berjalan sesuai dengan kebutuhan pengguna. Hasil penelitian ini telah menghasilkan sistem yang sesuai rancangan, dengan seluruh fitur utama berfungsi baik berdasarkan pengujian. Persentase penerimaan pengguna dari hasil UAT rata-rata 92.11% membuktikan sistem ini mampu memenuhi kebutuhan booking ruang rapat dan transportasi di PT SISI.

Kata Kunci: Sistem Informasi, Booking, Real-time, Transportasi, React Native

BAB I
PENDAHULUAN

Bab ini berisi penjelasan tentang latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, serta sistematika penulisan pada proposal tugas akhir ini.

1.1. Latar Belakang

PT Sinergi Informatika Semen Indonesia (SISI), yang didirikan pada 9 Juni 2014 sebagai anak usaha PT Semen Indonesia (Persero) Tbk, merupakan perusahaan teknologi informasi yang berkomitmen menjadi pionir dalam penerapan solusi digital untuk meningkatkan efektivitas dan koordinasi antar unit. Dengan jumlah karyawan berkisar 200-500 orang yang terdiri dari karyawan tetap, kontrak, freelance, serta tenaga kerja dengan sistem remote dan hybrid guna menyesuaikan kebutuhan proyek digital yang dijalankan, SISI berperan strategis sebagai penggerak utama pengembangan dan dukungan operasional teknologi informasi dan komunikasi (ICT) untuk Semen Indonesia Group. Perusahaan ini tidak hanya menyediakan layanan *Shared Services*, yaitu model di mana fungsi-fungsi pendukung bisnis (seperti HR, keuangan, dan IT) dikelola secara terpusat untuk meningkatkan efisiensi, tetapi juga senantiasa berinovasi dalam pengelolaan keberlanjutan dan memperluas dampaknya di sektor BUMN dan industri lainnya. Dalam upaya meningkatkan kemudahan dan kelancaran operasional, PT SISI memandang pentingnya solusi teknologi untuk mengelola kebutuhan pemesanan ruang rapat dan layanan transportasi. Sistem berbasis web yang ada saat ini belum memadai untuk mendukung fleksibilitas pengguna, terutama karyawan yang menggunakan perangkat *mobile*. Proses yang lambat dan ketergantungan pada website yang masih memiliki beberapa kekurangan membuat pemesanan jadi sulit dan berpotensi menyebabkan bentrok jadwal. Pengguna juga tidak bisa melihat ketersediaan ruangan atau transportasi yang ingin dipesan pada tanggal dan waktu tertentu. Selain itu, sistem ini juga tidak mampu mengakomodasi kebutuhan akan pemberitahuan melalui email, *reminder booking*, serta melakukan *reschedule*.

Selanjutnya, berdasarkan wawancara yang dilakukan pada 31 Oktober 2024 dengan Inka Cut Nurbaiti selaku admin sistem *booking*, ditemukan beberapa masalah utama yang menjadi perhatian. Pertama, sistem yang ada saat ini memiliki keterbatasan dalam hal pemesanan *real-time* dan belum dilengkapi dengan sistem notifkasi atau pemberitahuan melalui email. Akibatnya, pengguna sering kesulitan mendapatkan informasi terkini terkait status pemesanan mereka. Selain itu, aplikasi yang kurang responsif pada perangkat *mobile* menyulitkan karyawan yang ingin melakukan pemesanan melalui ponsel, sehingga kenyamanan pengguna saat melakukan pemesanan menjadi terganggu. Di sisi lain, manajemen juga menghadapi kendala dalam memantau dan menganalisis data penggunaan fasilitas. Hasil wawancara tersebut mengidentifikasi kebutuhan akan pembangunan aplikasi mobile dan web manajemen yang responsif serta dilengkapi dengan fitur pemberitahuan melalui email, guna meningkatkan kemudahan akses dan pengalaman pengguna dalam melakukan pengelolaan data secara real time. Selain itu, manajemen juga memerlukan fitur *dashboard* yang dapat menyajikan data penggunaan ruang rapat dan layanan transportasi secara r*eal-time*, sehingga dapat mempermudah pemantauan dan analisis. Penambahan opsi ekspor data ke format Excel juga diperlukan untuk mendukung pembuatan laporan berkala serta dokumentasi yang lebih rapi dan terstruktur.

Dalam mendukung penelitian ini, beberapa studi sebelumnya dapat dijadikan referensi. Pertama, penelitian oleh Rachmaniar et al., (2024) berjudul "*Mobile Application For Reservation Of Meeting Rooms and Event Spaces at Marquee Executive Offices*" menunjukkan bahwa pengembangan aplikasi *mobile* terbukti mempercepat proses reservasi. Aplikasi ini memungkinkan pengguna untuk melihat ketersediaan ruang secara *real-time*, memesan langsung melalui perangkat *mobile*, serta menerima konfirmasi instan. Hasil penelitian ini menunjukkan bahwa penerapan teknologi *mobile* pada sistem reservasi ruang dapat mengoptimalkan penggunaan ruang, mengurangi risiko kesalahan manusia, dan memperkuat citra perusahaan sebagai penyedia layanan yang modern dan unggul. Kedua, penelitian oleh Ikorasaki et al., (2024) berjudul "Aplikasi Pemesanan Ruangan Panti Jompo Yayasan Karya Asih Berbasis Android" menunjukkan bahwa pengembangan aplikasi berbasis Android ini memudahkan

keluarga dalam memilih dan memesan ruangan di panti jompo. Aplikasi ini menyediakan beragam pilihan kelas ruangan dengan fasilitas berbeda, mulai dari VIP hingga standard, sehingga pengguna dapat menyesuaikan pilihan berdasarkan kebutuhan dan anggaran mereka. Sistem ini tidak hanya menghemat waktu dan tenaga pengguna, tetapi juga memperbaiki kualitas layanan bagi pengelola panti jompo.

Berdasarkan berbagai hasil studi dan kondisi sistem yang sedang berjalan di PT SISI, dapat disimpulkan bahwa perusahaan memerlukan sebuah sistem baru yang mampu mengatasi keterbatasan sistem web saat ini. Sistem tersebut perlu mendukung akses yang lebih fleksibel melalui perangkat *mobile*, memungkinkan proses pemesanan ruang rapat dan transportasi secara *real-time*, memberikan pemberitahuan melalui email, serta mengirimkan *reminder* untuk setiap jadwal pemesanan. Selain itu, dibutuhkan pula kemudahan bagi pengguna untuk memantau status booking secara langsung melalui aplikasi. Di sisi lain, pihak manajemen membutuhkan sistem *back office* berbasis web untuk mengelola dan memantau penggunaan fasilitas, lengkap dengan fitur analitik, pelaporan, serta kemampuan filterisasi data. Untuk itu, perlu dibangun Sistem Informasi *Booking* Ruang Rapat dan Transportasi berbasis *mobile* yang dirancang sesuai dengan kebutuhan pengguna dan manajemen PT SISI. Sistem ini diharapkan dapat menjadi solusi yang tepat dalam mendukung proses pemesanan serta pengelolaan fasilitas secara menyeluruh. Pernyataan kebutuhan ini menjadi dasar dari rumusan masalah dalam penelitian ini, yaitu bagaimana membangun Sistem Informasi *Booking* Ruang Rapat dan Transportasi Berbasis *Mobile* pada PT SISI.

Berdasarkan latar belakang tersebut, penelitian ini bertujuan untuk merancang dan membangun aplikasi *mobile* yang dapat memenuhi kebutuhan pemesanan ruang rapat dan transportasi dengan cara yang lebih praktis. Dengan adanya sistem ini, diharapkan PT SISI memiliki solusi yang mendukung pengelolaan fasilitas secara lebih baik dan meningkatkan kinerja operasional perusahaan. Penelitian dengan judul "PEMBANGUNAN SISTEM INFORMASI *BOOKING* RUANG RAPAT DAN TRANSPORTASI BERBASIS *MOBILE* PADA PT SISI (SINERGI INFORMATIKA SEMEN INDONESIA)" ini

diharapkan dapat menghadirkan solusi yang efektif dalam mengatasi permasalahan yang dihadapi perusahaan.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan sebelumnya, rumusan masalah pada penelitian ini adalah bagaimana membangun Sistem Informasi *Booking* Ruang Rapat dan Transportasi Berbasis *Mobile* Pada PT SISI.

1.3. Batasan Masalah

Agar penelitian tidak menyimpang terlalu jauh dan ruang lingkup sistem yang akan dibangun tidak terlalu luas, maka dari masalah yang ada ditentukan batasan-batasan masalah sebagai berikut.

1. Sistem yang dikembangkan hanya mencakup fitur *booking* ruang rapat, *booking* transportasi, *reschedule booking*, *cancel booking,* pemberitahuan melalui email, pengelolaan fasilitas dan ekspor data *booking* ke excel.
2. Teknologi yang digunakan meliputi React Native untuk aplikasi mobile (hanya untuk platform Android), Next.js untuk web admin, nodemailer untuk pemberitahuan melalui email dan reminder, serta MySQL sebagai database.
3. Sistem hanya untuk penggunaan internal PT Sinergi Informatika Semen Indonesia (SISI) dan tidak terintegrasi dengan sistem keuangan perusahaan.
4. Pengujian sistem dilakukan dengan metode *black box* testing untuk memastikan fitur berjalan sesuai kebutuhan pengguna.

1.4. Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah untuk membangun dan mengimplementasikan Sistem Informasi Booking Ruang Rapat dan Transportasi Berbasis Mobile yang fungsionalitas nya sesuai kebutuhan dan keinginan pengguna di PT Sinergi Informatika Semen Indonesia, sehingga mampu menyelesaikan permasalahan booking ruang rapat dan transportasi di perusahaan tersebut.

1.5. Manfaat Penelitian

Adapun manfaat dari penelitian tugas akhir ini, yaitu:

1. Memudahkan PT SISI dalam monitoring dan pengelolaan pemesanan ruang rapat dan transportasi secara *real-time* serta menghasilkan laporan penggunaan sumber daya yang akurat dan terintegrasi.
2. Memberikan kemudahan bagi karyawan untuk melakukan pemesanan ruang rapat dan transportasi melalui aplikasi *mobile* dengan informasi fasilitas yang lengkap.
3. Membantu admin dalam mengelola jadwal serta melakukan verifikasi pemesanan dengan dukungan pemberitahuan melalui email yang meningkatkan kelancaran operasional.
4. Membantu manajemen dalam mengambil keputusan strategis terkait penggunaan sumber daya berdasarkan data analitik yang tersedia pada dashboard monitoring, sehingga dapat meningkatkan optimalisasi pengelolaan fasilitas perusahaan.

1.6. Sistematika Penelitian

Sistematika penulisan laporan tugas akhir ini dibagi menjadi tiga bab yaitu sebagai berikut:

BAB I : PENDAHULUAN

Bab ini berisi tentang latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan laporan.

BAB II : TINJAUAN PUSTAKA

Bab ini berisi tentang landasan teori dan informasi pendukung yang relevan berkaitan dengan penelitian ini.

BAB III : METODOLOGI PENELITIAN

Bab ini berisi tentang objek penelitian, metode pengumpulan data dan *flowchart* penelitian.

BAB IV : ANALISIS DAN PERANCANGAN SISTEM

Bab ini berisi tentang analisis sistem yang berjalan, kebutuhan fungsional dan perancangan pada sistem usulan untuk menjawab permasalahan pada sistem lama yang digambarkan melalui diagram dan tools pendukung lainnya.

BAB V : IMPLEMENTASI DAN PENGUJIAN

Bab ini berisi tentang implementasi dari hasil analisis dan perancangan sistem yang telah dijabarkan pada bab sebelumnya. Pada bab ini juga akan dijelaskan pengujian yang dilakukan terhadap sistem usulan berdasarkan rancangan yang telah dibuat pada bab sebelumnya.

BAB VI : PENUTUP

Bab ini berisi tentang kesimpulan dari penelitian dan saran terhadap pengembangan kedepannya.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan tentang latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, serta sistematika penulisan pada proposal tugas akhir ini.

2.1. Struktur Organisasi Perusahaan

PT Sinergi Informatika Semen Indonesia (PT SISI) memiliki *Top Management* yang dipimpin oleh Bapak Achmad Tholchah, yang menjabat sebagai Direktur Utama sejak tahun 2022. Berbekal pengalaman luas di bidang teknologi dan strategi bisnis, beliau terus membawa PT SISI tumbuh dan berkontribusi dalam mendorong transformasi digital di Indonesia. Beliau menyelesaikan pendidikan Sarjana Teknik Mesin di Institut Teknologi Bandung (ITB) dan meraih gelar Magister Manajemen Strategis dari Prasetya Mulya pada tahun 2014. Dalam perjalanan kariernya, berbagai posisi strategis pernah beliau jalani, seperti *Vice President Digital Solution* di PT SISI dan *General Manager IT* di PT Semen Indonesia (Persero) Tbk, serta turut terlibat dalam penyusunan regulasi di bidang teknologi informasi di lingkungan Kementerian BUMN. Kepemimpinan beliau yang berlandaskan kejujuran, kepercayaan, dan ketekunan mengantarkan dirinya meraih sejumlah penghargaan bergengsi, antara lain *TOP DIGITAL Awards* 2024 dan Indonesia *Digital Innovation and Achievement Awards* 2024, sebagaimana dijelaskan oleh PT Sinergi Informatika Semen Indonesia (2024).

Di bawah kepemimpinan Direktur, PT Sinergi Informatika Semen Indonesia memiliki struktur organisasi yang terdiri atas beberapa departemen utama yang berfokus pada pengembangan solusi digital, layanan teknologi informasi, serta dukungan operasional bagi perusahaan induk dan anak usaha di lingkungan Semen Indonesia Group. Struktur organisasi ini dirancang untuk mendukung efisiensi koordinasi, efektivitas pelaksanaan proyek, serta peningkatan kualitas layanan berbasis teknologi informasi.

Masing-masing departemen memiliki fungsi dan tanggung jawab yang berbeda sesuai bidang keahliannya. *Department of Corporate Services* terdiri dari

Unit of HC & Procurement serta *Unit of Research System/App*, yang didukung oleh *Section of Corporate Performance/HR. Department of Sales & Marketing* menaungi *Unit of Solution Architect* dan *Unit of Sales*, dengan dukungan *Section of Marketing & Partnership*. Sementara itu, *Department of Service Delivery* mencakup *Unit of IT Service Centre*, *Unit of Business Operation Service*, *Unit of System Integration, serta Unit of R&D Innovation Business*, yang didukung oleh *Section of Service Delivery Quality Assurance*.

Selain itu, terdapat fungsi Internal Audit yang berada langsung di bawah CEO untuk memastikan efektivitas pengendalian internal dan penerapan tata kelola perusahaan yang baik.

Struktur organisasi PT SISI secara lengkap dapat dilihat pada Gambar 2.1 berikut.

Gambar 2.1 Struktur Organisasi PT SISI (Data Internal PT SISI, 2025)

2.2. Sistem Informasi

Menurut Rasefta dan Esabella (2020), sistem adalah kumpulan komponen yang saling terkait satu dengan yang lainnya untuk mencapai suatu tujuan tertentu. Komponen-komponen ini saling berinteraksi dan berkoordinasi untuk melaksanakan kegiatan pokok dalam mencapai tujuan. Sedangkan informasi adalah data mentah yang telah diolah sedemikian rupa sehingga menghasilkan sesuatu yang bermakna bagi penggunanya dalam mengambil sebuah keputusan. Sistem informasi merupakan rangkaian prosedur formal dimana data dikelompokkan, diproses menjadi informasi, dan didistribusikan kepada pengguna Rasefta dan Esabella, (2020). Sistem informasi terdiri dari beberapa komponen

yang saling terkait yaitu komponen input, model, output, teknologi, basis data dan control.

Menurut Damayanti et al., (2021), sistem informasi adalah suatu sistem didalam suatu organisasi yang mempertemukan kebutuhan pengelolaan transaksi harian, mendukung operasi, bersifat manajerial, dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang dibutuhkan. Rasefta dan Esabella (2020), mendefinisikan sistem informasi akademik sebagai suatu sistem yang digunakan untuk mengelola informasi dan data-data akademik sekolah berupa data siswa, data guru, jadwal pelajaran, dan data penilaian. Sehingga semua informasi tersebut dapat diakses oleh guru dan siswa dalam satu sistem yaitu Sistem Informasi Akademik Berbasis Web dengan cara yang mudah dan cepat.

Berdasarkan uraian diatas maka didapatkan kesimpulan bahwa sistem informasi adalah sebuah kombinasi dari teknologi, data, orang, dan proses yang terintegrasi dalam organisasi untuk mengumpulkan, mengolah, menyimpan, dan menyebarkan informasi. Sistem ini mendukung pengambilan keputusan, operasional harian, manajemen, sehingga meningkatkan kemudahan bagi organisasi dalam mencapai tujuan bersama.

2.3. *Booking* Ruang Rapat

Studi terdahulu menjelaskan bahwa penelitian mengenai sistem reservasi telah banyak dilakukan untuk meningkatkan kelancaran dan meminimalkan kesalahan manusiawi dalam pengelolaan ruang rapat. Menurut Marianko (2020), sistem reservasi ruang rapat memungkinkan pengguna untuk melakukan pemesanan berdasarkan kebutuhan peserta, tipe pertemuan, dan fasilitas yang diperlukan. Menurut Rachmaniar (2024), menambahkan bahwa penggunaan aplikasi mobile memberikan kemudahan bagi pengguna untuk melihat ketersediaan ruang, melakukan reservasi secara langsung, dan menerima konfirmasi instan, yang dapat mengurangi potensi konflik jadwal. Hal ini sangat relevan dengan permasalahan yang dihadapi PT SISI, di mana sistem yang ada saat ini tidak mampu menyediakan informasi ketersediaan ruang secara *real-time*, sehingga karyawan sering kali kesulitan dalam memastikan apakah ruang rapat yang diinginkan tersedia pada waktu yang diinginkan.

Sistem *Booking Meeting Room* merupakan bagian dari teknologi reservasi modern yang didukung oleh teori efisiensi operasional. Teknologi dalam transportasi dan reservasi ruang rapat mampu mempermudah proses manajemen fasilitas dengan mengintegrasikan data secara *real-time*. Sebagai contoh, penerapan teknologi ini akan sangat bermanfaat bagi PT SISI yang saat ini masih menghadapi kendala dalam pemesanan ruang rapat yang belum dilengkapi dengan sistem pemberitahuan melalui email dan kurang responsif di perangkat *mobile*. Proses yang lambat dan ketergantungan pada website yang tidak optimal membuat sistem pemesanan saat ini kurang praktis, berpotensi menyebabkan bentrok jadwal dan mempersulit pengguna dalam melakukan pemesanan.

Dari uraian di atas, pembuatan aplikasi untuk perangkat *mobile* akan memberikan jawaban untuk semua masalah yang dihadapi PT SISI. Sistem ini tidak hanya membuat karyawan bisa memesan ruangan sesuai kebutuhan seperti jumlah orang dan perlengkapan yang dibutuhkan, tapi juga menunjukkan info ketersediaan ruangan secara langsung, pemberitahuan cepat, dan bisa melakukan pemesanan dari mana saja. Dengan memakai teknologi ini, PT SISI bisa mengatasi masalah bentrok jadwal, mempercepat proses pemesanan, dan memanfaatkan dengan baik fasilitas ruang rapat dan transportasi. Penggunaan aplikasi pada perangkat *mobile* juga akan memberi kemudahan bagi karyawan dan mengurangi keharusan menggunakan komputer atau laptop, sesuai dengan kebutuhan bekerja di mana saja dalam lingkungan kerja sekarang.

2.4. Transportasi

Transportasi merupakan bagian penting dalam kegiatan operasional perusahaan yang mendukung perpindahan sumber daya dan karyawan. Sasmito (2022), menjelaskan bahwa masalah utama dalam pengelolaan transportasi terutama terdapat pada sistem pemesanan dan penggabungan data transportasi. Dalam penelitiannya, berbagai jenis kendaraan seperti *truck* (engkel, diesel, tronton, container, truck gandeng), pick up, dan motor roda tiga dibuat berbeda sesuai dengan keperluan tertentu. Proses pengangkutan sering menghadapi kesulitan terutama dalam pengelolaan data transportasi yang belum terhubung satu sama lain, seperti jumlah dan identitas pemilik usaha alat transportasi, jenis

dan jumlah alat transportasi. Keadaan ini menjadi salah satu kendala dalam penggunaan transportasi yang baik pada kegiatan perusahaan.

Sistem pemesanan transportasi yang menggunakan teknologi informasi hadir sebagai jawaban untuk masalah pengelolaan transportasi. Aplikasi *Transcorp* yang dibuat Sasmito (2022), memperlihatkan bahwa penggunaan sistem pemesanan transportasi berbasis web dapat memperbaiki kualitas transaksi data dengan hasil pengujian mencapai 92,5%. Sistem pemesanan transportasi yang terhubung memungkinkan pengaksesan informasi secara langsung tentang ketersediaan kendaraan dan memudahkan proses *booking*. Hal ini membantu penggunaan sumber daya transportasi dengan lebih baik dan meningkatkan mutu layanan. Pembuatan sistem pemesanan transportasi menggunakan cara Prototyping terbukti berhasil dengan hasil uji pengguna mencapai 90% (sangat baik) berdasarkan *Usability Testin*g, yang menunjukkan tingginya kepuasan pengguna terhadap sistem pemesanan transportasi yang dibuat.

2.5. Alat Pemodelan Sistem

Alat pemodelan sistem berfungsi untuk menganalisis dan merancang sistem yang akan dikembangkan. Dalam laporan ini, *Business Process Modeling Notation* (BPMN) digunakan untuk menggambarkan logika dan aliran proses bisnis, sedangkan *Unified Modeling Language* (UML) dipakai untuk memvisualisasikan perangkat lunak. Elemen UML yang digunakan meliputi *use case diagram, use case scenario* dan *sequence diagram.*

2.4.1. *Business Process Modelling Notation (BPMN)*

Business Process Modeling Notation (BPMN) adalah standar grafis yang digunakan untuk menggambarkan alur proses bisnis secara jelas dan terstruktur. BPMN memungkinkan pemodelan proses bisnis dengan simbol-simbol yang merepresentasikan berbagai elemen dalam suatu proses, seperti event, activity, dan gateway. Dengan menggunakan BPMN, organisasi dapat menggambarkan urutan kerja, pengambilan keputusan, serta interaksi antar pihak yang terlibat dalam proses bisnis. Diagram BPMN terdiri dari elemen-elemen penting, termasuk Flow Objects, Connecting Objects, Swimlanes, dan Artifacts, yang

masing-masing memiliki fungsi spesifik untuk mendefinisikan alur dan hubungan antar aktivitas bisnis (Novian Et al., 2022).

Penerapan BPMN memungkinkan organisasi untuk memvisualisasikan dan memetakan proses bisnis yang kompleks, yang sangat penting untuk mengidentifikasi ketidakefisienan serta potensi perbaikan dalam alur kerja. Dalam penelitian ini, BPMN digunakan untuk menganalisis dan merancang ulang alur proses bisnis, dengan tujuan untuk meningkatkan efisiensi dan mengurangi kesalahan yang sering terjadi pada sistem yang ada. Dengan pendekatan BPMN, pemahaman antar stakeholder tentang alur proses menjadi lebih jelas, dan memungkinkan tercapainya optimasi sistem yang lebih terstandarisasi serta peningkatan kualitas pelayanan yang lebih baik (Supit et al., 2021).

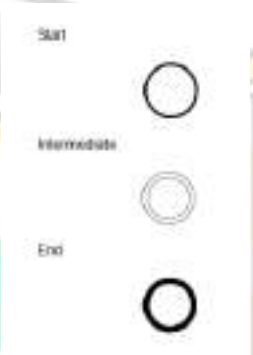
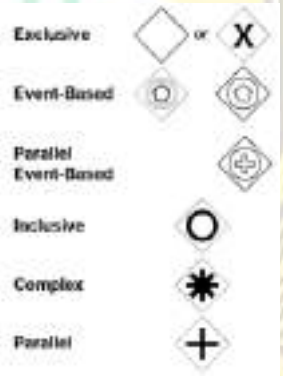
Diagram BPMN terdiri dari beberapa elemen kunci yang dikelompokkan ke dalam empat kategori utama:

1. *Flow Objects*: Ini adalah elemen inti dalam diagram BPMN. Mereka meliputi:
    - *Event*: Mewakili sesuatu yang terjadi selama proses (misalnya, *event* mulai, intermediate, dan selesai) (Supit et al., 2021).
    - *Activity*: Mewakili pekerjaan atau tugas yang harus dilakukan (misalnya, task, sub-proses).
    - *Gateway*: Mewakili titik keputusan yang mengendalikan alur berdasarkan kondisi tertentu.
2. *Connecting Objects*: Menunjukkan hubungan antar *flow objects*.
    - *Sequence Flow*: Mewakili alur langkah-langkah proses.
    - *Message Flow*: Menunjukkan komunikasi antar proses.
    - *Association*: Digunakan untuk menghubungkan artifact dengan *flow objects.*
3. *Swimlanes*: Digunakan untuk mengorganisasi elemen-elemen dalam diagram BPMN ke dalam kategori, seperti *Pools* dan *Lanes*, untuk memisahkan peran atau departemen yang bertanggung jawab atas berbagai aktivitas dalam proses.

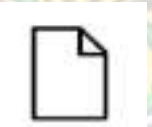
4. *Artifacts*: Elemen ini memberikan informasi tambahan dan konteks pada diagram.
    - *Data Objects*: Mewakili data yang diperlukan atau dihasilkan oleh aktivitas.
    - *Groups*: Digunakan untuk mengelompokkan aktivitas tanpa mempengaruhi alur.
    - *Annotations*: Memberikan informasi tambahan agar diagram lebih mudah dipahami.

Berikut ini adalah simbol-simbol BPMN, seperti terlihat pada Tabel 2.1 di bawah ini:

Tabel 2.1 Elemen BPMN (Irfan et al., 2023)

Elemen	Nama Notasi	Notasi
Flow Objects	Event	
	Activities	
Flow Objects	Gateway	
Connecting Objects	Sequence Flow	
	Message Flow	
	Association	

Tabel 2.1 Simbol Use case Diagram (Irfan et al., 2023) (Lanjutan)

Elemen	Nama Notasi	Notasi
Connecting Objects	Sequence Flow	
	Message Flow	
	Association	
Swimlanes	Pool	
	Lane	
Artifacts	Annotation	
	Group	
Artifacts	Data Object	
	Data Store	

2.4.2. *Use Case Diagram*

Use Case Diagram adalah diagram yang menggambarkan interaksi antara pengguna (aktor) dengan sistem yang sedang dikembangkan melalui serangkaian skenario. Diagram ini fokus pada mendeskripsikan fungsionalitas yang harus dipenuhi sistem, bukan bagaimana sistem bekerja. *Use Case Diagram* terdiri dari aktor dan use case, di mana aktor bisa berupa manusia, sistem lain, atau perangkat keras yang berinteraksi dengan sistem yang sedang dianalisis (Arifina & Siahaan, 2020). Aktor mewakili entitas eksternal, sementara *use case* menggambarkan layanan yang disediakan oleh sistem. Hubungan antara aktor dan *use case* digambarkan dengan asosiasi, yang menunjukkan interaksi yang terjadi. *Use Case* Diagram memberikan gambaran jelas tentang kebutuhan dan interaksi dalam sistem (Arifina & Siahaan, 2020). Wayahdi & Ruziq (2023) menambahkan bahwa diagram ini juga sangat berguna dalam merancang sistem dari perspektif pengguna, serta memudahkan komunikasi antara pengembang dan pihak-pihak terkait dalam pemodelan sistem berbasis UML.

Dalam konteks sistem *booking* ruang rapat dan transportasi, Use Case Diagram menggambarkan alur pemesanan yang dilakukan oleh karyawan, mulai dari awal proses *booking* hingga konfirmasi *booking*. Diagram ini membantu memastikan sistem dapat memenuhi kebutuhan pengguna dengan jelas, serta memberikan gambaran yang lebih baik tentang interaksi dalam proses pemesanan (Wayahdi & Ruziq, 2023). Berikut ini adalah simbol-simbol *Use case Diagram*, seperti terlihat pada Tabel 2.2 di bawah ini:

Tabel 2.2 Simbol Use case Diagram (Irfan et al., 2023)

Simbol	Penjelasan
Aktor 	Aktor: mewakili peran orang, sistem yang lain, atau alat ketika berkomunikasi dengan use case
Use case 	Use case: abstraksi atau interaksi antara system dengan aktor

Tabel 2. 2 Simbol Use case Diagram (Irfan et al., 2023) (Lanjutan)

Association 	Association: abstraksi dari penghubung antara aktor dengan use case
Generalisasi 	Generalisasi: menunjukkan spesialisasi aktor untuk dapat berpartisipasi dengan *use case*
Includes <<Include>> 	*Includes*: menunjukkan bahwa suatu *use case* seluruhnya merupakan fungsionalitas dari *use case* lainnya
Extends <<extend>> 	*Extends*: menunjukkan bahwa suatu *use case* merupakan tambahan fungsional dari *use case* lainnya jika suatu kondisi terpenuhi

2.4.3. *Use Case Scenario*

Use case scenario adalah deskripsi tekstual yang merinci interaksi antara aktor dan sistem untuk mencapai tujuan tertentu, dan ini merupakan elemen utama dalam pemodelan use case yang bahkan lebih penting daripada diagramnya. Skenario ini mendeskripsikan urutan langkah atau aksi yang dilakukan aktor, mencakup alur yang berhasil maupun yang gagal. Penulisannya dapat disajikan dalam format singkat (brief), informal (casual), atau lengkap (fully dressed), dimana format lengkap adalah yang paling detail dan sering digunakan dalam praktik. Format lengkap ini mencakup beberapa bagian penting: aktor primer yang memulai interaksi, prakondisi sebagai syarat sebelum use case dimulai, alur utama yang mengarah pada skenario sukses, alur alternatif untuk penanganan pilihan atau kegagalan, dan kondisi akhir yang merupakan keadaan setelah use case berhasil dijalankan.

Menurut Martin (2023) Use case scenario digunakan untuk menjelaskan sebuah interaksi yang terjadi antara aktor dan sistem. Selain itu, use case scenario dijelaskan secara tekstual sesuai dengan kebutuhannya, yaitu singkat, informal dan juga lengkap.

2.4.4. *Sequence Diagram*

Menurut (Fintri Indriyani et al., 2019), *Sequence Diagram* merupakan alat pemodelan yang menggambarkan alur pesan (*message*) antar objek dalam sebuah *use case* dari waktu ke waktu. Diagram ini memvisualisasikan seluruh objek yang terlibat dalam suatu proses, seperti interaksi antara pengguna (*actor*) dan sistem, serta urutan langkah yang terjadi selama eksekusi fungsi tertentu. Contohnya, dalam sistem *booking* ruangan, *Sequence Diagram* dapat menunjukkan bagaimana pengguna mengirim permintaan *booking*, sistem memvalidasi ketersediaan ruangan, hingga mengonfirmasi status reservasi. Pendekatan ini membantu mengidentifikasi kebutuhan fungsional sistem secara detail dan memastikan alur proses terdefinisi dengan baik (Fintri Indriyani et al., 2019).

(Sinambela et al., 2024) menambahkan bahwa *Sequence Diagram* tidak hanya merepresentasikan alur pesan, tetapi juga menekankan kolaborasi dinamis antar objek dalam sistem. Diagram ini memperlihatkan bagaimana objek-objek saling berinteraksi melalui pertukaran pesan, termasuk kondisi atau respons yang mungkin terjadi. Misalnya, dalam konteks *booking* ruang rapat, *Sequence Diagram* dapat menggambarkan interaksi antara pengguna, sistem *booking*, dan admin, seperti pengiriman pemberitahuan melalui email konfirmasi *booking*, hingga *reminder*. Kolaborasi ini memastikan sistem dapat menangani skenario kompleks, seperti pembatalan atau perubahan jadwal (Sinambela et al., 2024).

Kedua pandangan di atas menunjukkan bahwa *Sequence Diagram* sangat penting untuk menggambarkan urutan komunikasi dan hubungan antar bagian dalam sistem. Untuk sistem pemesanan ruang rapat dan transportasi, penggunaan *Sequence Diagram* membantu menjelaskan proses pemesanan dari awal hingga akhir, mulai dari pengguna memasukkan data, sistem mengecek ketersediaan ruangan atau kendaraan, sampai data disimpan. Diagram ini juga menunjukkan bagaimana bagian-bagian sistem seperti menu pemesanan, pengguna, dan admin saling berkomunikasi, serta bagaimana sistem menangani situasi seperti pembatalan, perubahan jadwal, atau bentrok pemesanan. Dengan menggunakan *Sequence Diagram*, rancangan sistem menjadi lebih jelas, mengurangi kemungkinan kesalahan, dan memastikan semua kebutuhan sistem terpenuhi.

2.6. *Tools* Pengembangan

Dalam membangun aplikasi atau sistem, pemilihan *tools* pengembangan memainkan peran krusial dalam keberhasilan proyek. *Tools* pengembangan adalah perangkat lunak dan teknologi yang digunakan untuk membuat, menguji, dan memelihara aplikasi (Telkom University, 2024). Untuk proyek Sistem *Booking* Ruang Rapat dan Transportasi di PT SISI, beberapa *tools* yang digunakan adalah TypeScript sebagai bahasa pemrograman dengan fitur tipe data statis, Express.js sebagai *framework backend* untuk membangun API, Next.js sebagai *framework frontend* untuk pengembangan web, React Native untuk mengembangkan aplikasi *mobile* yang berjalan di berbagai platform, Postman untuk pengujian API, dan *Visual Studio Code* (VSCode) sebagai editor kode untuk menulis dan mengelola kode program. Kombinasi teknologi ini menciptakan ekosistem pengembangan yang mudah digunakan dan modern untuk menghubungkan berbagai komponen sistem.

2.5.1. *React Native*

React Native adalah framework JavaScript open-source yang dikembangkan oleh Facebook untuk membangun aplikasi mobile yang dapat berjalan di platform Android dan iOS. Framework ini memungkinkan pengembang menggunakan basis kode yang sama untuk berbagai platform, sehingga mengurangi waktu dan biaya pengembangan. React Native menggunakan konsep Virtual DOM, yang membuat pengolahan elemen UI lebih cepat. JSX, sebagai bagian dari React Native, memberikan sintaks intuitif untuk mendeskripsikan tampilan antarmuka melalui kombinasi HTML dan JavaScript, sehingga mempermudah pengembangan aplikasi modern (Hielmi Sulaeman & Anita Fira Waluyo, 2023).

React Native menggunakan komponen native yang langsung mengakses antarmuka sistem operasi. Hal ini memberikan pengalaman pengguna yang hampir setara dengan aplikasi native, berbeda dengan teknologi berbasis web view yang memiliki keterbatasan dalam performa dan fleksibilitas. React Native terbukti efektif digunakan dalam berbagai aplikasi, seperti pengelolaan keuangan berbasis mobile yang mendukung pengambilan keputusan finansial real-time melalui integrasi API dan pengelolaan database.

Framework ini juga diterapkan dalam aplikasi pemesanan hotel berbasis mobile, di mana fitur navigasi modern dan modularitasnya mempermudah manajemen pemesanan kamar. React Native memungkinkan integrasi gateway pembayaran dan pelacakan data real-time, sehingga meningkatkan efisiensi operasional aplikasi (R. Karthikeyan & S. Karthick, 2023).

Dibandingkan Flutter, React Native memiliki beberapa keunggulan utama. Pertama, komunitas React Native lebih besar, sehingga menyediakan banyak pustaka dan plugin tambahan yang mempercepat pengembangan. Kedua, React Native menggunakan JavaScript, bahasa pemrograman yang lebih dikenal dibandingkan Dart yang digunakan oleh Flutter. Selain itu, React Native memberikan fleksibilitas lebih besar karena mendukung integrasi dengan kode native jika diperlukan, sedangkan Flutter lebih terfokus pada rendering UI dengan mesin khusus yang dapat menyebabkan aplikasi menjadi lebih besar.

React Native terus berkembang sebagai solusi lintas platform yang populer. Keunggulannya terletak pada kemampuannya untuk menyediakan pengalaman pengguna yang optimal dengan waktu pengembangan yang lebih singkat. Studi terdahulu membuktikan bahwa framework ini mampu meningkatkan efisiensi dan kualitas aplikasi mobile di berbagai bidang, seperti sistem manajemen keuangan dan pemesanan hotel (Kausar Bagwan & Swati Ghule, 2019).

2.5.2. *MySQL*

MySQL adalah sistem manajemen basis data yang sering disingkat sebagai DBMS dan bersifat open-source. MySQL dapat digunakan untuk membuat dan mengelola basis data beserta isinya, mulai dari yang terkecil hingga yang sangat besar, serta menyampaikan informasi kepada penggunanya. MySQL juga termasuk dalam RDBMS atau *Relational Database Management System*, di mana struktur basis datanya memungkinkan proses pengambilan data menggunakan *metode relational database* dan menjadi penghubung antara perangkat lunak dan server basis data. Yang perlu diingat adalah MySQL dapat digunakan secara gratis (Wahyudi et al., 2022).

Ramadhan dan Mukhaiyar (2020) menjelaskan bahwa MySQL dipilih karena memiliki beberapa keunggulan, di antaranya adalah bersifat open source yang artinya dapat digunakan secara gratis, memiliki keamanan yang tinggi,

mendukung Multi User yang artinya dapat digunakan oleh beberapa pengguna secara bersamaan, dan memiliki fleksibilitas antarmuka (*interface*) pada beberapa aplikasi dan bahasa pemrograman.

Berdasarkan uraian di atas, dapat disimpulkan bahwa MySQL adalah sistem manajemen basis data relasional yang bersifat *open-source,* memungkinkan pengguna untuk membuat, mengelola, dan memodifikasi data dengan mudah dan cepat. MySQL mendukung berbagai bahasa pemrograman dan sistem operasi, sehingga menjadikannya pilihan populer bagi pengembang dalam membangun aplikasi berbasis data, termasuk untuk pengembangan aplikasi pemesanan (*booking*) seperti sistem *booking* ruang rapat dan transportasi di PT SISI. Dengan keunggulan MySQL dalam hal keamanan data dan kemampuan *multi-user*, basis data ini sangat cocok untuk mengelola informasi pemesanan ruang rapat dan kendaraan perusahaan yang membutuhkan penanganan data yang andal dan mudah diakses baik melalui aplikasi mobile maupun web.

2.5.3. Next.js

Next.js merupakan framework React yang digunakan untuk pengembangan aplikasi web dengan pendekatan yang lebih modern. Dalam penelitian terkait perancangan sistem manajemen pembelajaran dan forum diskusi di MTs Nurul Fiqri, Next.js digunakan sebagai framework utama dalam merancang website karena kemampuannya yang mendukung penggunaan berbagai bahasa pemrograman seperti Typescript, HTML, dan CSS serta kompatibilitasnya dengan *library* pihak ketiga (Ispandi et al., 2024). Framework ini menawarkan fitur-fitur canggih seperti *server-side rendering, static site generation*, dan API *routes* yang menjadikannya pilihan ideal untuk aplikasi yang membutuhkan performa tinggi.

Penelitian lain yang dilakukan oleh (Maulana et al., 2023) tentang aplikasi manajemen ruangan, presensi, dan notulensi rapat pada Bappeda Kota Pontianak menunjukkan bahwa Next.js efektif diimplementasikan pada sisi *front-end* bersamaan dengan Express.js pada sisi *back-end.* Kombinasi ini memungkinkan pengembangan aplikasi berbasis web yang responsif dan *user-friendly.* Pendekatan berorientasi objek yang diterapkan dengan Next.js memudahkan pengorganisasian kode sehingga lebih terstruktur dan mudah dipelihara.

Framework ini mendukung pengembangan aplikasi yang bersifat *hybrid*, di mana satu basis kode dapat digunakan untuk menghasilkan baik aplikasi web maupun komponen mobile dengan bantuan React Native. Selain itu, fitur *pre-rendering* pada Next.js dapat meningkatkan performa aplikasi pemesanan ruang rapat dan transportasi yang membutuhkan respons cepat dan *real-time updates.*

Dari kedua penelitian diatas menunjukkan bahwa Next.js merupakan pilihan yang tepat untuk aplikasi yang memerlukan interaktivitas tinggi dan pengelolaan data dinamis seperti sistem pemesanan. Kemampuannya dalam menangani state management, routing, dan integrasi dengan berbagai API akan sangat bermanfaat dalam mengembangkan sistem booking yang mudah digunakan di PT SISI.

2.5.4. Express.js

Express.js merupakan *framework* Node.js yang dirancang untuk mempermudah dan mempercepat proses pembuatan aplikasi berbasis Node.js dengan pola desain yang fleksibel. *Framework* ini dikenal karena sifatnya yang ringan, tidak memerlukan banyak dependensi tambahan, sehingga ideal untuk pengembangan aplikasi web dan API (Bachtiar et al., 2024). Dalam penelitian tentang perancangan backend aplikasi mobile Fruityfit, Express.js menjadi pilihan utama karena kemampuannya dalam menyediakan fitur-fitur penting seperti routing, request handling, dan response..

Penelitian lain yang dilakukan oleh (Nugraha et al., 2022) tentang aplikasi reservasi berbasis web menunjukkan bahwa Express.js menjadi solusi yang efektif untuk mengatasi permasalahan pencatatan dan pengolahan data. *Framework* ini memungkinkan pengembang untuk merancang backend API dengan arsitektur REST (*Representational State Transfer*) yang menawarkan keseragaman *interface* dan skalabilitas yang tinggi. Sistem yang menerapkan prinsip-prinsip REST dengan Express.js juga membantu meminimalkan latensi, memberikan respons yang cepat terhadap request yang masuk.

Dalam konteks pengembangan aplikasi mobile dan web booking ruang rapat dan transportasi PT SISI, Express.js menawarkan berbagai keunggulan. Platform Node.js yang mendasari Express.js memiliki karakteristik *non-blocking*, yang memungkinkan aplikasi menangani banyak *request* secara bersamaan tanpa

menurunkan kinerja sistem. Hal ini sangat penting untuk aplikasi pemesanan ruangan dan transportasi yang mungkin diakses oleh banyak orang pada waktu yang sama.

Dari kedua penelitian tersebut, dapat disimpulkan bahwa Express.js merupakan pilihan yang tepat untuk membangun *backend* aplikasi *booking* yang memerlukan pengelolaan data secara mudah dan lancar bagi banyak pengguna, sehingga aplikasi tetap berjalan baik walaupun diakses banyak orang secara bersamaan. *Framework* ini menyediakan infrastruktur yang kokoh untuk menangani fungsi-fungsi seperti autentikasi pengguna, pencatatan pemesanan, dan manajemen ketersediaan ruangan serta kendaraan. Kemampuan Express.js untuk terintegrasi dengan berbagai teknologi penyimpanan data seperti MySQL dan Google Cloud Platform juga memungkinkan fleksibilitas dalam merancang sistem yang kompleks namun tetap performatif untuk aplikasi booking PT SISI.

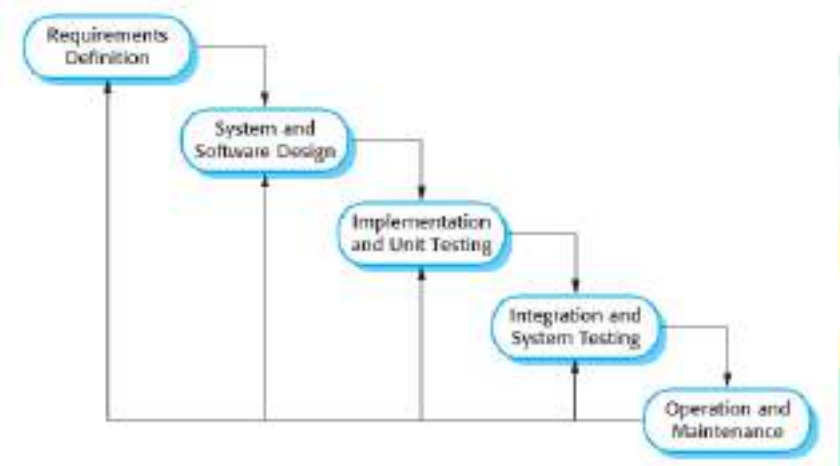
2.5.5. Tailwind Css

Tailwind CSS adalah framework CSS yang menerapkan pendekatan *'utility-first'* untuk mempercepat proses pengembangan antarmuka pengguna (UI) pada aplikasi web. Dengan menyediakan berbagai *utility class*, Tailwind CSS memungkinkan para pengembang untuk merancang tampilan yang unik dan responsif tanpa perlu menulis banyak CSS secara manual (Agustini & Kurniawan, 2020). Pendekatan ini memungkinkan pengembang untuk memanfaatkan kelas-kelas utilitas yang telah disediakan untuk menciptakan desain tanpa batasan, sehingga menyederhanakan proses pengembangan.

Selanjutnya, untuk menyederhanakan penyebutan, kita akan menggunakan istilah Tailwind CSS (Komaran et al., 2023). Dengan desain yang fleksibel dan kontrol lebih besar kepada pengembang, Tailwind CSS menjadi pilihan yang menarik dalam menciptakan aplikasi web. Tailwind CSS dirancang untuk mempercepat proses prototyping, memberikan keleluasaan kepada pengembang dalam membangun desain sesuai dengan keinginan mereka. Hal ini mengurangi ketergantungan pada framework lain seperti Bootstrap, yang sering kali menghasilkan tampilan yang serupa di banyak situs web. Dengan demikian, istilah "tampilan sejuta umat" tidak lagi relevan (Fataha, 2022).

2.7. Pengembangan Sistem Metode *Waterfall*

Metode *Waterfall* adalah model pengembangan perangkat lunak yang prosesnya dilakukan secara berurutan dan bertahap, mulai dari tahap perencanaan, analisis kebutuhan, perancangan, implementasi, hingga pemeliharaan. Pada model ini, setiap tahapan harus diselesaikan terlebih dahulu sebelum melanjutkan ke tahap berikutnya, dan tidak bisa kembali ke tahap sebelumnya jika ada perubahan atau kesalahan. Model *Waterfall* sering dipilih karena alurnya yang rapi dan mudah dipantau, sehingga cocok untuk proyek yang kebutuhannya sudah jelas sejak awal. Namun, metode ini kurang fleksibel jika ada perubahan di tengah jalan karena setiap langkah harus diikuti secara ketat (Pricillia & Zulfachmi, 2021). Metode *Waterfall* memiliki beberapa tahapan, Tahapan metode *waterfall* dapat dilihat pada Gambar 2.1 berikut:



Gambar 2.2 Tahapan Metode *Waterfall* (Sommerville, 2011)

Secara umum, Penjelasan dari metode *waterfall* terdiri dari langkah-langkah berikut: *Requirements Definition, System and Software Design, Implementation and Unit Testing, Integration and System Testing,* dan *Operation and Maintenance* sebagai berikut:

a. *Requirements Definition*

   Requirements Definition adalah tahap awal dalam pengembangan sistem yang bertujuan untuk mengidentifikasi dan mendokumentasikan kebutuhan pengguna serta spesifikasi sistem yang akan dibangun. Pada tahap ini, semua kebutuhan fungsional maupun non-fungsional

dikumpulkan dari berbagai pemangku kepentingan agar pengembangan berjalan sesuai harapan.

b. *System and Software Design*

Setelah kebutuhan sistem terdefinisi, tahap selanjutnya adalah *System* and *Software Design*. Pada tahap ini, kebutuhan yang telah dikumpulkan diterjemahkan ke dalam desain sistem secara keseluruhan, termasuk arsitektur, struktur *database*, serta desain perangkat lunak dan antarmuka pengguna.

c. *Implementation and Unit Testing*

Tahap *Implementation and Unit Testing* dilakukan dengan mengubah desain yang telah dibuat menjadi kode program menggunakan teknologi yang sesuai. Setiap komponen perangkat lunak diuji secara terpisah (*unit testing*) untuk memastikan setiap bagian berfungsi dengan baik sesuai spesifikasi.

d. *Integration and System Testing*

Setelah seluruh komponen selesai diimplementasikan, tahap berikutnya adalah *Integration and System Testing*. Pada tahap ini, semua komponen yang telah diuji secara unit diintegrasikan menjadi satu kesatuan sistem, lalu diuji kembali untuk memastikan seluruh sistem berjalan dengan baik dan sesuai kebutuhan.

e. *Operation and Maintenance*

Tahap terakhir adalah *Operation and Maintenance*, di mana sistem yang telah selesai dikembangkan mulai digunakan oleh pengguna. Pada tahap ini juga dilakukan pemeliharaan, perbaikan, serta penyesuaian jika ditemukan masalah atau ada kebutuhan baru selama sistem dioperasikan.

2.8. Penelitian Sejenis

Pada sub bab ini membahas penelitian-penelitian terdahulu yang relevan dengan pengembangan dan penerapan sistem informasi untuk meningkatkan kelancaran operasional perusahaan, khususnya dalam konteks pemesanan ruang rapat dan transportasi. Hasil-hasil penelitian tersebut dirangkum dalam Tabel 2.3. Berdasarkan kajian terhadap penelitian sebelumnya, sistem reservasi terbukti memberikan kontribusi yang signifikan dalam mengoptimalkan pengelolaan

fasilitas serta mempercepat proses administrasi yang berkaitan dengan pemesanan fasilitas perusahaan.

Sistem ini juga mempermudah koordinasi antar unit bisnis dengan menyediakan informasi yang akurat terkait ketersediaan ruang dan transportasi. Namun, banyak sistem reservasi yang ada saat ini masih memiliki keterbatasan. Banyak sistem yang belum mampu memberikan *visibilitas real-time* terkait status ketersediaan fasilitas, sehingga pengguna sering kesulitan dalam memastikan apakah ruang rapat atau transportasi yang diinginkan tersedia pada waktu yang diinginkan. Selain itu, sebagian besar sistem belum mengintegrasikan secara komprehensif informasi *inventaris* fasilitas, status kendaraan, serta pemberitahuan melalui email dalam satu platform. Laporan yang dihasilkan juga seringkali kurang mendetail dan tidak mendukung proses perencanaan atau evaluasi dengan baik.

Berdasarkan identifikasi masalah tersebut, penelitian ini berfokus pada pembangunan aplikasi pemesanan ruang rapat dan transportasi berbasis *mobile* di PT Sinergi Informatika Semen Indonesia (SISI), dengan menambahkan fitur-fitur unggulan yang dirancang untuk mengatasi keterbatasan sistem yang ada saat ini. Fitur utama yang akan dikembangkan meliputi dashboard analitik *real-time* yang menyediakan pemantauan visual terkait status ketersediaan ruang rapat dan transportasi, yang dapat diakses oleh pengguna dan administrator secara langsung. Fitur ini bertujuan untuk meminimalkan bentrok jadwal dan meningkatkan akurasi dalam pengelolaan fasilitas. Sistem ini juga akan mengintegrasikan data inventaris fasilitas, seperti jumlah dan jenis ruang rapat serta status kendaraan operasional, untuk memastikan ketersediaan fasilitas dapat dikelola. Selain itu, aplikasi ini akan menyediakan pembuatan laporan yang mendetail dan dapat diekspor ke dalam format excel, yang mendukung proses evaluasi dan perencanaan yang lebih baik, serta memberikan gambaran yang lebih jelas kepada manajemen mengenai kemudahan penggunaan fasilitas. Pemberitahuan melalui email juga akan diimplementasikan untuk memberikan pembaruan kepada pengguna terkait status pemesanan, baik itu konfirmasi, pembatalan, atau perubahan jadwal. Fitur pemberitahuan melalui email ini diharapkan dapat meningkatkan transparansi dan memperbaiki komunikasi antar unit yang terlibat dalam pemesanan.

Inovasi-inovasi ini diharapkan dapat meningkatkan produktivitas dan kelancaran operasional PT SISI, sekaligus memperbaiki koordinasi dan komunikasi antar unit terkait. Dengan adanya aplikasi berbasis mobile ini, perusahaan dapat mengoptimalkan penggunaan fasilitas yang tersedia, meningkatkan pengalaman pengguna dalam proses pemesanan, serta mendukung pengambilan keputusan strategis yang lebih efektif dalam pengelolaan fasilitas ruang rapat dan transportasi. Penelitian ini diharapkan dapat menjadi solusi yang mendukung keberlanjutan layanan dan operasional PT SISI serta entitas bisnis terkait di masa depan.

Tabel 2.3 Studi Literatur

No	Judul	Peneliti	Tahun	Fitur	Metode	Kesimpulan
1.	Smartphone Technician Service Booking Application (Technician2U)	Mohamad Aqmal Syafiq Zaihan, Mohd Amin Mohd Yunus	2022	Login dan Registrasi, Pemesanan, Perbandingan Harga, Navigasi Teknisi, Notifikasi Push, Peringkat Layanan	Prototype	Aplikasi ini menyederhanakan proses pemesanan untuk layanan perbaikan ponsel pintar, memungkinkan pengguna untuk dengan mudah menemukan teknisi terdekat, membandingkan harga, dan menerima layanan secara mudah. Aplikasi ini memberikan fleksibilitas bagi teknisi untuk mengatur janji temu dan menavigasi ke pelanggan.
2.	Hotel Booking Mobile and Web Application	R. Karthikeyan & S. Karthick	2023	Room booking, Admin dashboard, Real-time tracking, Payment integration	Waterfall	Sistem ini meningkatkan kemudahan dalam pemesanan dengan pelacakan dan otomatisasi waktu nyata.
3.	Perancangan Aplikasi Pemesanan Tiket Bioskop di Kota Medan Berbasis Android	Laila Nurhidayah, Ayna Salsabillah, Fitri Rahma Yanti	2023	Login, Registrasi, Daftar Film, Pemilihan Kursi, Pembayaran, QR Code Tiket	Waterfall	Aplikasi ini berhasil dirancang menggunakan React Native dengan fitur pemesanan tiket yang memudahkan pengguna menghindari antrean panjang dan memberikan pengalaman yang lebih baik.

Tabel 2.3 Studi Literatur (Lanjutan)

4.	Aplikasi Pemesanan Ruangan Panti Jompo Yayasan Karya Asih Berbasis Android	Frans Ikorasaki, Rida Utami, Cindy Paramitha Lubis, Nur Anzelina Hrp, Bintang Wildan Utama, Nandri Marsan Sitinjak	2024	Registrasi, Login, Pemesanan Ruangan, Kategori Kamar (VIP, Top Level, Medium, Standard), Pembayaran, Riwayat Pemesanan, Konfirmasi Pembayaran	Waterfall	Aplikasi ini mempermudah keluarga memesan ruangan di panti jompo Yayasan Karya Asih secara online, mengoptimalkan akses dan penggunaan fasilitas panti jompo melalui perangkat Android.
5.	Android-Based Sports Infrastructure E-Booking Application at Provincial Youth and Sports Office Using Waterfall Method	Firda Aulia Rahma, Samsudin	2024	Login, Register, Penelusuran, Pemesanan Online, Unggah Bukti Transaksi, Verifikasi Admin, Manajemen Status	Waterfall	Aplikasi ini menyederhanakan proses pemesanan infrastruktur olahraga, memungkinkan pengguna untuk memesan fasilitas secara online dan menyediakan proses yang mudah bagi admin untuk mengelola pemesanan dan memverifikasi transaksi.

BAB III
METODOLOGI PENELITIAN

Bab ini berisi penjelasan tentang objek penelitian, metode pengumpulan data dan flowchart penelitian pada Sistem Informasi Manajemen Booking Ruang Rapat dan Driver di PT Sinergi Informatika Semen Indonesia (SISI).

3.1. Objek Penelitian

Objek penelitian ini adalah PT Sinergi Informatika Semen Indonesia (SISI), sebuah perusahaan teknologi informasi yang merupakan anak usaha dari PT Semen Indonesia (Persero) Tbk (SIG). Berdiri sejak 9 Juni 2014, SISI pada awalnya merupakan tim inti yang bertanggung jawab atas pengembangan dan dukungan operasional ICT untuk seluruh ekosistem SIG Group. Seiring waktu, perusahaan ini bertransformasi menjadi mitra transformasi digital yang memperluas layanannya tidak hanya untuk internal grup, tetapi juga untuk lingkungan BUMN dan industri lainnya.

Sebagai perusahaan penyedia solusi IT, PT SISI memiliki portofolio bisnis yang komprehensif dan terbagi menjadi tiga pilar utama. Pilar pertama adalah *Shared Services*, yang menyediakan layanan terpusat untuk operasional keuangan, sumber daya manusia, pembelian, dan IT. Pilar kedua, *Digital Solution*, menawarkan produk berbasis *Software as a Service* (SaaS) seperti FORCA ERP dan FORCA HR, serta platform e-commerce SobatBangun. Pilar ketiga adalah *System Integrator*, yang mencakup layanan konsultasi, pengembangan aplikasi, hingga penyediaan perangkat keras dan lunak. Kompleksitas dan skala operasional perusahaan ini didukung oleh komitmen terhadap standar mutu melalui sertifikasi ISO 9001:2015 dan keamanan informasi dengan ISO 27001.

3.2. Metode Pengumpulan Data

Metode pengumpulan data yang digunakan dalam penelitian ini meliputi wawancara dan studi literatur. Adapun deskripsi dari masing-masing metode dijabarkan sebagai berikut:

1. Wawancara

   Wawancara dilakukan untuk memperoleh informasi mendalam terkait proses bisnis dan kebutuhan operasional dalam Sistem Informasi Manajemen *Booking* Ruang Rapat dan Transportasi di PT Sinergi Informatika Semen Indonesia (SISI). Wawancara ini dilakukan dengan pihak-pihak terkait, termasuk manajer dan staf operasional, untuk mengidentifikasi tantangan yang dihadapi dalam pengelolaan sistem yang ada. Melalui wawancara, peneliti berusaha menggali informasi kualitatif yang relevan, seperti kebutuhan akan fitur baru, kendala dalam penggunaan sistem saat ini, serta harapan terhadap pengembangan sistem yang lebih baik.

2. Observasi

   Observasi dilakukan dengan cara mengamati secara langsung tampilan dan alur kerja pada sistem website yang sudah ada sebelumnya. Tujuannya adalah untuk memahami bagaimana proses booking ruang rapat dan transportasi dijalankan saat ini, serta mengidentifikasi kekurangan dari sisi fungsionalitas dan antarmuka pengguna. Hasil observasi ini menjadi dasar dalam merancang sistem baru yang lebih responsif dan sesuai dengan kebutuhan pengguna.

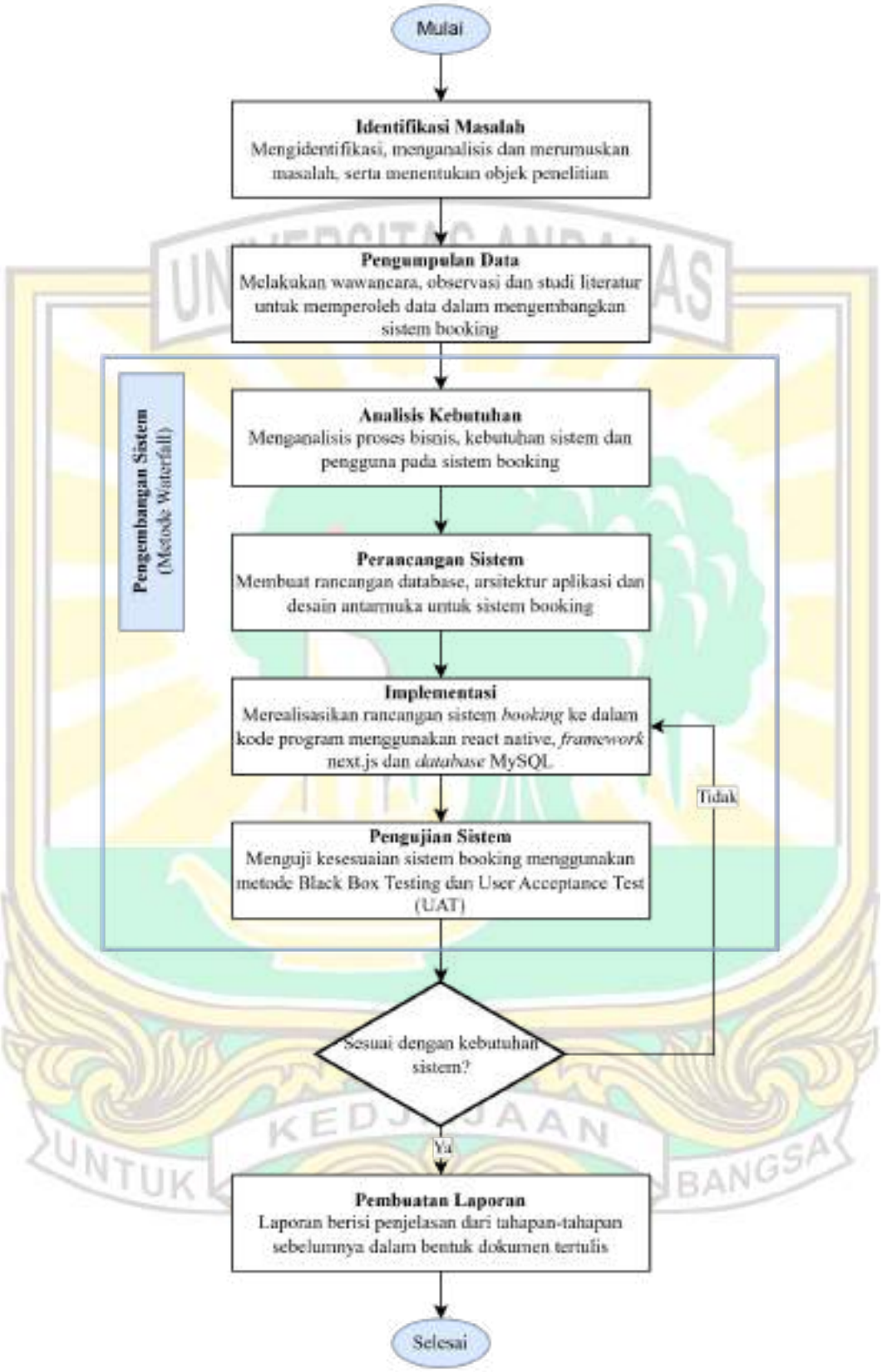
3. Studi Literatur

   Studi literatur dilakukan dengan menganalisis berbagai sumber tertulis yang relevan untuk mendukung penelitian ini. Sumber yang dikaji mencakup jurnal ilmiah, buku, dan artikel di internet yang berhubungan dengan praktik terbaik dalam pengelolaan ruang rapat dan kendaraan. Dengan melakukan studi literatur, peneliti dapat memperoleh wawasan yang lebih luas mengenai konsep dan teori yang berkaitan, serta

menemukan studi kasus yang dapat dijadikan referensi dalam pengembangan sistem di PT SISI.

3.3. *Flowchart* Penelitian

Metode atau langkah-langkah yang dilakukan dalam penelitian tentang Pembangunan Sistem Informasi *Booking* Ruang Rapat dan Transportasi di PT Sinergi Informatika Semen Indonesia (SISI) ini digambarkan pada flowchart penelitian yang dapat dilihat pada Gambar 3.1 berikut:



Gambar 3.1 *Flowchart* Penelitian

Penjelasan flowchart penelitian dari Gambar 3.1 dapat diuraikan sebagai berikut:

1. Identifikasi Masalah

   Identifikasi Masalah Pada tahap ini peneliti mengidentifikasi, menganalisis dan merumuskan masalah, serta menentukan objek penelitian terkait dengan pengembangan sistem *booking*. Tahap ini melibatkan identifikasi kebutuhan sistem yang akan dibangun, termasuk permasalahan yang sering terjadi dalam proses *booking*. Melalui proses ini, akan ditentukan ruang lingkup dan tujuan dari sistem yang akan dikembangkan.
2. Pengumpulan Data

   Pengumpulan data dilakukan melalui wawancara, observasi, dan studi literatur. Wawancara dilakukan dengan pihak terkait untuk memahami proses booking yang berjalan, observasi dilakukan terhadap sistem website sebelumnya, dan studi literatur digunakan untuk memperoleh referensi teoritis dalam pengembangan sistem.
3. Analisis Kebutuhan

   Pada tahap ini dilakukan analisis proses bisnis, kebutuhan sistem, dan pengguna pada sistem *booking*. Analisis mencakup identifikasi kebutuhan fungsional dan non-fungsional sistem, serta menganalisis kebutuhan pengguna yang akan menggunakan sistem *booking* tersebut.
4. Perancangan Sistem

   Tahap ini fokus pada pembuatan rancangan database, arsitektur aplikasi dan desain antarmuka untuk sistem *booking*. Perancangan dilakukan berdasarkan hasil analisis kebutuhan yang telah dilakukan sebelumnya untuk memastikan sistem yang dibangun sesuai dengan kebutuhan.
5. Implementasi

   Pada tahap ini dilakukan realisasi rancangan sistem *booking* ke dalam kode program menggunakan *framework* Laravel dan database MySQL. Implementasi mencakup pengkodean seluruh fitur dan fungsi yang telah dirancang sebelumnya.

6. Pengujian Sistem

   Pengujian dilakukan untuk menguji kesesuaian sistem booking menggunakan metode Black Box Testing dan User Acceptance Test (UAT). Metode Black Box Testing berfokus pada pengujian fungsional sistem untuk memastikan bahwa sistem berjalan sesuai dengan spesifikasi yang telah ditentukan tanpa melihat kode program internal, sedangkan UAT dilakukan untuk menilai sejauh mana sistem telah memenuhi kebutuhan dan harapan pengguna akhir.

7. Laporan

   Setelah seluruh tahapan selesai, peneliti menyusun laporan penelitian. Laporan ini berisi penjelasan rinci mengenai setiap tahap yang telah dilalui, yang dituangkan dalam bentuk dokumen tertulis, sehingga dapat menjadi referensi bagi penelitian atau pengembangan sistem lebih lanjut.

BAB IV
ANALISIS DAN PERANCANGAN SISTEM

Bab ini menjelaskan hasil analisis dan perancangan sistem dari Sistem Informasi Booking Ruang Rapat dan Transportasi Berbasis *Mobile* Pada PT SISI (Sinergi Informatika Semen Indonesia). Hasil analisis tersebut mencakup *Business Process Model Notation* (BPMN), analisis kebutuhan fungsional, *use case diagram*, *use case scenario*, dan *sequence diagram*. Sedangkan perancangan sistem meliputi perancangan *database*, struktur tabel dan basis data, dan perancangan antarmuka

4.1. Analisis Sistem

Tahap analisis sistem menjelaskan kondisi sistem yang sedang berjalan dan sistem yang diusulkan, yang divisualisasikan menggunakan *Business Process Model and Notation* (BPMN). Selain itu, analisis sistem juga dimodelkan menggunakan *Unified Modeling Language* (UML), yang meliputi *use case diagram*, *use case scenario*, dan *sequence diagram* untuk menggambarkan alur dan interaksi sistem secara detail.

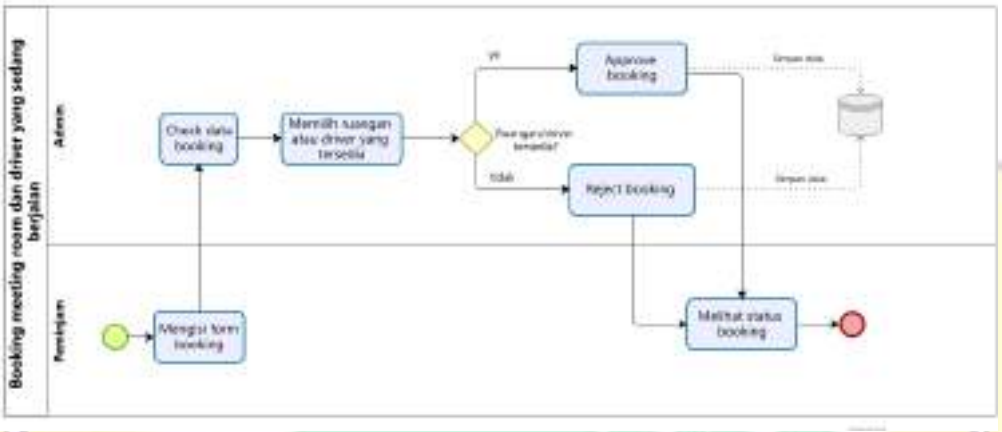
4.1.1. Alur Sistem Booking yang Sedang Berjalan

Proses *Booking* Ruang Rapat dan Transportasi yang sedang berjalan dimulai dari peminjam mengisi *form booking*, yang kemudian diterima oleh Admin. Admin memilih ruang atau *driver* yang tersedia dan mengecek ketersediaannya. Jika tersedia, peminjaman dikonfirmasi dan data *approval* disimpan. Jika tidak tersedia, peminjaman ditolak. Terakhir, Peminjam mengecek status *booking* untuk melihat hasilnya di riwayat *booking*. Untuk detail alur proses *booking* ruang rapat dan transportasi yang sedang berjalan dapat dilihat pada Gambar 4.1.

Berdasarkan Gambar 4.1, berikut penjelasan tahapan pada proses *booking* ruang rapat dan transportasi yang sedang berjalan pada PT SISI.

1. Proses dimulai ketika peminjam mengisi *form booking*
2. Selanjutnya, admin akan mengecek data *booking* tersebut
3. Kemudian admin memilih ruangan atau driver yang tersedia

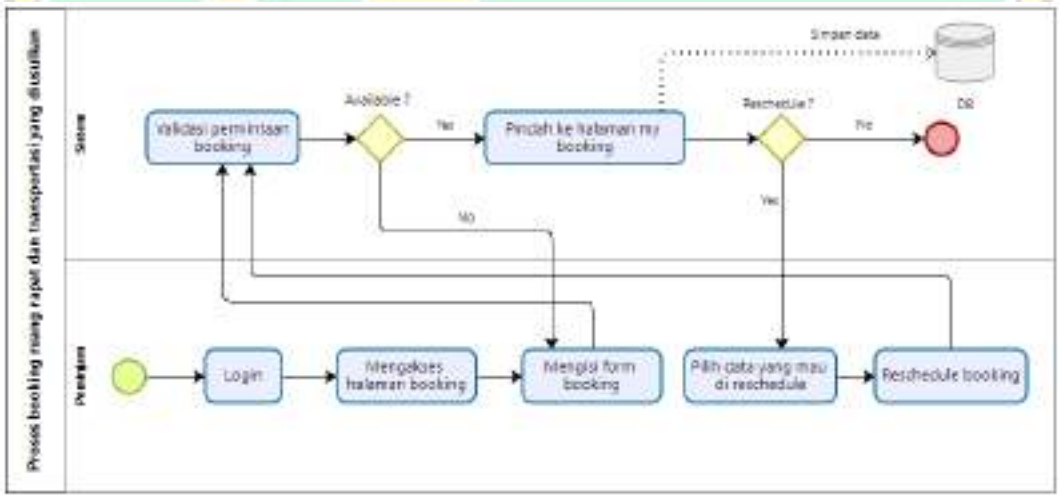
4. Jika tersedia, maka admin akan *approve booking*, namun jika tidak tersedia, admin akan *reject booking*
5. Setelah itu, peminjam dapat melihat status *booking*, proses selesai



Gambar 4.1 BPMN *Booking* Ruang Rapat dan Transportasi yang Sedang Berjalan

4.1.2. Alur Sistem Booking yang Diusulkan

Proses *Booking* Ruang Rapat dan Transportasi yang diusulkan dimulai dari peminjam mengisi *form booking*, jika form sudah diisi dengan benar dan lolos validasi form peminjam mengirimkan data bookingnya dengan memencet tombol submit pada form.. Untuk detail alur proses *booking* ruang rapat dan transportasi yang diusulkan dapat dilihat pada Gambar 4.2 berikut.



Gambar 4.2 BPMN *Booking* Ruang Rapat dan Transportasi yang Diusulkan

Berikut penjelasan sistem booking ruang rapat dan transportasi yang diusulkan berdasarkan Gambar 4.2.

1. Proses dimulai ketika peminjam mengisi *form booking*
2. Selanjutnya, sistem akan memvalidasi data *booking* tersebut
3. Jika available, maka sistem akan menampilkan halaman my booking, namun jika tidak, sistem akan tampilkan *alert error* sehingga mengharuskan peminjam memilih jadwal yang available
4. Jika peminjam ingin melakukan reschedule bookingannya, maka tinggal memilih data bookingnya di halaman my booking dan reschedule jadwal booking yang diinginkan.
5. Proses selesai

4.1.3. Kebutuhan Fungsional

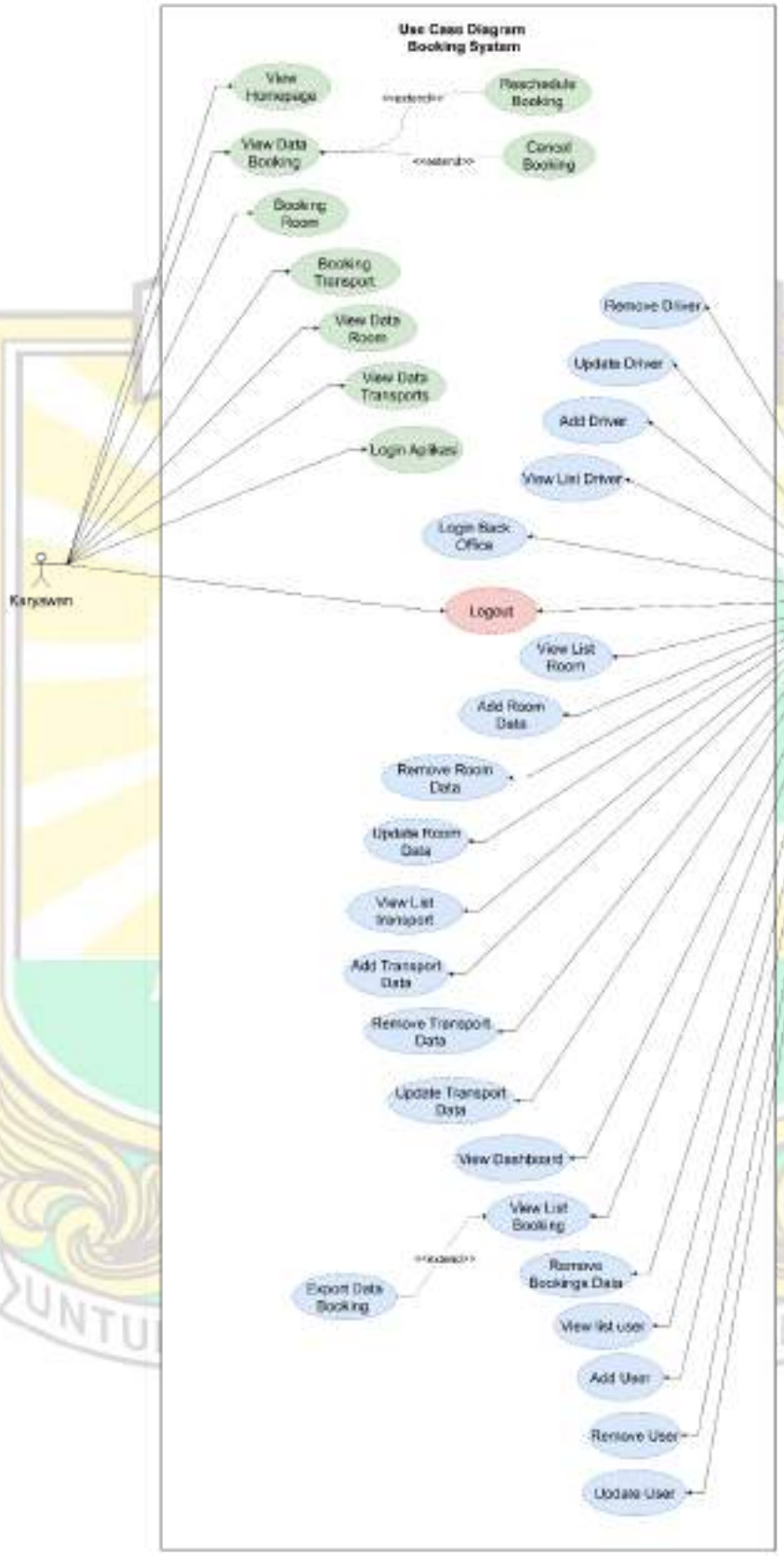
Analisis kebutuhan fungsional didasarkan pada alur proses bisnis dan wawancara terkait sistem yang diusulkan, dengan menyesuaikan kebutuhan fungsional sesuai peran masing-masing aktor yang terlibat. Aktor yang terlibat dalam sistem usulan ini terdiri dari Admin dan Karyawan. Berikut kebutuhan fungsional yang telah dirumuskan berdasarkan analisis sistem usulan.

1. Admin dapat login *back office*
2. Admin dapat logout *back office*
3. Admin dapat melihat *dashboard*
4. Admin dapat melihat data ruangan
5. Admin dapat menambah data ruangan
6. Admin dapat menghapus data ruangan
7. Admin dapat mengedit data ruangan
8. Admin dapat melihat data transportasi
9. Admin dapat menambah data transportasi
10. Admin dapat menghapus data transportasi
11. Admin dapat mengedit data transportasi
12. Admin dapat melihat data *user*
13. Admin dapat menambah data *user*
14. Admin dapat menghapus data *user*
15. Admin dapat mengedit data *user*

16. Admin dapat melihat detail *booking*
17. Admin dapat menghapus data *booking*
18. Admin dapat mengekspor data *booking*
19. Karyawan dapat login aplikasi *mobile*
20. Karyawan dapat logout aplikasi *mobile*
21. Karyawan dapat melihat halaman *home*
22. Karyawan dapat melihat *profile*
23. Karyawan dapat melakukan *booking* ruang rapat
24. Karyawan dapat melihat data *my booking*
25. Karyawan dapat melakukan *reschedule booking*
26. Karyawan dapat melakukan *cancel booking*
27. Karyawan dapat melakukan *booking* transportasi
28. Karyawan dapat melihat data *explore*

4.1.4. *Use Case Diagram*

Berdasarkan hasil analisis kebutuhan fungsional, setiap kebutuhan yang telah diidentifikasi dihubungkan dengan aktor-aktor yang berperan dalam sistem dan kemudian dimodelkan menjadi sebuah *use case diagram*. Setiap aktor memiliki hak akses terhadap fungsional sistem yang berbeda-beda sesuai dengan peran dan tanggung jawabnya masing-masing. *Use case diagram* untuk aplikasi Booking Ruang Rapat dan Transportasi dapat dilihat pada Gambar 4.3 berikut.



Gambar 4.3 *Use Case Diagram*

4.1.5. *Use Case Scenario*

Use case scenario merupakan serangkaian deskripsi proses yang menggambarkan interaksi antara aktor dengan sistem dalam melaksanakan suatu kegiatan. *Use case scenario* yang akan dijelaskan pada subbab ini meliputi menambah data ruangan, *booking* ruang rapat, *reschedule* dan *cancel booking*. *Use case scenario* lainnya dapat dilihat pada Lampiran A.

4.1.5.1. Menambahkan Data Ruangan

Use case scenario ini menjelaskan tentang kegiatan yang dilakukan oleh Admin untuk menambahkan data ruangan. *Use case scenario* menambahkan data ruangan dapat dilihat pada Tabel 4.1 berikut.

Tabel 4.1 *Use Case Scenario* Menambahkan Data Ruangan

Use Case :	Menambahkan Data Ruangan
Actor :	Admin
Preconditions :	1. Aktor telah *login* ke dalam aplikasi 2. Aktor berada di halaman *list room*
Flow of Event :	1. Aktor klik "*Add Room*" di halaman *list room*. 2. Sistem menampilkan halaman *form input* untuk *add room*. 3. Aktor mengisi *form* dengan data *room* yang diperlukan 4. Aktor klik tombol "*create room*". 5. Sistem melakukan validasi terhadap *input*. 6. Jika semua validasi berhasil, sistem menyimpan data *room* ke dalam *database*. 7. Sistem menampilkan pemberitahuan "room created successfully". 8. Sistem mengarahkan aktor kembali ke halaman *list room.*
Alternate Flows :	A1. Jika aktor tidak mengisi semua *field* yang diperlukan, sistem menampilkan pesan error.
Postconditions :	Aktor berhasil menambah data ruangan.

4.1.5.2. Melakukan *Booking* Ruang Rapat

Use case scenario ini menjelaskan tentang kegiatan yang dilakukan oleh Karyawan untuk booking ruang rapat. *Use case scenario* melakukan booking ruang rapat dapat dilihat pada Tabel 4.2 berikut.

Tabel 4.2 *Use Case Scenario Booking* Ruang Rapat

Use Case :	Booking Ruang Rapat
Actor :	Karyawan
Preconditions :	1. Aktor telah *login* ke dalam aplikasi 2. Aktor berada di halaman *booking room*
Flow of Event :	1. Aktor klik opsi "*Room*" di halaman opsi *booking*. 2. Sistem menampilkan halaman *form input* untuk *booking room* baru. 3. Aktor mengisi *form* dengan data *booking* yang diperlukan 4. Aktor klik tombol "*Submit*" . 5. Sistem melakukan validasi terhadap *input*. 6. Jika semua validasi berhasil, sistem menyimpan data baru ke dalam *database*. 7. Sistem menampilkan pemberitahuan "*Booking submitted successfully*". 8. Sistem mengarahkan aktor ke halaman my *booking*
Alternate Flows :	A1. Jika aktor tidak mengisi semua *field* yang diperlukan, sistem menampilkan pesan error.
Postconditions :	Aktor berhasil melakukan booking ruang rapat

4.1.5.3. Melakukan *Reschedule Booking*

Use case scenario ini menjelaskan tentang kegiatan yang dilakukan oleh Karyawan untuk melakukan *reschedule booking*. *Use case scenario* melakukan *reschedule booking* dapat dilihat pada Tabel 4.4 berikut.

Tabel 4.3 *Use Case Scenario Reschedule Booking*

Use Case :	*Reschedule Booking*
Actor :	Karyawan
Preconditions :	1. Aktor telah *login* ke dalam aplikasi 2. Aktor berada di halaman *my booking*
Flow of Event :	1. Aktor pilih *booking* dengan status "Pending" 2. Sistem menampilkan halaman *detail booking*. 3. Aktor klik tombol "*reschedule*" 4. Sistem akan menampilkan halaman *form reschedule* 5. Aktor memilih jadwal atau waktu yang ingin di *reschedule* 6. Aktor klik tombol "*Save*" 7. Sistem melakukan validasi terhadap *input*, jika semua validasi berhasil, sistem menyimpan data baru ke dalam *database*. 8. Sistem menampilkan pemberitahuan "*Booking updated successfully*". 9. Sistem mengarahkan aktor kembali ke halaman *my booking*
Alternate Flows :	A1. Jika aktor memilih tanggal dan waktu yang sudah lewat, sistem menampilkan pesan error
Postconditions :	Aktor berhasil melakukan r*eschedule booking*

4.1.5.4. Melakukan *Cancel Booking*

Use case scenario ini menjelaskan tentang kegiatan yang dilakukan oleh Karyawan untuk melakukan *cancel booking*. *Use case scenario* melakukan *cancel booking* dapat dilihat pada Tabel 4.5 berikut.

Tabel 4.4 *Use Case Scenario Cancel Booking*

Use Case :	*Cancel Booking*
Actor :	Karyawan
Preconditions :	1. Aktor telah *login* ke dalam aplikasi 2. Aktor berada di halaman *my booking*
Flow of Event :	1. Aktor pilih booking dengan status "*pending*" 2. Sistem menampilkan halaman *detail booking.* 3. Aktor memilih tombol "*Cancel*". 4. Sistem menampilkan pemberitahuan *"are u sure ?*". 5. Aktor klik tombol "*Yes*". 6. Sistem mengarahkan aktor kembali ke halaman *my booking.*
Postconditions :	Aktor berhasil melakukan *cancel booking*

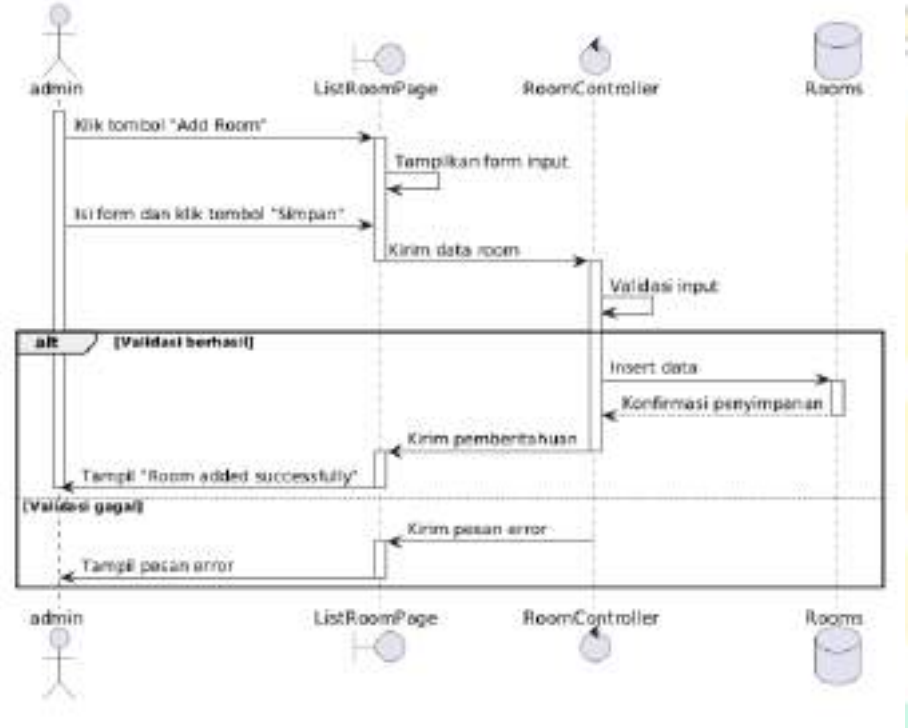
4.1.6. *Sequence Diagram*

Sequence diagram menjelaskan aliran pesan dan data dalam proses interaksi antar objek di dalam sistem secara berurutan. Berdasarkan *use case scenario* yang telah diuraikan pada subbab sebelumnya, *sequence diagram* yang dibahas pada subbab ini mencakup menambah data ruangan, booking ruang rapat, *reschedule* dan c*ancel booking*. *Sequence diagram* lainnya dapat dilihat pada Lampiran B.

4.1.6.1. Menambahkan Data Ruangan

Sequence diagram ini menjelaskan proses interaksi yang terjadi pada saat admin ingin menambahkan data ruangan. Proses dimulai ketika admin mengklik tombol "*Add Room*", lalu akan muncul form input dan diisi sesuai kebutuhan. Setelah itu, admin memencet tombol "*Create Room*". Sistem akan melakukan

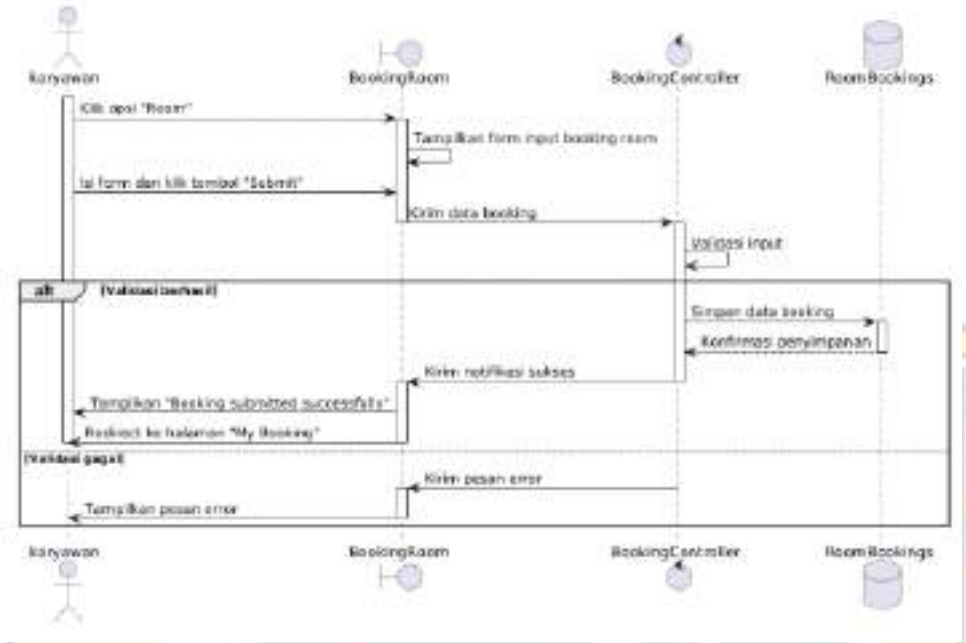
validasi terhadap data yang dimasukkan. Jika validasi berhasil, data akan dikirim ke *RoomController* untuk diproses dan disimpan ke dalam sistem, kemudian ditampilkan *alert* "*Room added successfully!*". Namun, jika validasi gagal, maka sistem akan menampilkan pesan *error*. *Sequence diagram* menambahkan data ruangan dapat dilihat pada Gambar 4.4 berikut.



Gambar 4.4 *Sequence Diagram* Menambahkan Data Ruangan

4.1.6.2. Melakukan *Booking* Ruang Rapat

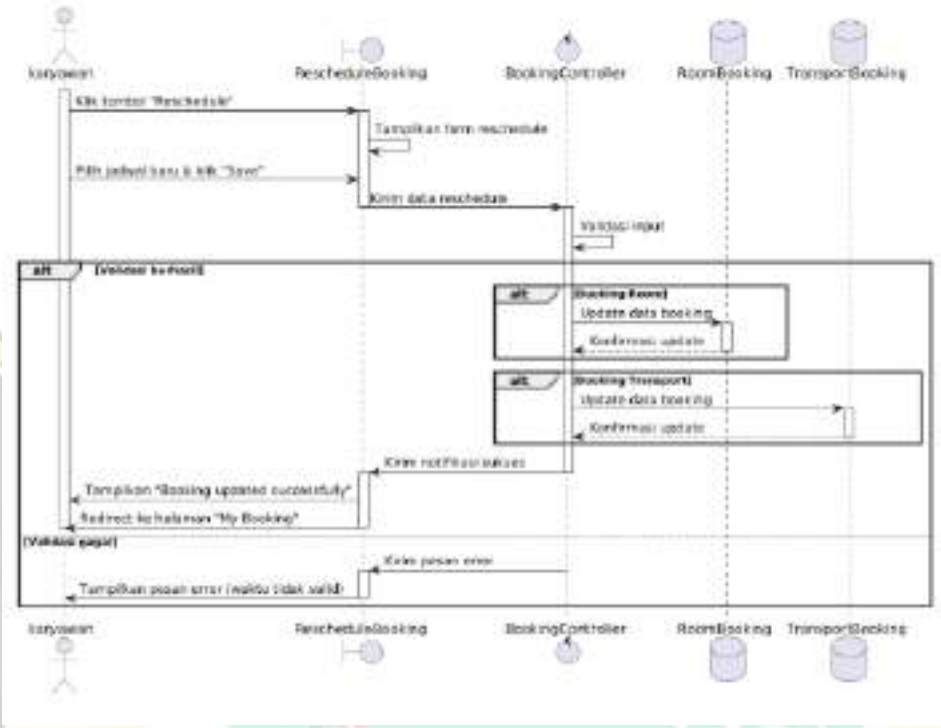
Sequence diagram ini menjelaskan proses interaksi yang terjadi ketika karyawan ingin melakukan pemesanan (*booking*) ruangan. Proses dimulai saat karyawan mengklik tombol "*Book Room*", kemudian sistem akan meminta dan menampilkan data ruangan. Setelah itu, karyawan mengisi form pemesanan dan memencet tombol "*Book Now*". Sistem akan melakukan validasi terhadap data yang dimasukkan. Jika validasi berhasil, data pemesanan dikirim ke *RoomBookingController* untuk disimpan, lalu ditampilkan *alert* "*Room booked successfully!*". Namun, jika validasi gagal, sistem akan menampilkan pesan error. Sequence diagram pemesanan ruangan dapat dilihat pada Gambar 4.5 berikut.



Gambar 4.5 *Sequence Diagram* Melakukan *Booking* Ruang Rapat

4.1.6.3. Melakukan *Reschedule Booking*

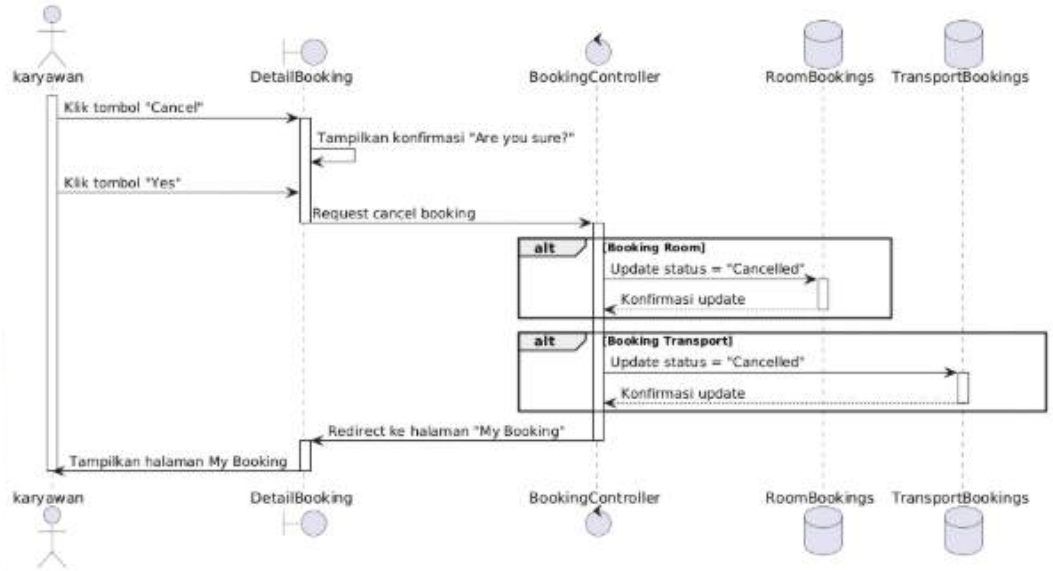
Sequence diagram ini menjelaskan proses interaksi saat karyawan melakukan *reschedule* terhadap permintaan *booking* ruangan yang berstatus *pending*. Proses dimulai ketika karyawan memilih *booking* yang ingin dijadwalkan ulang, kemudian sistem akan menampilkan *detail booking* dan form *reschedule*. Setelah karyawan memilih jadwal baru dan mengisi form, sistem akan melakukan validasi. Jika validasi berhasil, sistem akan memperbarui data booking dan menampilkan pesan *"Booking updated successfully!*". Namun jika gagal, sistem akan menampilkan pesan error. Sequence diagram penjadwalan ulang *booking* dapat dilihat pada Gambar 4.6 berikut.



Gambar 4.6 *Sequence Diagram* Melakukan *Reschedule Booking*

4.1.6.4. Melakukan *Cancel Booking*

Sequence diagram ini menjelaskan proses interaksi saat karyawan untuk melakukan *cancel booking* yang telah dilakukan. Proses dimulai ketika karyawan memilih data *booking* yang ingin dibatalkan, kemudian sistem akan menampilkan *detail booking*. Setelah karyawan memencet tombol "*Cancel*", sistem akan meminta konfirmasi dan mengirimkan data pembatalan. Jika pembatalan berhasil, sistem akan memperbarui status *booking* menjadi *Canceled* dan menampilkan pesan "*Booking canceled successfully!*". Sequence diagram pembatalan *booking* dapat dilihat pada Gambar 4.7 berikut.



Gambar 4.7 *Sequence Diagram* Melakukan *Cancel Booking*

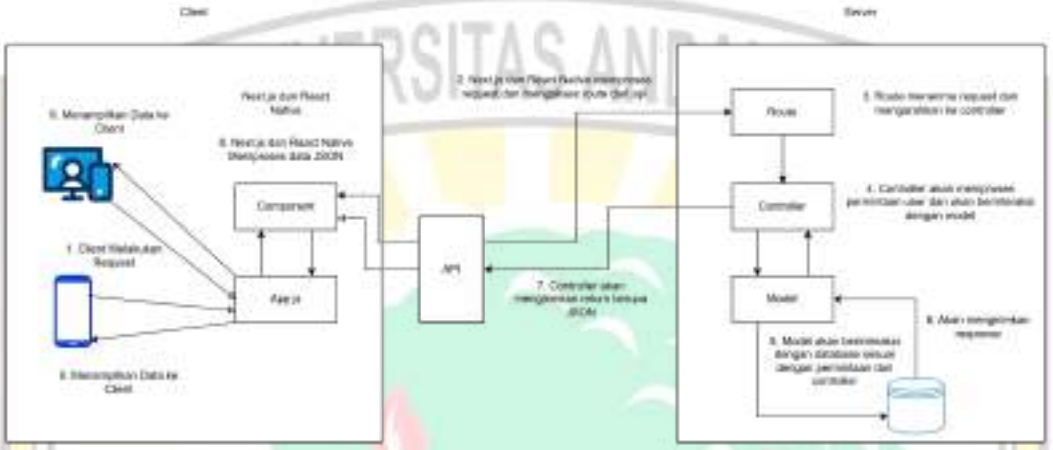
4.2. Perancangan Sistem

Berdasarkan analisis proses bisnis dan kebutuhan sistem, diperoleh rancangan Sistem Informasi *Booking* Ruang Rapat dan Transportasi Berbasis *Mobile* Pada PT SISI. Rancangan ini mencakup perancangan *database*, struktur tabel, dan antarmuka.

4.2.1. Arsitektur Aplikasi

Arsitektur aplikasi pada sistem informasi *booking* ruang rapat dan transportasi PT SISI menerapkan model *client-server*. Pada sisi *server*, Node.js dengan Express.js berfungsi sebagai pusat pengolahan data, menangani permintaan dari client, dan berkomunikasi dengan database MySQL. Pada sisi *client*, aplikasi web admin dikembangkan menggunakan Next.js dan React yang menyajikan tampilan responsif untuk berbagai ukuran layar desktop, sedangkan aplikasi *mobile* untuk karyawan dibangun dengan React Native khusus untuk *smartphone*. Model *client-server* ini memberikan solusi untuk sistem lama melalui pembangunan aplikasi *mobile* bagi karyawan, implementasi fitur pemberitahuan melalui email, dan penerapan *dashboard* monitoring untuk manajemen. Komunikasi antara *client* dan *server* dirancang untuk mendukung pertukaran data, yang memungkinkan penggunaan fitur-fitur seperti *booking* ruang rapat dan

transportasi. Hal ini mendukung tujuan penelitian untuk membangun sistem yang mempercepat proses operasional dan membantu pengambilan keputusan berdasarkan data analitik yang tersedia pada *dashboard*. Gambar 4.8 menampilkan arsitektur aplikasi.



Gambar 4.8 Arsitektur Aplikasi

4.2.2. Perancangan *Database*

Pada bab ini, akan dijelaskan mengenai perancangan *database* untuk sistem informasi yang dikembangkan. Sistem informasi ini menggunakan MySQL sebagai platform untuk penyimpanan data, yang menawarkan keandalan dan skalabilitas dalam mengelola informasi. Struktur database terdiri dari enam tabel utama, yaitu Admin, *RoomBookings*, *DetailBookingRoom, Rooms*, *TransportBookings*, *DetailBookingTransport*, *Transports*, *Drivers* dan *Users*. Setiap tabel memiliki fungsi spesifik yang saling berinteraksi untuk mendukung fungsionalitas sistem secara keseluruhan.

Selanjutnya, akan dijelaskan lebih mendalam mengenai keberadaan masing-masing entitas dan alasan mengapa entitas-entitas ini diperlukan dalam perancangan database. Setiap tabel memiliki peran yang sangat penting dalam menjaga kelancaran operasional sistem, memastikan pengelolaan data yang tepat, dan mendukung kebutuhan pengguna. Berikut adalah penjelasan lebih lanjut mengenai masing-masing entitas dalam database ini:

1. Admin

   Tabel Admin digunakan untuk menyimpan data mengenai administrator sistem, yang memiliki hak akses untuk mengelola dan memantau data dalam sistem. Admin bertanggung jawab atas pengelolaan akun pengguna, pengaturan fasilitas yang tersedia, dan pengaturan lainnya yang diperlukan untuk kelancaran operasional. Tanpa tabel admin, tidak ada pihak yang dapat memantau atau mengelola data di dalam sistem. Hal ini akan membuat sistem sulit untuk dikelola dan lebih rentan terhadap penyalahgunaan atau kesalahan pengaturan data. Fungsi tabel ini sangat penting karena menjaga kontrol dan integritas data dalam sistem serta menyediakan antarmuka bagi pengelola sistem untuk memonitor dan melakukan pemeliharaan.

2. DetailBookingRoom

   Tabel *DetailBookingRoom* digunakan untuk menyimpan rincian dari setiap pemesanan ruang rapat yang ada di tabel *RoomBookings*. Data yang disimpan dapat meliputi *idbooking*, nomor transaksi, (foreign key ke *RoomBookings*), *room name* (foreign key ke *Rooms*), *booking date, dan shift* (*time slots* per 30 menit). Tanpa tabel DetailBookingRoom, sistem tidak akan dapat menyimpan informasi mendetail mengenai kegiatan yang dilakukan dalam satu pemesanan ruang, sehingga sulit melakukan pelacakan aktivitas rapat.

3. RoomBookings

   Tabel *RoomBookings* menyimpan informasi tentang pemesanan ruang rapat yang dilakukan oleh pengguna. Informasi ini meliputi *id booking, id user, id room*, dan tanggal/waktu pemesanan. Tanpa tabel *RoomBookings*, sistem tidak dapat mencatat transaksi atau memantau ketersediaan ruang rapat. Ini akan menyebabkan ketidakmampuan dalam mengelola ruangan dan memastikan bahwa ruangan tersebut tidak dipesan ganda atau digunakan secara tidak teratur. Fungsi penting dari tabel ini adalah untuk

mencatat setiap pemesanan ruang rapat, sehingga dapat melacak siapa yang memesan dan kapan.

4. Rooms

   Tabel *Rooms* berfungsi untuk menyimpan informasi tentang *rooms* yang tersedia dalam sistem, termasuk id ruangan, nama ruangan, kapasitas dan fasilitas. Tanpa tabel *Rooms*, sistem tidak akan tahu ruangan mana yang tersedia, kapasitasnya, dan fasilitas apa saja yang disediakan. Ini akan menyulitkan pengguna dalam memilih ruang rapat yang sesuai dengan kebutuhan mereka. Fungsi tabel ini adalah untuk memberikan gambaran mengenai ruang rapat yang tersedia dalam sistem, sehingga pemesanan dapat dilakukan berdasarkan informasi yang tepat.

5. DetailBookingTransport

   Tabel *DetailBookingTransport* berfungsi untuk menyimpan rincian dari setiap pemesanan transportasi yang terdapat pada tabel *TransportBookings*. Data yang tersimpan dapat mencakup *bookingid*, nomor transaksi (foreign key ke *TransportBookings*), nomor kendaraan (foreign key ke *Transport*), *booking date dan shift* (*time slots* per 30 menit). Fungsi tabel ini adalah untuk memberikan informasi rinci mengenai setiap pemesanan transportasi, agar sistem dapat mengatur jadwal perjalanan, mengoptimalkan penggunaan kendaraan, serta memantau operasional transportasi dengan lebih baik.

6. TransportBookings

   Tabel *TransportBookings* menyimpan data pemesanan transportasi oleh pengguna, termasuk id pemesanan, id pengguna, id transportasi, dan tanggal/waktu pemesanan. Tabel ini mencatat penggunaan transportasi perusahaan untuk membawa karyawan berdasarkan section atau individu sesuai kebutuhan, termasuk keperluan logistik. Tanpa tabel ini, sistem tidak dapat mencatat atau mengelola pemesanan transportasi, sehingga alokasi kendaraan dan perencanaan perjalanan menjadi tidak teratur. Fungsi utamanya adalah mengelola dan mencatat setiap pemesanan transportasi agar penjadwalan dan penggunaan kendaraan tetap efisien.

7. Transports

   Tabel *Transports* digunakan untuk menyimpan informasi mengenai transportasi yang tersedia, seperti id transportasi, jenis transportasi (mobil, bus, dll.) dan kapasitas yang dapat ditampung. Tanpa tabel *Transports*, sistem tidak akan tahu kendaraan mana yang tersedia, kapasitasnya, dan jenis transportasinya. Tanpa informasi ini, proses pemesanan transportasi menjadi tidak teratur dan tidak dapat dikendalikan dengan baik. Fungsi tabel ini adalah untuk menyediakan informasi mengenai transportasi yang tersedia sehingga pengguna dapat memilih transportasi yang sesuai dengan kebutuhan mereka.

8. Drivers

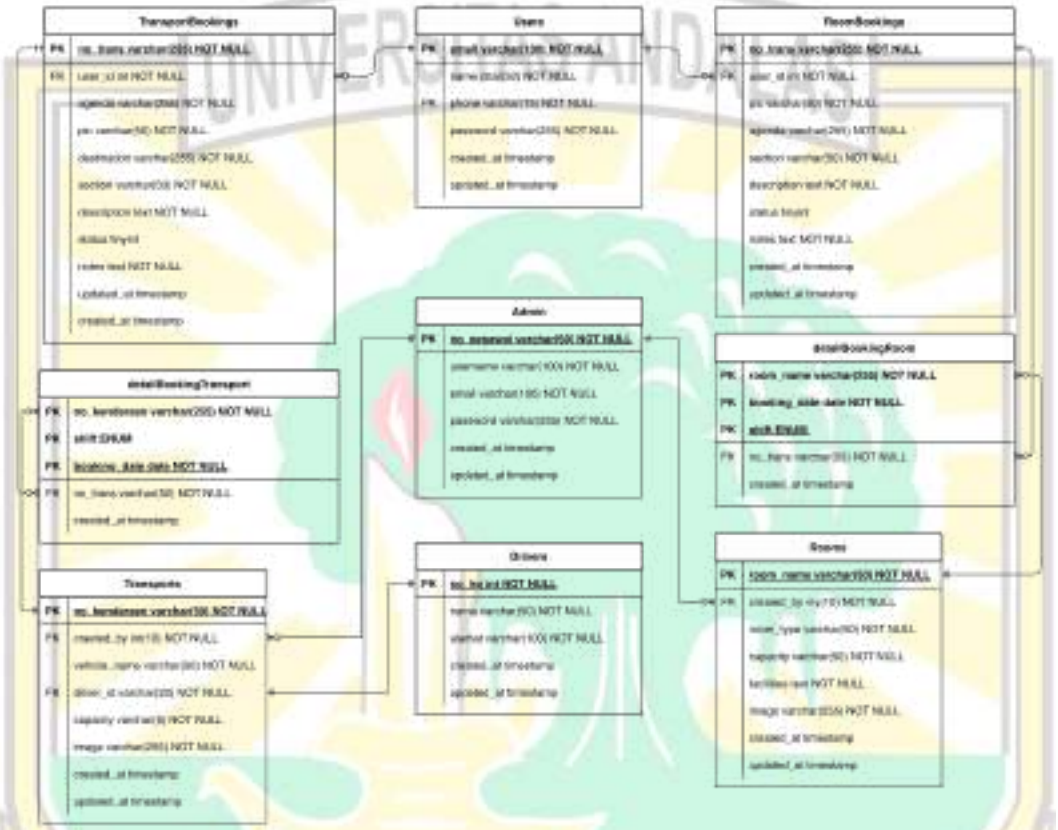
   Tabel Drivers digunakan untuk menyimpan informasi mengenai para pengemudi yang terdaftar, seperti nomor handphone, nama lengkap,dan alamat. Tanpa tabel Drivers, sistem tidak akan dapat mengidentifikasi siapa saja pengemudi yang ada, bagaimana cara menghubungi mereka, dan siapa yang siap untuk bertugas. Tanpa informasi ini, proses penugasan pengemudi ke sebuah pemesanan kendaraan menjadi mustahil dilakukan dan tidak dapat dilacak. Fungsi tabel ini adalah untuk menyediakan daftar terpusat mengenai data pengemudi sehingga sistem dapat menugaskan pengemudi yang tepat untuk setiap perjalanan yang dipesan.

9. Users

   Tabel *Users* menyimpan data tentang pengguna sistem, termasuk id pengguna, nama, alamat email, dan password. Ini memungkinkan pengguna untuk mendaftar, masuk, dan mengakses layanan sistem. Tanpa tabel *Users*, tidak ada cara untuk mengidentifikasi siapa yang menggunakan sistem dan memberikan akses yang berbeda berdasarkan peran pengguna. Ini sangat penting dalam hal keamanan dan manajemen akses dalam sistem. Fungsi penting dari tabel ini adalah untuk menyimpan data pengguna yang memungkinkan pengguna untuk mengakses sistem,

melakukan pemesanan ruang rapat atau transportasi, dan memiliki kontrol terhadap data mereka.

Perancangan database Sistem Informasi *Booking* Ruang Rapat dan Transportasi Berbasis *Mobile* Pada PT SISI dapat dilihat pada Gambar 4.9 berikut.



Gambar 4.9 *Entity Relationship Diagram* (ERD)

4.2.3. Struktur Tabel dan Basis Data

Struktur tabel dan basis data menjelaskan hubungan antar tabel serta atribut-atributnya, termasuk status seperti *Primary Key* (PK) dan *Foreign Key* (FK), beserta nama atribut, tipe data, dan nama tabel. Bagian ini akan membahas struktur tabel dan basis data yang mencakup tabel berikut: Admin, *RoomBookings, Rooms, DetailBookingRoom, DetailBookingTransport, TransportBookings, Transports, Users* dan *Drivers.*

Tabel 4.5 Struktur Tabel Admin

Nama atribut	Tipe data	Keterangan
no_pegawai	voucher(255)	*Primary Key*
name	varchar(50)	
email	varchar(100)	*Primary Key*
password	varchar(255)	
created_at	timestamp	
updated_at	timestamp	

Tabel 4.6 Struktur Tabel *RoomBookings*

Nama atribut	Tipe data	Keterangan
no_trans	varchar(255)	*Primary Key*
user_id	int	*Foreign Key*
pic	varchar(50)	
section	varchar(50)	
agenda	varchar(255)	
description	text	
status	tinyint	
notes	text	
created_at	timestamp	
update_at	timestamp	

Tabel 4.7 Struktur Tabel *DetailBookingRoom*

Nama atribut	Tipe data	Keterangan
booking_id	int(11)	*Primary Key*
no_trans	varchar(255)	*Foreign Key*
room_name	varchar(50)	*Foreign Key*
booking_date	date	
shift	enum	
created_at	timestamp	

Tabel 4.8 Struktur Tabel *DetailBookingTransport*

Nama atribut	Tipe data	Keterangan
booking_id	int(11)	*Primary Key*
no_trans	varchar(255)	*Foreign Key*
no_kendaraan	varchar(255)	*Foreign Key*
booking_date	date	
shift	enum	
created_at	timestamp	

Tabel 4.9 Struktur Tabel *Rooms*

Nama atribut	Tipe data	Keterangan
room_name	int	*Primary Key*
created_by	int(10)	*Foreign Key*
name	varchar(50)	
capacity	varchar(5)	
facilities	text	
image	varchar(255)	
created_at	timestamp	
update_at	timestamp	

Tabel 4.10 Struktur Tabel *TransportBookings*

Nama atribut	Tipe data	Keterangan
no_trans	varchar(255)	*Primary Key*
user_id	int	*Foreign Key*
pic	varchar(50)	
agenda	varchar(255)	
destination	varchar(255)	
section	varchar(50)	
description	text	
status	tinyint	
notes	text	

Tabel 4.11 Struktur Tabel Transports

Nama atribut	Tipe data	Keterangan
no_kendaraan	varchar(255)	*Primary Key*
created_by	int(10)	*Foreign Key*
vehicle_name	varchar(50)	
driver_id	varchar(255)	*Foreign Key*
capacity	varchar(5)	
image	varchar(20)	
created_at	timestamp	
update_at	timestamp	

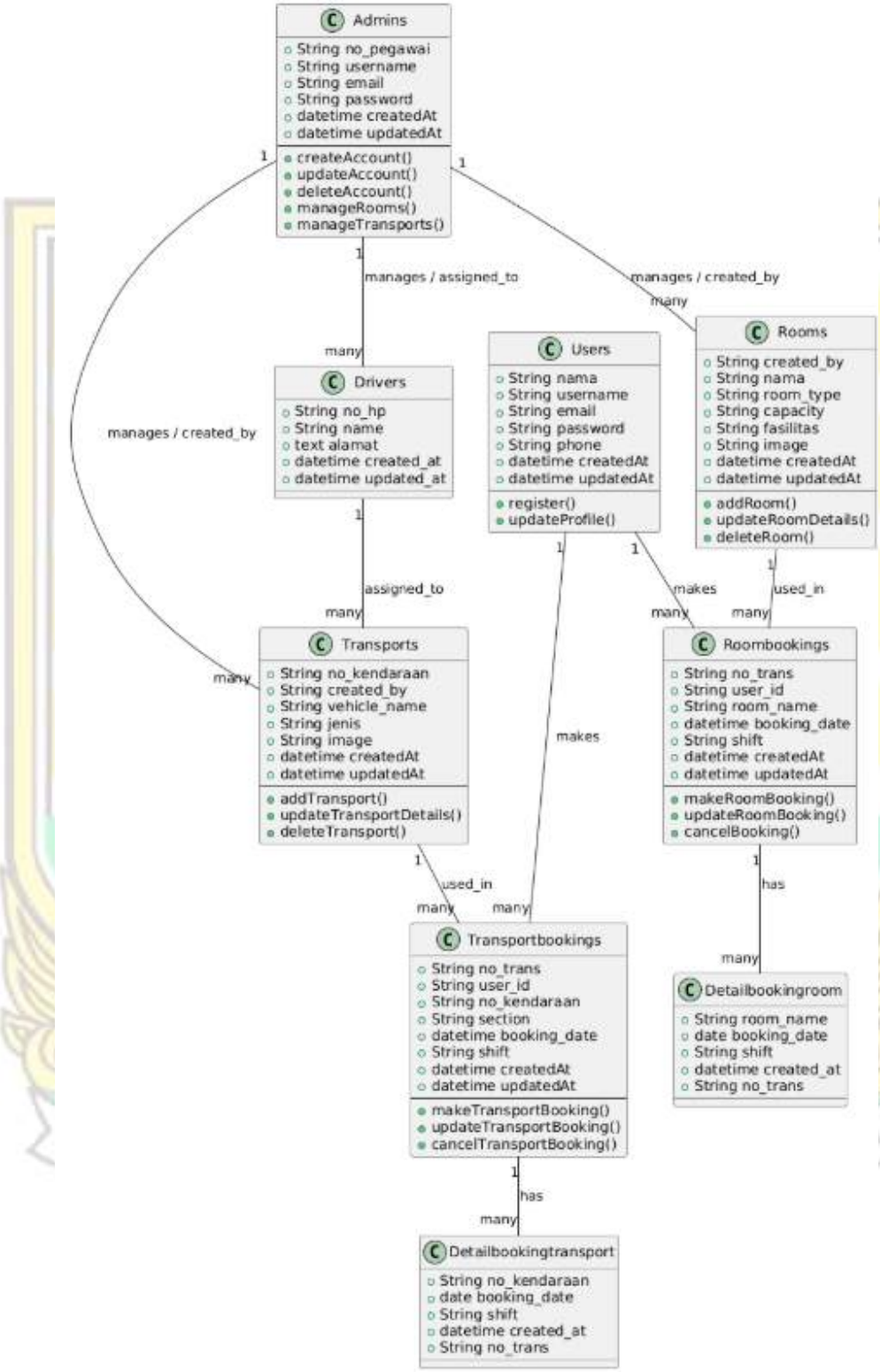
Tabel 4.12 Struktur Tabel Users

Nama atribut	Tipe data	Keterangan
email	varchar(100)	*Primary Key*
name	char(50)	
phone	varchar(15)	*Primary Key*
password	varchar(255)	
created_at	timestamp	
updated_at	timestamp	

4.2.4. Class Diagram

Class diagram merupakan alat pemodelan yang menggambarkan struktur statis sistem dengan menunjukkan kelas-kelas dan hubungan antar kelas. Menurut (Subhiyakto dan Astuti, 2020), class diagram memudahkan pemodelan masalah dan solusi perangkat lunak dengan cara yang terstruktur dan jelas. Diagram ini membantu memvisualisasikan elemen-elemen seperti kelas, atribut, metode, dan hubungan antar kelas dalam sistem. Berikut merupakan class diagram Sistem

Informasi *Booking* Ruang Rapat dan Transportasi Berbasis *Mobile* Pada PT SISI dapat dilihat pada Gambar 4.10 berikut.



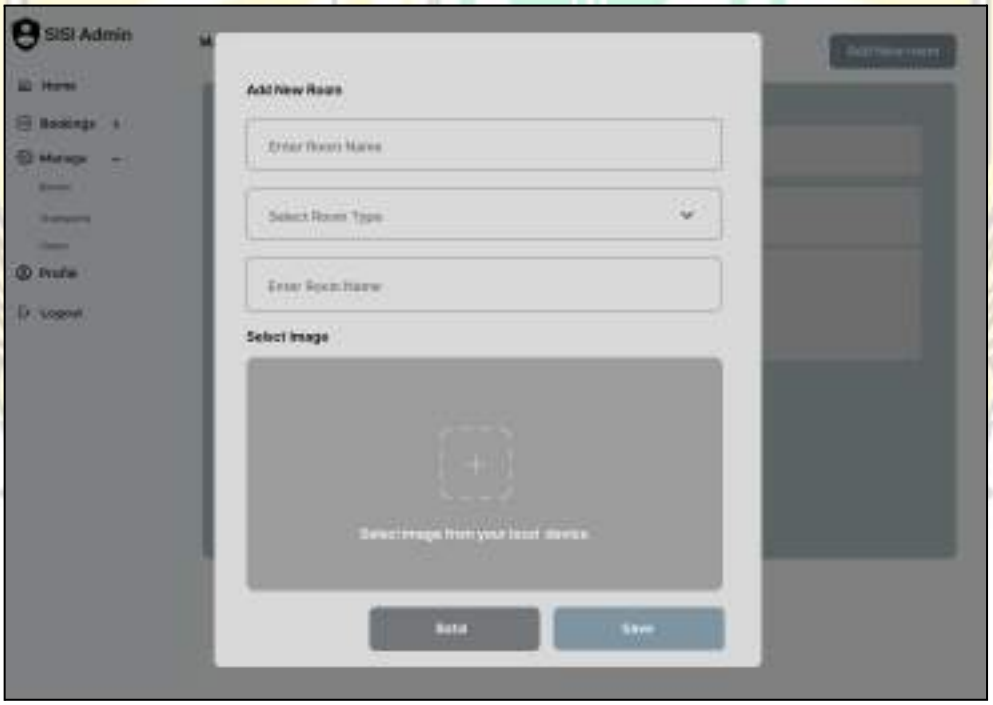
Gambar 4.10 *Class Diagram*

4.2.5. Perancangan Antarmuka

Antarmuka adalah tampilan grafis yang menghubungkan pengguna dengan sistem, yang dikembangkan berdasarkan hasil analisis kebutuhan pengguna. Sub bab ini akan membahas desain antarmuka untuk berbagai fungsi, termasuk menambah data ruangan, *booking* ruang rapat, *reschedule* dan *cancel booking*. Desain antarmuka lainnya dapat dilihat pada Lampiran C.

4.2.5.1. Antarmuka Halaman Menambahkan Data Ruangan

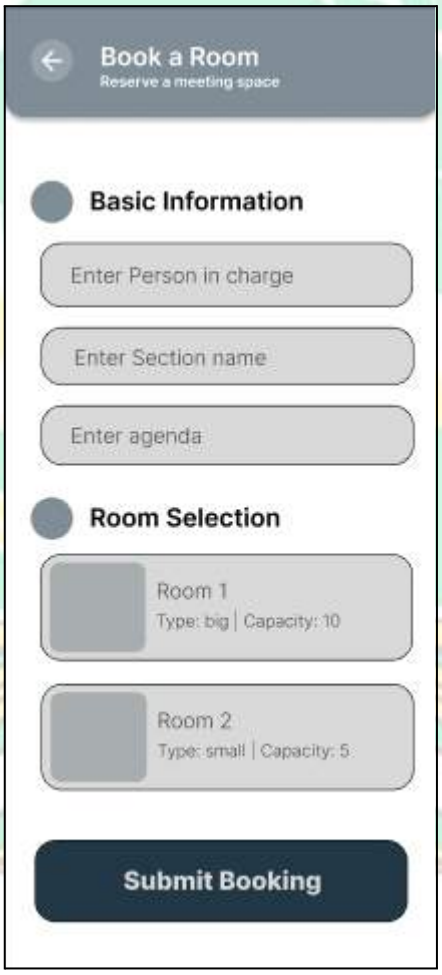
Halaman ini dirancang untuk menambahkan data ruang rapat baru yang hanya dapat diakses dan dilakukan oleh admin. Halaman ini berisi formulir input yang memungkinkan admin untuk memasukkan informasi terkait ruangan yang baru, seperti nama, tipe, dan gambar. Admin dapat memilih jenis ruangan yang sesuai dan mengunggah gambar yang menggambarkan ruangan tersebut. Setelah informasi diisi, admin dapat menyimpan data ruang baru tersebut dengan memencet tombol "*Save*". Fitur ini mempermudah admin dalam mengelola data ruang rapat yang tersedia dalam sistem. Tampilan halaman dapat dilihat pada Gambar 4.11 berikut.



Gambar 4.11 Rancangan Antarmuka Halaman Menambahkan Data Ruangan

4.2.5.2. Antarmuka Halaman Melakukan Booking Ruang Rapat

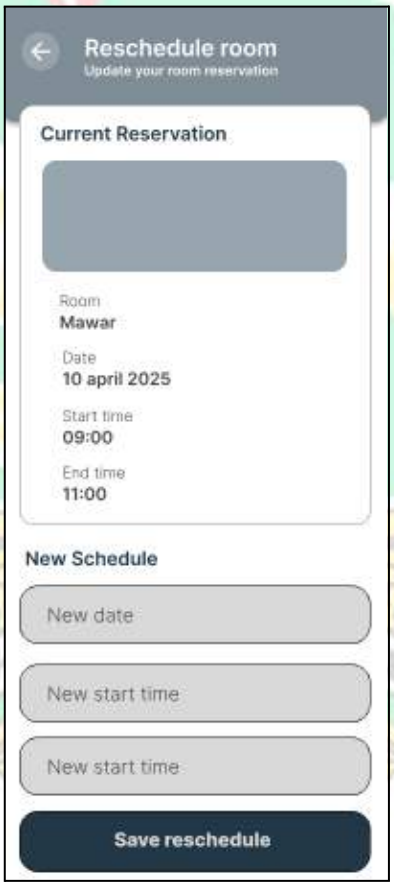
Halaman ini dirancang untuk melakukan *booking* ruang rapat di aplikasi *mobile*. Halaman ini berisi formulir input yang digunakan oleh pengguna untuk melakukan *booking* ruang rapat. Pengguna dapat memilih ruang rapat yang ingin dipesan, menentukan tanggal dan waktu mulai serta waktu selesai. Selain itu, pengguna dapat mengisi informasi tambahan seperti *person in charge, section* dan agenda dari meeting tersebut. Setelah mengisi semua informasi yang dibutuhkan, pengguna dapat mengonfirmasi pemesanan dengan memencet tombol *"Submit"*. Halaman ini dirancang untuk mempermudah proses *booking* ruang rapat secara langsung dari aplikasi *mobile*, memberikan kenyamanan bagi pengguna untuk melakukan pemesanan kapan saja. Tampilan halaman dapat dilihat pada Gambar 4.12 berikut.



Gambar 4.12 Rancangan Antarmuka Halaman *Booking* Ruang Rapat

4.2.5.3. Antarmuka Halaman Melakukan Reschedule Booking

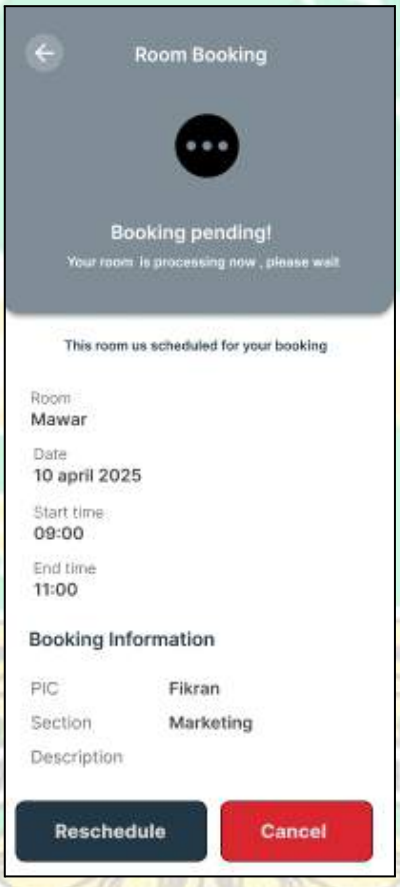
Halaman *Reschedule Booking* di aplikasi *mobile* ini memungkinkan pengguna untuk mengubah tanggal dan waktu *booking* ruang rapat yang sebelumnya telah dilakukan. Halaman ini menampilkan detail informasi *booking* yang sudah ada, seperti nama ruangan, tanggal dan waktu *booking* sebelumnya, serta status *booking*. Pengguna kemudian dapat mengubah tanggal dan waktu sesuai dengan kebutuhan mereka melalui formulir input yang disediakan. Setelah memilih tanggal dan waktu baru, pengguna dapat mengonfirmasi perubahan tersebut dengan memencet tombol *Reschedule*. Fitur ini memberikan fleksibilitas kepada pengguna untuk menyesuaikan jadwal *booking* mereka tanpa perlu melakukan *booking* ulang, sehingga mempermudah proses pengaturan ulang jadwal *booking* ruang rapat. Tampilan halaman dapat dilihat pada Gambar 4.13 berikut.



Gambar 4.13 Rancangan Antarmuka Halaman *Reschedule Booking*

4.2.5.4. Antarmuka Halaman Melakukan Cancel Booking

Halaman *Cancel Booking* ini dapat diakses melalui halaman *Detail Booking* untuk *booking* yang masih berstatus *pending*. Di halaman ini, pengguna dapat melihat rincian lengkap dari *booking* yang telah dilakukan, seperti nama ruang rapat, waktu *booking* , dan status *booking* . Jika pengguna memutuskan untuk membatalkan *booking* , mereka cukup memencet tombol *Cancel* yang tersedia di halaman tersebut. Setelah tombol *Cancel* diklik, status *booking* akan berubah menjadi *Cancelled* dan *booking* tersebut akan dibatalkan dalam sistem. Fitur ini memberikan kemudahan bagi pengguna untuk membatalkan *booking* yang masih dalam status *pending*, sehingga proses pengelolaan ruang rapat dapat tetap berjalan dengan lancar. Tampilan halaman dapat dilihat pada Gambar 4.14 berikut.



Gambar 4.14 Rancangan Antarmuka Halaman *Cancel Booking*

BAB V
IMPLEMENTASI DAN PENGUJIAN SISTEM

Bab ini menjelaskan implementasi dan pengujian sistem berdasarkan hasil analisis serta perancangan yang telah dijabarkan sebelumnya. Implementasi dilakukan dengan mentransformasikan rancangan ke dalam bahasa pemrograman hingga menjadi aplikasi yang siap digunakan. Setelah implementasi, dilakukan proses pengujian untuk mengidentifikasi kekurangan, kesalahan, serta memastikan kesesuaian aplikasi dengan spesifikasi dan kebutuhan yang telah dirancang sebelumnya.

5.1. Implementasi Sistem

Sistem Informasi Booking Ruang Rapat dan Transportasi yang dibangun di penelitian ini merupakan suatu sistem yang diusulkan kepada PT SISI (Sinergi Informatika Semen Indonesia) yang berfungsi untuk mengelola proses booking ruangan rapat dan transportasi perusahaan. Implementasi sistem ini dilakukan dengan menggunakan device dengan spesifikasi seperti berikut:

1. Laptop dengan *processor* 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz 3.11 GHz
2. *Random Access Memory* (RAM) 8 GB.
3. *Solid State Drive* (SSD) dengan kapasitas 512 GB.

Spesifikasi perangkat lunak yang digunakan untuk implementasi sistem ini adalah sebagai berikut.

1. Sistem operasi Windows 11.
2. *Web browser* Brave versi 1.78
3. Code editor Visual Studio Code
4. Bahasa pemrograman *Typescript* dan *Framework Front-End* React Native versi 0.72.5, Next.js versi 14.2.5, React versi 18.2.0 , Express JS versi 4.18.3 dan Node JS versi 22.14.0, TailwindCSS versi 3.4.1.
5. ORM(*Object-Relational Mapping*) Sequelize versi 6.32.1
6. *Database* MySQL

5.1.1. Pengkodean Program

Pada bagian ini, akan dijelaskan implementasi kode program yang dikembangkan berdasarkan rancangan dan arsitektur aplikasi yang telah dijabarkan sebelumnya. Kode program ini dibangun menggunakan *Framework Express.js*. Kode program lainnya yang mendukung implementasi ini akan dijelaskan lebih rinci pada Lampiran D.

5.1.1.1. Kode Program Routing

Pada bagian ini membahas tentang *routing*, yaitu proses penting dalam aplikasi yang berfungsi untuk menangani dan mengatur arah permintaan dari pengguna. Potongan kode program routing untuk aplikasi ini dapat dilihat pada Gambar 5.1 berikut.

```
1
2  // src/routes/roomBookingRoutes.ts
3  import { Router } from 'express';
4  import combinedAuthMiddleware from '../middlewares/combinedAuthMiddleware';
5  import adminMiddleware from '../middlewares/adminMiddleware';
6  import RoomBookingController from '../controllers/roomBookingController';
7
8  const router = Router();
9
10 router.post('/', combinedAuthMiddleware, RoomBookingController.createBooking);
11
12 router.get('/', combinedAuthMiddleware, RoomBookingController.getAllBookings);
13
14 router.get('/:id', combinedAuthMiddleware, RoomBookingController.getBookingById);
15
16 router.put('/:id', combinedAuthMiddleware, RoomBookingController.updateBooking);
17
18 router.delete('/:id', adminMiddleware, RoomBookingController.deleteBooking);
19
20 export default router;
21
22
```

Gambar 5.1 Potongan Kode Program Routing

5.1.1.2. Kode Program Controller

Pada bagian ini membahas tentang *controller*, *controller* adalah bagian kode program yang mengatur logika untuk menangani permintaan user melalui endpoint yang telah ditentukan dalam routing. Saat endpoint diakses, permintaan akan diarahkan ke controller yang sesuai untuk diproses. Potongan kode program controller untuk aplikasi ini dapat dilihat pada Gambar 5.2 berikut.

TA-APP - roomBookingController.ts

```ts
// src/controllers/RoomBookingController.ts
import { Request, Response } from 'express';
import RoomBookingService from '../services/RoomBookingService';

class RoomBookingController {
  public static async createBooking(req: Request, res: Response) {
    try {
      const booking = await RoomBookingService.createBooking({
        ...req.body,
        user_id: (req as any).user.id,
      });
      res.status(201).json(booking);
    } catch (error: any) {
      res.status(400).json({ error: error.message });
    }
  }
```

Gambar 5.2 Potongan Kode Program *Controller*

Pada proses *create booking* dibuat menggunakan Express di Node.js. Kode ini mengimpor Request dan Response dari Express, serta *RoomBookingService* yang berfungsi untuk mengelola pemesanan ruang. Di dalam kelas *RoomBookingController*, terdapat method statis *createBooking* yang menerima permintaan dari pengguna dan memanggil fungsi *createBooking* dari *RoomBookingService* untuk menyimpan data pemesanan ruang. Setelah pemesanan berhasil, server mengirimkan respons dengan status 201 dan data pemesanan. Jika terjadi kesalahan, server mengirimkan respons dengan status 400 beserta pesan *error*.

5.1.1.3. Kode Program *Model*

Model adalah sebuah representasi dari entitas pada sebuah tabel di database yang berinteraksi dengan Controller. Potongan kode program model untuk aplikasi ini dapat dilihat pada Gambar 5.3 berikut.

Gambar 5.3 Potongan Kode Program *Model*

5.1.1.4. Kode Program Tampilan

Pada bagian ini membahas tentang tampilan, yaitu proses penting dalam merepresentasikan visual halaman admin dan halaman mobile untuk menampilkan data yang dikirimkan oleh Controller dari Model. Potongan kode program tampilan untuk aplikasi ini dapat dilihat pada Gambar 5.4 berikut.

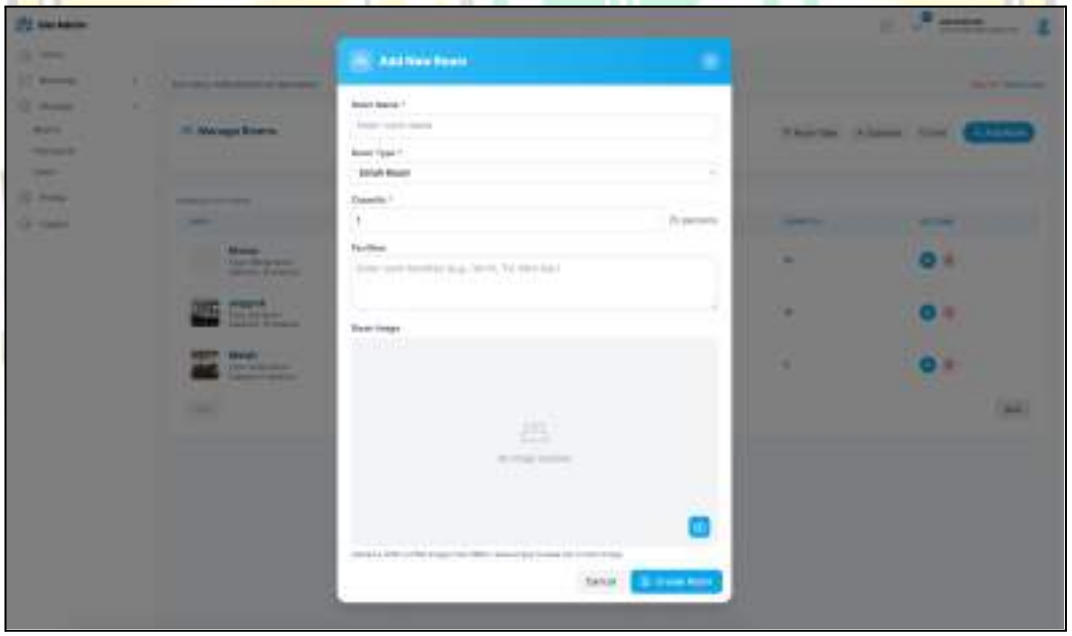
Gambar 5.4 Potongan Kode Program Tampilan *Booking* Ruang Rapat

5.1.2. Implementasi Antarmuka Aplikasi

Implementasi antarmuka sistem menggambarkan tampilan dari aplikasi yang telah dibangun. Sistem informasi ini memiliki beberapa halaman yang diakses sesuai dengan rolenya. Pada bagian ini dijelaskan menambah data ruangan, *booking* ruang rapat, *reschedule* dan *cancel booking*. Implementasi antarmuka lainnya dapat dilihat pada Lampiran E.

5.1.2.1. Antarmuka Halaman Menambahkan Data Ruangan

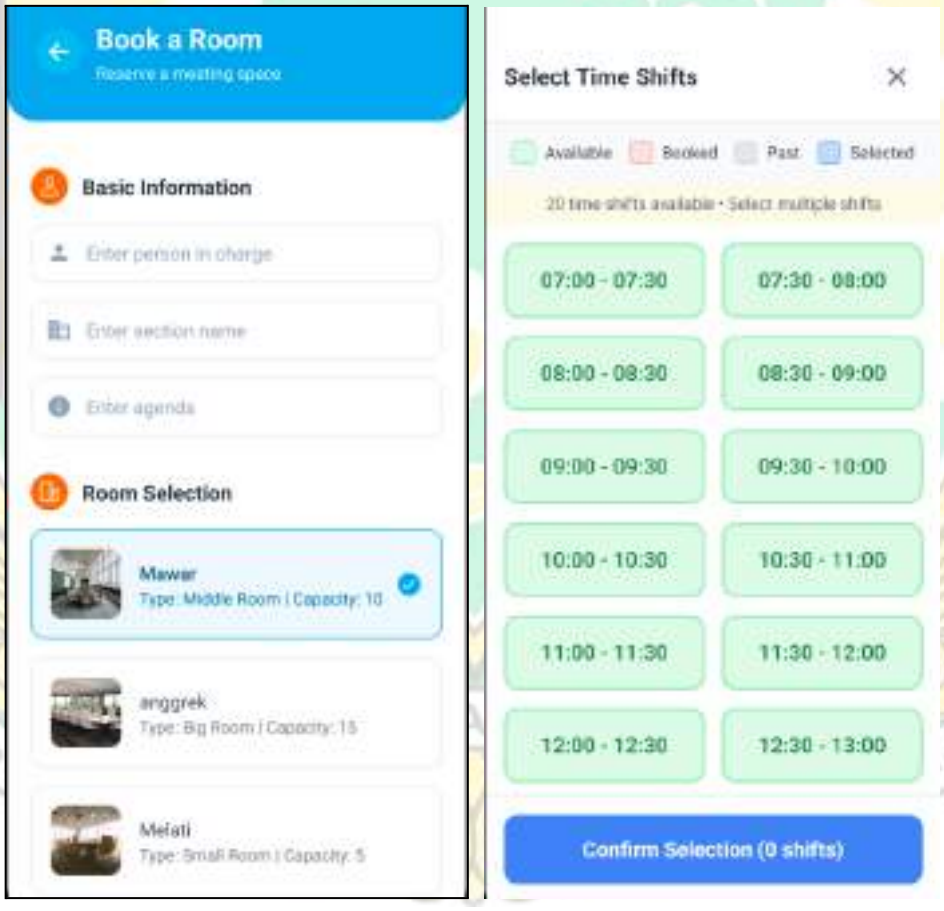
Antarmuka pada tambah data ruang rapat baru yang hanya dapat diakses dan dilakukan oleh admin. Halaman ini berisi formulir input yang memungkinkan admin untuk memasukkan informasi terkait ruangan yang baru, seperti nama, tipe, dan gambar. Admin dapat memilih jenis ruangan yang sesuai dan mengunggah gambar yang menggambarkan ruangan tersebut. Setelah informasi diisi, admin dapat menyimpan data ruang baru tersebut dengan memencet tombol "*Save*". Fitur ini mempermudah admin dalam mengelola data ruang rapat yang tersedia dalam sistem. Tampilan dari halaman *booking* ruang rapat dapat dilihat pada Gambar 5.5 berikut.



Gambar 5.5 Halaman Menambahkan Data Ruangan

5.1.2.2. Antarmuka Halaman *Booking* Ruang Rapat

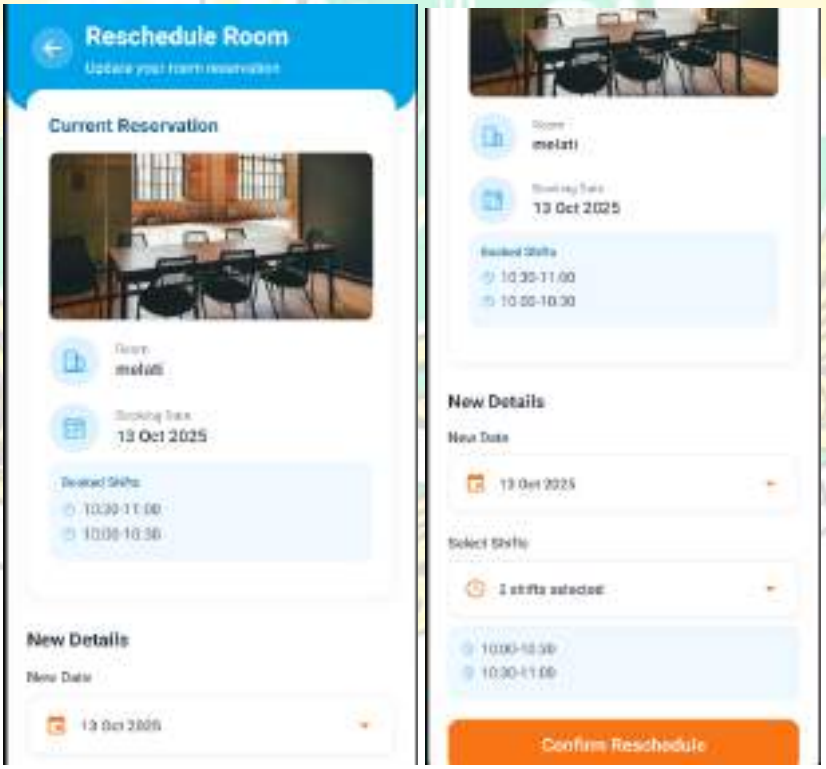
Halaman ini berisi formulir input yang digunakan oleh karyawan untuk melakukan *booking* ruang rapat. Karyawan dapat memilih ruangan yang ingin dipesan, menentukan tanggal dan waktu mulai serta waktu selesai. Selain itu, pengguna dapat mengisi informasi tambahan seperti *person in charge, section* dan agenda dari meeting tersebut. Setelah mengisi semua informasi yang dibutuhkan, pengguna dapat mengonfirmasi pemesanan dengan memencet tombol *"Submit"*. Halaman ini dirancang untuk mempermudah proses *booking* ruang rapat secara langsung dari aplikasi *mobile*, memberikan kenyamanan bagi pengguna untuk melakukan pemesanan kapan saja.. Tampilan dari halaman *booking* ruang rapat dapat dilihat pada Gambar 5.6 berikut.



Gambar 5.6 Halaman *Booking* Ruang Rapat

5.1.2.3. Antarmuka Halaman Reschedule Booking

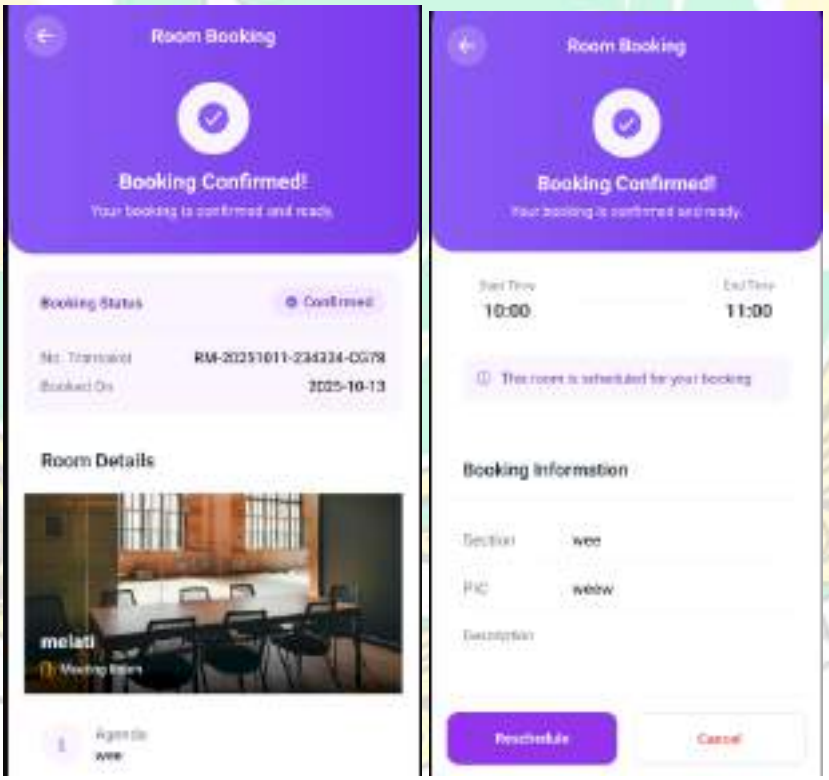
Halaman *Reschedule Booking* di aplikasi *mobile* ini memungkinkan pengguna untuk mengubah tanggal dan waktu *booking* ruang rapat yang sebelumnya telah dilakukan. Halaman ini menampilkan detail informasi *booking* yang sudah ada, seperti nama ruangan, tanggal dan waktu *booking* sebelumnya, serta status *booking*. Pengguna kemudian dapat mengubah tanggal dan waktu sesuai dengan kebutuhan mereka melalui formulir input yang disediakan. Setelah memilih tanggal dan waktu baru, pengguna dapat mengonfirmasi perubahan tersebut dengan memencet tombol *Reschedule*. Fitur ini memberikan fleksibilitas kepada pengguna untuk menyesuaikan jadwal *booking* mereka tanpa perlu melakukan *booking* ulang, sehingga mempermudah proses pengaturan ulang jadwal *booking* ruang rapat. Tampilan dari halaman *booking* ruang rapat dapat dilihat pada Gambar 5.7 berikut.



Gambar 5.7 Halaman *Reschedule Booking* Ruang Rapat

5.1.2.4. Antarmuka Halaman Cancel Booking

Halaman *Cancel Booking* ini dapat diakses melalui halaman *Detail Booking* untuk *booking* yang masih berstatus *pending*. Di halaman ini, pengguna dapat melihat rincian lengkap dari *booking* yang telah dilakukan, seperti nama ruang rapat, waktu *booking* , dan status *booking* . Jika pengguna memutuskan untuk membatalkan *booking* , mereka cukup memencet tombol *Cancel* yang tersedia di halaman tersebut. Setelah tombol *Cancel* diklik, status *booking* akan berubah menjadi *Cancelled* dan *booking* tersebut akan dibatalkan dalam sistem. Fitur ini memberikan kemudahan bagi pengguna untuk membatalkan *booking* yang masih dalam status *pending*, sehingga proses pengelolaan ruang rapat dapat tetap berjalan dengan lancar. Tampilan dari halaman *booking* ruang rapat dapat dilihat pada Gambar 5.8 berikut.



Gambar 5.8 Halaman *Cancel Booking* Ruang Rapat

5.2. Pengujian Sistem

Pengujian sistem merupakan langkah penting untuk memastikan aplikasi sesuai dengan desain yang direncanakan. Dengan menggunakan metode *black box testing*, fungsionalitas aplikasi Sistem Informasi *Booking* Ruang Rapat dan Transportasi Berbasis *Mobile* Pada PT SISI Untuk hasil pengujian yang lebih lengkap dapat dilihat pada Lampiran F.

5.2.1. Fokus Pengujian

Berdasarkan fungsionalitas aplikasi Sistem Informasi *Booking* Ruang Rapat dan Transportasi Berbasis *Mobile* Pada PT SISI, tahap pengujian ini akan berfokus pada 15 item uji yang dijelaskan secara rinci dalam Tabel 5.1 berikut.

Tabel 5.1 Tabel Fokus Pengujian

No	Item Uji	User	Detail Pengujian
1	Login *back office*	Admin	Login
3	Dashboard	Admin	View
4	Manage Room	Admin	View, Add, Delete, dan Edit
5	Manage Transportasi	Admin	View, Add, Delete, dan Edit
6	Manage User	Admin	View, Add, Delete, dan Edit
7	Booking	Admin	View, Delete dan Export
8	Login aplikasi *mobile*	Karyawan	Login
10	Home	Karyawan	View
11	Booking Room	Karyawan	Add, View, Reschedule, dan Cancel
12	Booking Transportasi	Karyawan	Add, View, Reschedule, dan Cancel
13	Explore Room dan Transportasi	Karyawan	View

5.2.2. Kasus Hasil Pengujian

Pada bagian ini, akan dibahas berbagai kasus serta hasil yang diperoleh dari pengujian sistem yang telah dilakukan. Pengujian dilakukan sesuai dengan fokus yang telah ditentukan sebelumnya, dengan memeriksa input yang dimasukkan ke dalam sistem dan output yang dihasilkan. Subbab ini akan menjelaskan tiga kasus hasil pengujian, yaitu pengujian tambah data ruang rapat, pengujian tambah data *booking* ruang rapat dan pengujian *reschedule booking* ruang rapat.

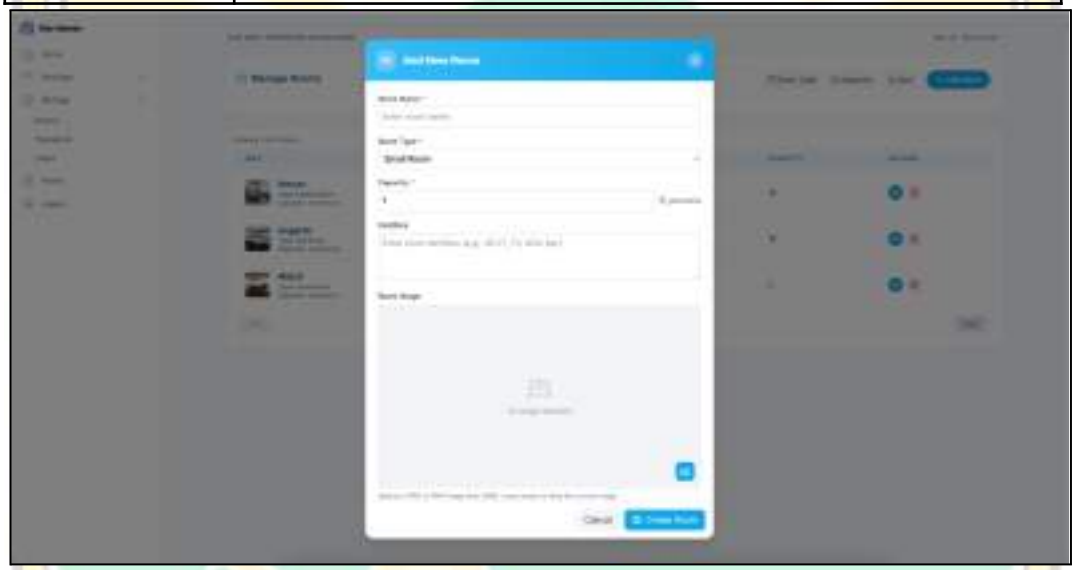
5.2.2.1. Pengujian Tambah Data Ruang Rapat

Pengujian penambahan data ruang rapat ini dilakukan oleh admin setelah berhasil login ke dalam *website back office*. Tujuannya adalah memastikan bahwa fitur penambahan data ruang rapat berfungsi sesuai harapan. Hasil pengujian menunjukkan bahwa data ruang rapat berhasil disimpan ke dalam *database*, dan sistem menampilkan popup pemberitahuan bahwa data ruang rapat berhasil disimpan. Hasil pengujian dapat dilihat pada Tabel 5.2 berikut.

Proses penambahan data dimulai ketika admin perlu menambahkan ruang rapat baru, kemudian mencatatnya di halaman *form* ruang rapatdalam sistem. Selanjutnya, admin mengisi informasi yang diperlukan seperti nama ruangan, kapasitas, dan fasilitas ruangan lalu mengklik tombol Simpan ruang rapat yang terletak di bagian kanan bawah halaman. Proses pengujian tambah data ruang rapat ini dapat dilihat pada Gambar 5.9 berikut.

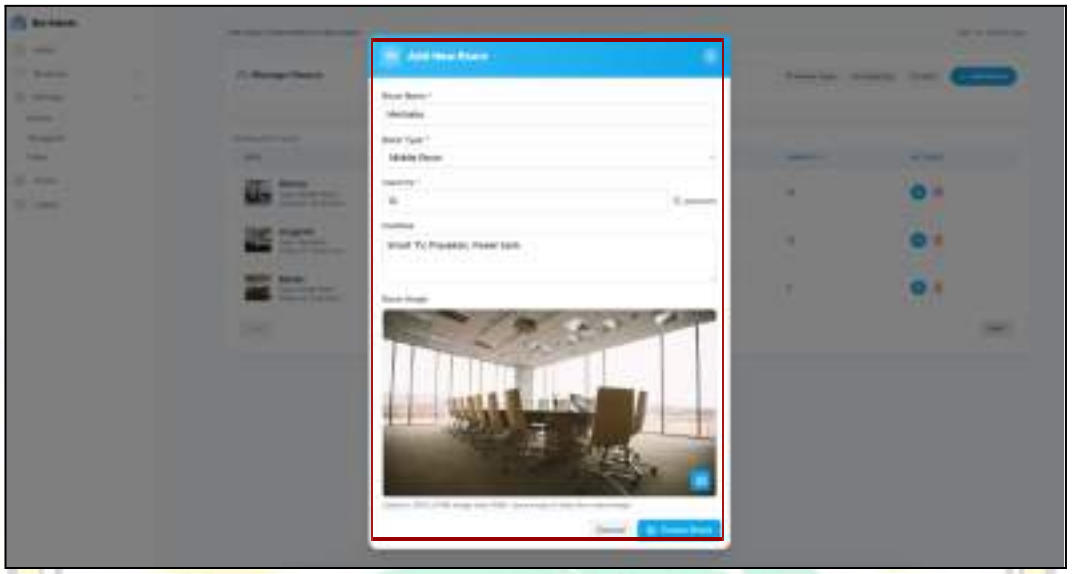
Tabel 5.2 Hasil Pengujian Tambah Ruang Rapat (Benar)

Kasus Hasil Pengujian (Benar)	
Data masukan	Semua data yang diperlukan pada *form Room,* yaitu *room name, room type, capacity* dan *facilities.*
Hal yang diharapkan	Data tersimpan ke dalam *database* dan sistem menampilkan *popup* "*Room created successfully!*"
Pengamatan	Sistem berhasil menyimpan data ke dalam *database* dan menampilkan *popup* "*Room created successfully*!"
Hasil	Sesuai



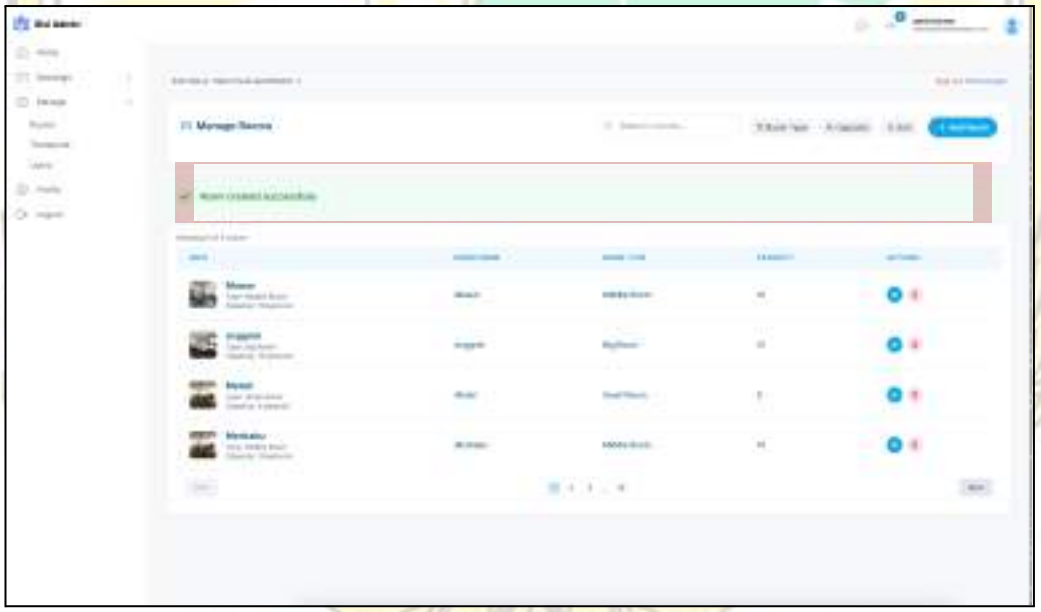
Gambar 5.9 Pengujian Tambah Data Ruang Rapat

Sistem akan menampilkan halaman f*orm add new room*. Kemudian karyawan mengisikan data yang diperlukan yaitu *room name*, *room type*, *capacity facilities* dan *room image*. Proses pengujian pengisian data dapat dilihat pada Gambar 5.10 berikut.



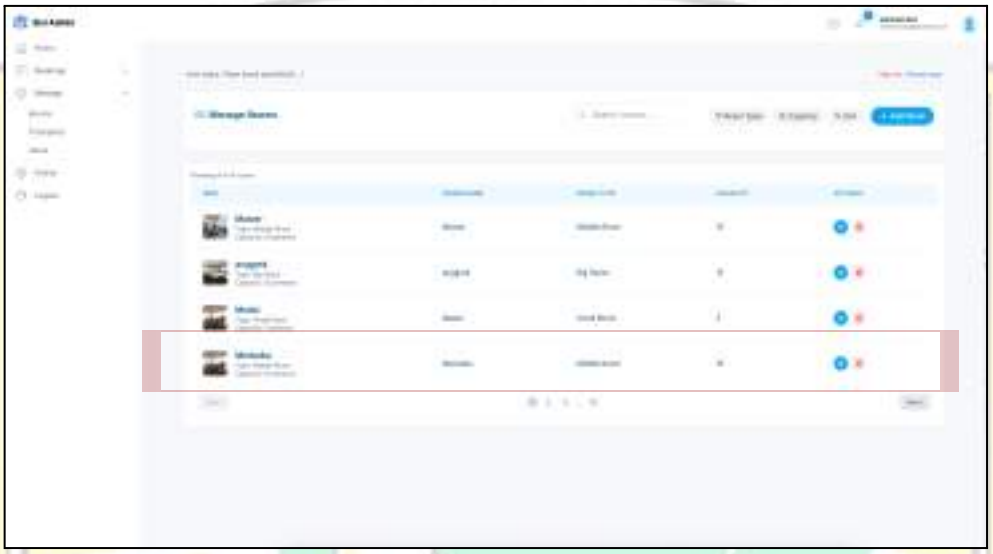
Gambar 5.10 Pengujian Pengisian Data Ruang Rapat

Pada saat semua data yang diisikan sudah sesuai, admin mengklik tombol "*Create Room*". Sistem akan menampilkan popup "*Room Created Successfully!*" Data telah berhasil ditambahkan ke dalam database. Tampilan proses pengujian tambah data *room* (benar) dapat dilihat pada Gambar 5.11 berikut.



Gambar 5.11 Pengujian Tambah Data Ruang Rapat *(*Benar)

Data ruang rapat yang ditambahkan telah berhasil tersimpan ke dalam database dan ditampilkan di halaman *manage rooms* dengan data yang telah ditambahkan itu. Tampilan data ruang rapat yang baru saja ditambahkan ke halaman *manage rooms* dapat dilihat pada Gambar 5.12, serta bukti data berhasil disimpan pada database di tabel *rooms*, dapat dilihat pada Gambar 5.13 berikut.



Gambar 5.12 *Output* Tambah Data Ruang Rapat

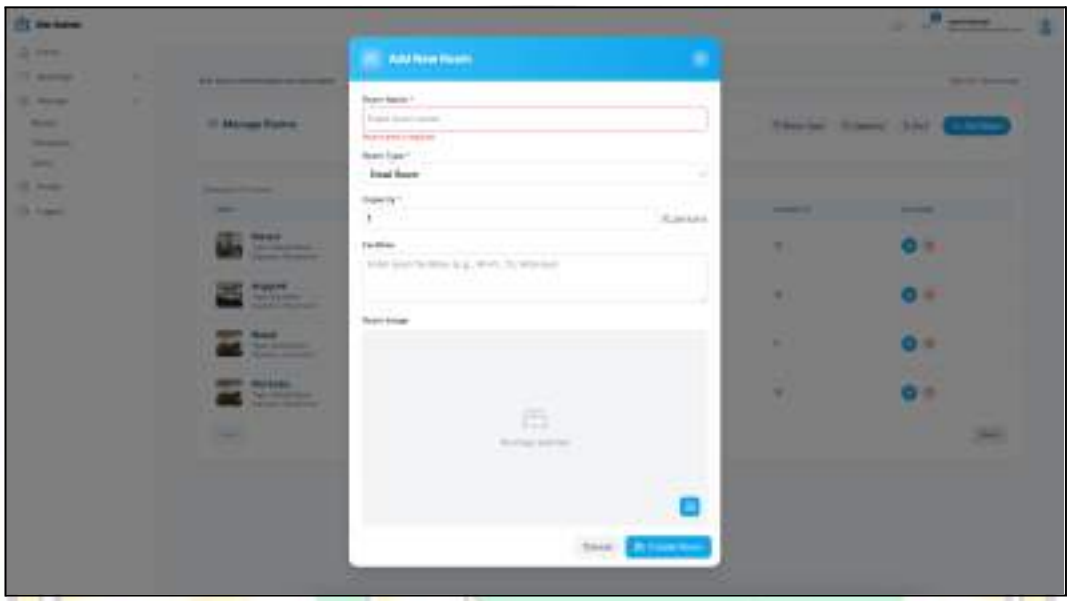
Gambar 5.13 Data *Booking* Ruang Rapat Tersimpan di *Database*

Sedangkan untuk hasil pengujian alternatif terhadap fungsional tambah data *booking* ruang rapat dapat dilihat pada Tabel 5.3 berikut.

Tabel 5.3 Hasil Pengujian Tambah Ruang Rapat (Alternatif)

Kasus Hasil Pengujian (Alternatif)	
Data masukan	Tidak mengisi data yang wajib diisi.
Hal yang diharapkan	Sistem menampilkan pesan kesalahan pada *form.*
Pengamatan	Sistem menampilkan pesan kesalahan pada *form.*
Hasil	Sesuai

Pengujian alternatif ini dilakukan dengan mengklik tombol “Create Room” sebelum semua data wajib terisi diisikan oleh admin, kemudian sistem akan menampilkan pesan kesalahan pada form. Hasil pengujian tambah data ruang rapat (alternatif) dapat dilihat pada Gambar 5.14 berikut.



Gambar 5.14 *Output* Pengujian Tambah Data Ruang Rapat *(Alternatif)*

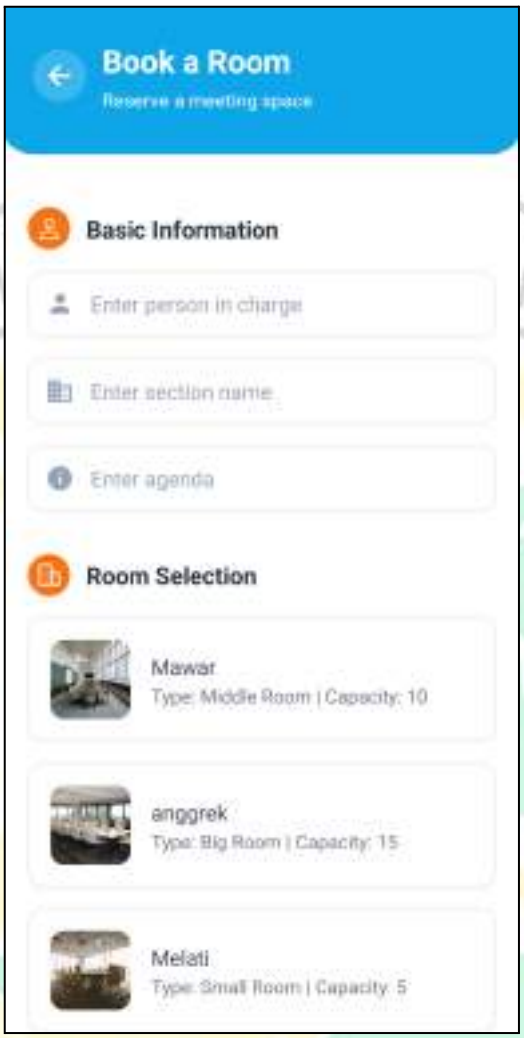
5.2.2.2. Pengujian Tambah Data *Booking* Ruang Rapat

Pengujian penambahan data *booking* ruang rapat ini dilakukan oleh karyawan setelah berhasil login ke dalam aplikasi *mobile*. Tujuannya adalah memastikan bahwa fitur pemesanan ruang rapat berfungsi sesuai harapan. Hasil pengujian menunjukkan bahwa data *booking* ruang rapat berhasil disimpan ke dalam *database*, dan aplikasi menampilkan popup pemberitahuan bahwa pemesanan ruang rapat berhasil disimpan. Hasil pengujian dapat dilihat pada Tabel 5.4 berikut.

Tabel 5.4 Hasil Pengujian Tambah *Booking* Ruang Rapat (Benar)

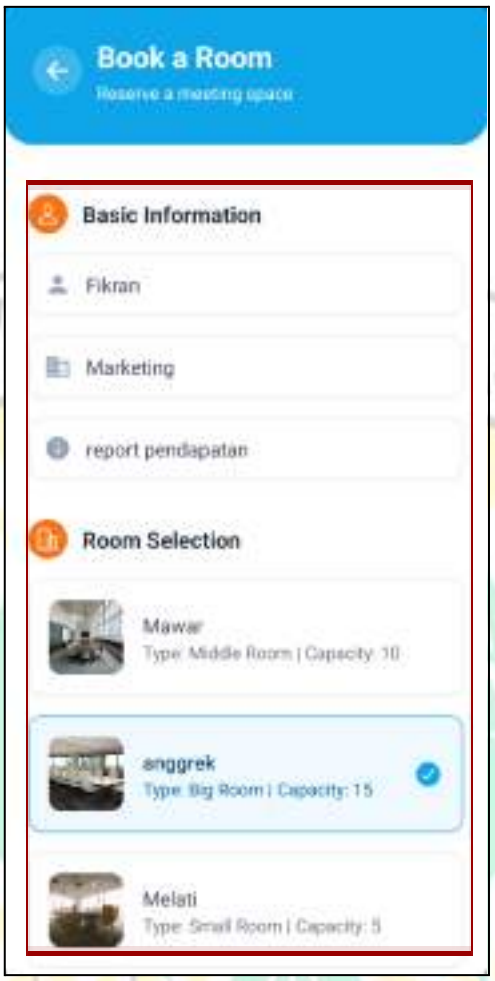
Kasus Hasil Pengujian (Benar)	
Data masukan	Semua data yang diperlukan pada *form booking,* yaitu tanggal, waktu mulai, waktu selesai, identitas dan agenda rapat.
Hal yang diharapkan	Data tersimpan ke dalam *database* dan sistem menampilkan popup "*Booking submitted successfully!*!"
Pengamatan	Sistem berhasil menyimpan data ke dalam *database* dan menampilkan *popup* "*Booking submitted successfully!*!!"
Hasil	Sesuai

Proses pemesanan ruang rapat dimulai ketika karyawan perlu memesan ruangan untuk rapat, kemudian mengakses halaman *Booking* Ruang Rapat dalam aplikasi *mobile*. Selanjutnya, karyawan mengisi informasi yang diperlukan seperti tanggal rapat, waktu mulai, waktu selesai, identitas dan agenda rapat lalu mengklik tombol *Booking Now* yang terletak di bagian bawah halaman. Proses pengujian tambah data *booking* ruang rapat ini dapat dilihat pada Gambar 5.15 berikut.



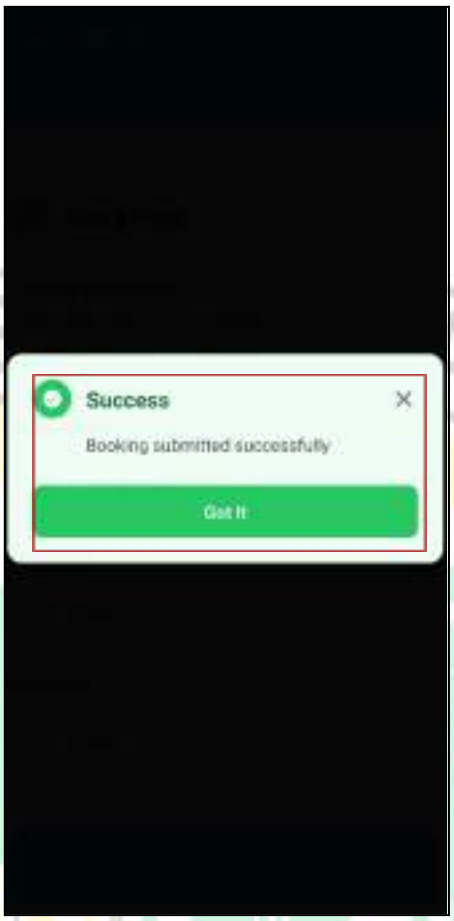
Gambar 5.15 Pengujian Tambah Data *Booking* Ruang Rapat

Sistem akan menampilkan halaman form *booking* ruang rapat. Kemudian karyawan mengisikan data yang diperlukan yaitu *person in charge*, *section name*, agenda, *room*, tanggal, waktu mulai dan waktu selesai. Proses pengujian pengisian data dapat dilihat pada Gambar 5.16 berikut.



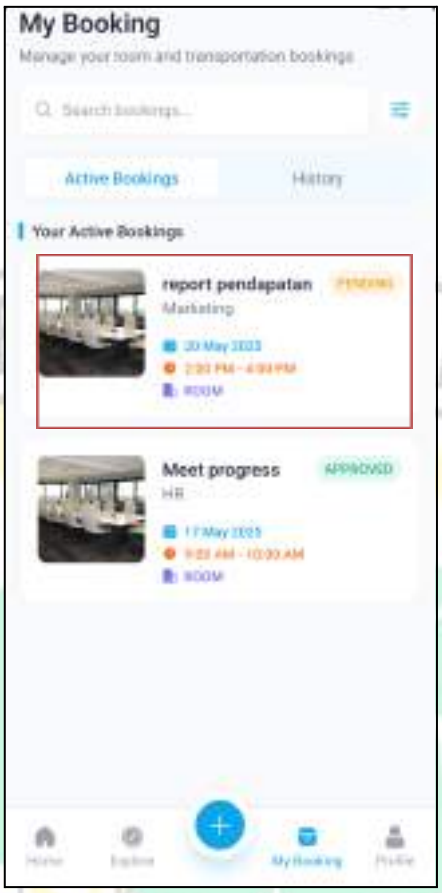
Gambar 5.16 Pengujian Pengisian Data *Booking* Ruang Rapat

Pada saat semua data yang diisikan sudah sesuai, karyawan mengklik tombol "*Submit Booking*". Sistem akan menampilkan popup "*Booking Submitted Successfully!*" Data telah berhasil ditambahkan ke dalam database. Tampilan proses pengujian tambah data *booking* ruang rapat (benar) dapat dilihat pada Gambar 5.17 berikut.



Gambar 5.17 Pengujian Tambah Data *Booking* Ruang Rapat *(*Benar)

Data *booking* ruang rapat yang ditambahkan telah berhasil tersimpan ke dalam database dan ditampilkan di halaman *my booking* dengan data yang telah ditambahkan itu. Tampilan data *booking* ruang rapat yang baru saja ditambahkan ke halaman *my booking* dapat dilihat pada Gambar 5.18, serta bukti data berhasil disimpan pada database di tabel *roombookings*, dapat dilihat pada Gambar 5.19 berikut.



Gambar 5.18 *Output* Tambah Data *Booking* Ruang Rapat

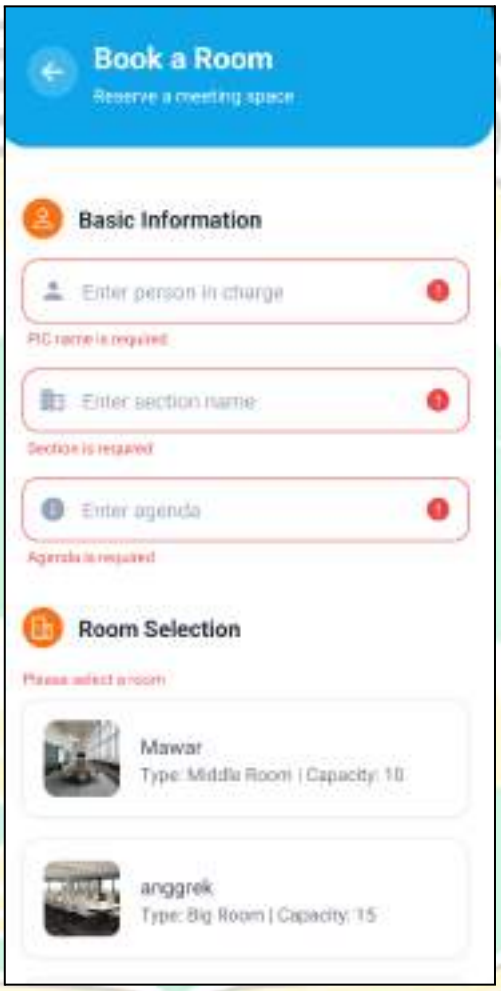
Gambar 5.19 Data *Booking* Ruang Rapat Tersimpan di *Database*

Sedangkan untuk hasil pengujian alternatif terhadap fungsional tambah data *booking* ruang rapat dapat dilihat pada Tabel 5.5 berikut.

Tabel 5.5 Hasil Pengujian Tambah *Booking* Ruang Rapat (Alternatif)

Kasus Hasil Pengujian (Alternatif)	
Data masukan	Tidak mengisi data yang wajib diisi.
Hal yang diharapkan	Sistem menampilkan pesan kesalahan pada *form.*
Pengamatan	Sistem menampilkan pesan kesalahan pada *form.*
Hasil	Sesuai

Pengujian alternatif ini dilakukan dengan mengklik tombol "*Submit Booking*" sebelum semua data wajib terisi diisikan oleh admin, kemudian sistem akan menampilkan pesan kesalahan pada form. Hasil pengujian tambah data *booking* ruang rapat (alternatif) dapat dilihat pada Gambar 5.20 berikut.



Gambar 5.20 *Output* Pengujian Tambah Data *Booking* Ruang Rapat *(Alternatif)*

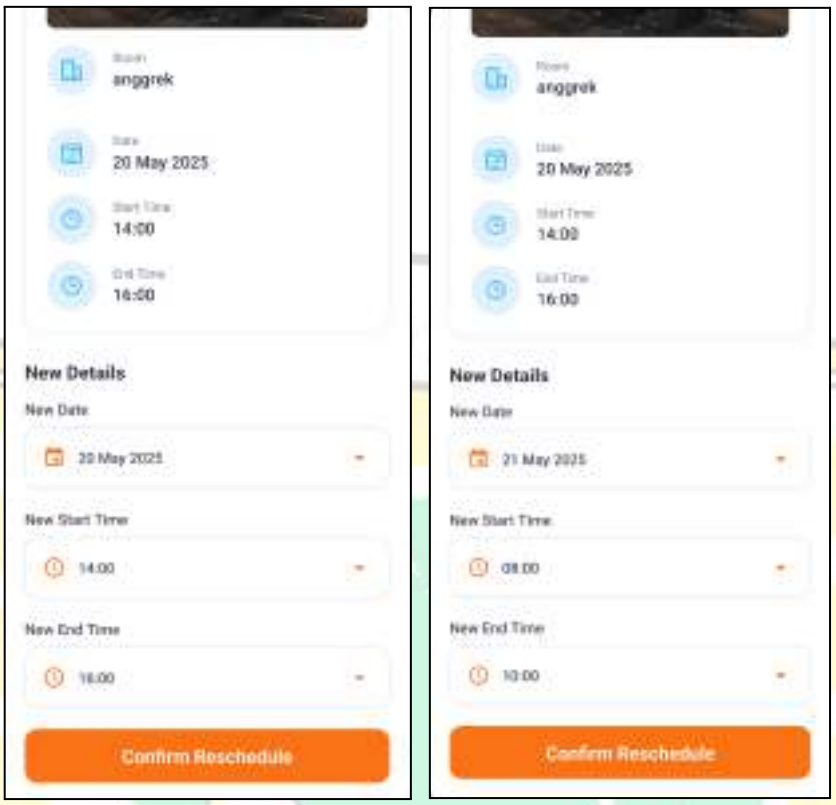
5.2.2.3. Pengujian *Reschedule Booking* Ruang Rapat

Pengujian *reschedule booking* ruang rapat ini dilakukan oleh karyawan setelah berhasil login ke dalam aplikasi *mobile*. Tujuannya adalah memastikan bahwa fitur penjadwalan ulang ruang rapat berfungsi sesuai harapan. Hasil pengujian menunjukkan bahwa data *reschedule booking* ruang rapat berhasil disimpan ke dalam *database*, dan aplikasi menampilkan popup pemberitahuan bahwa penjadwalan ulang ruang rapat berhasil disimpan. Hasil pengujian dapat dilihat pada Tabel 5.6 berikut.

Tabel 5.6 Hasil Pengujian Tambah Ruang Rapat (Benar)

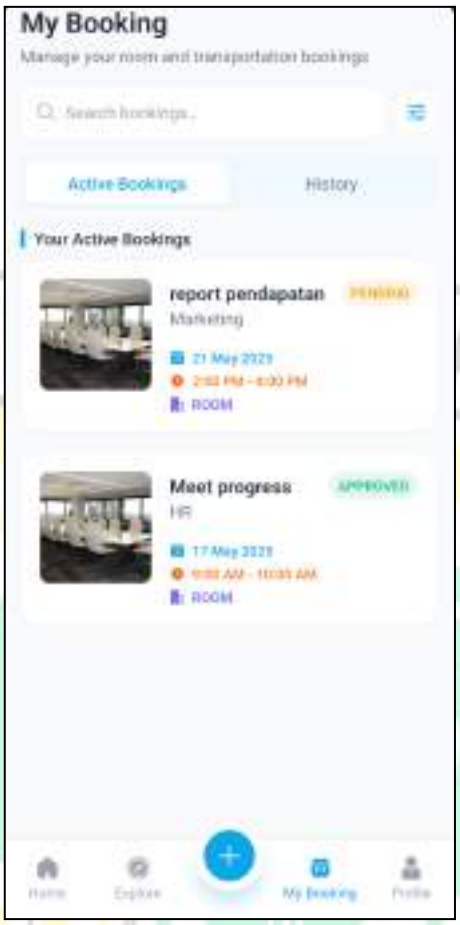
Kasus Hasil Pengujian (Benar)	
Data masukan	Semua data yang diperlukan pada *form booking,* yaitu tanggal, waktu mulai dan waktu selesai
Hal yang diharapkan	Data tersimpan ke dalam *database* dan sistem menampilkan popup "*Booking Reschedule successfully!*!"
Pengamatan	Sistem berhasil menyimpan data ke dalam *database* dan menampilkan *popup* "*Booking Reschedule successfully!*!!"
Hasil	Sesuai

Proses penjadwalan ulang ruang rapat dimulai ketika karyawan perlu mengubah jadwal ruangan yang sudah dipesan, kemudian mengakses halaman *My Booking* dalam aplikasi *mobile*. Selanjutnya, karyawan memilih *booking* yang ingin dijadwalkan ulang, lalu mengisi informasi yang diperlukan seperti tanggal rapat baru, waktu mulai baru dan waktu selesai baru lalu mengklik tombol *Reschedule Now* yang terletak di bagian bawah halaman. Proses pengujian *reschedule booking* ruang rapat ini dapat dilihat pada Gambar 5.21 berikut.



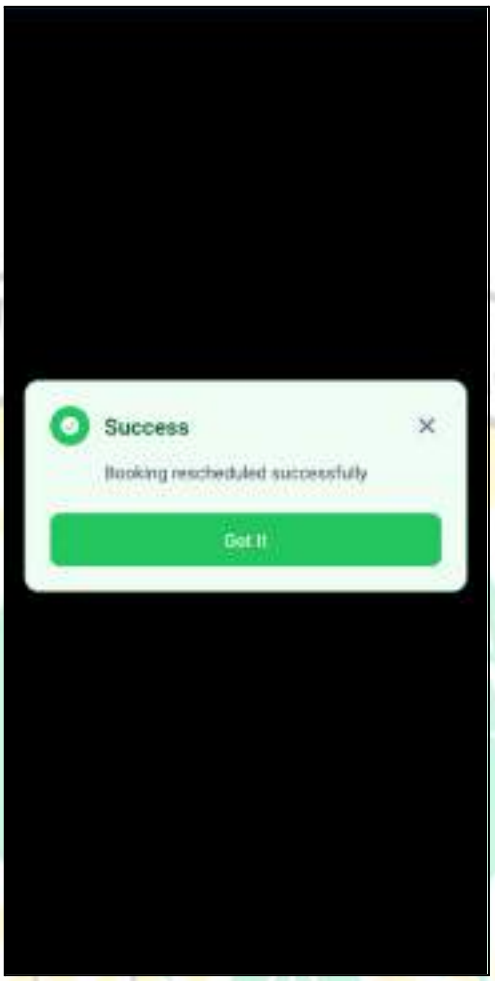
Gambar 5.21 Pengujian *Reschedule Booking* Ruang Rapat

Pada saat semua data yang diubah sudah sesuai, karyawan mengklik tombol "*Confirm Reschedule*". Sistem akan menampilkan popup "*Booking Reschedule Successfully!*" Data telah berhasil ditambahkan ke dalam *database*. Tampilan proses pengujian tambah data *reschedule booking* ruang rapat (benar) dapat dilihat pada Gambar 5.23 berikut.



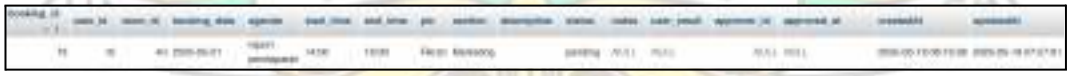
Gambar 5.23 *Output Reschedule Booking* Ruang Rapat

Sistem akan menampilkan halaman form reschedule *booking* ruang rapat. Kemudian karyawan mengganti data yang diperlukan yaitu tanggal, waktu mulai dan waktu selesai. Proses pengujian pengisian data dapat dilihat pada Gambar 5.22 berikut.



Gambar 5.22 Pengujian *Reschedule Booking* Ruang Rapat *(*Benar)

Data *Reschedule booking* ruang rapat yang ditambahkan telah berhasil tersimpan ke dalam database dan ditampilkan di halaman *my booking* dengan data yang telah ditambahkan itu. Tampilan data *reschedule booking* ruang rapat yang baru saja ditambahkan ke halaman *my booking* dapat dilihat pada Gambar 5.23, serta bukti data berhasil disimpan pada database di tabel *roombookings*, dapat dilihat pada Gambar 5.24 berikut.



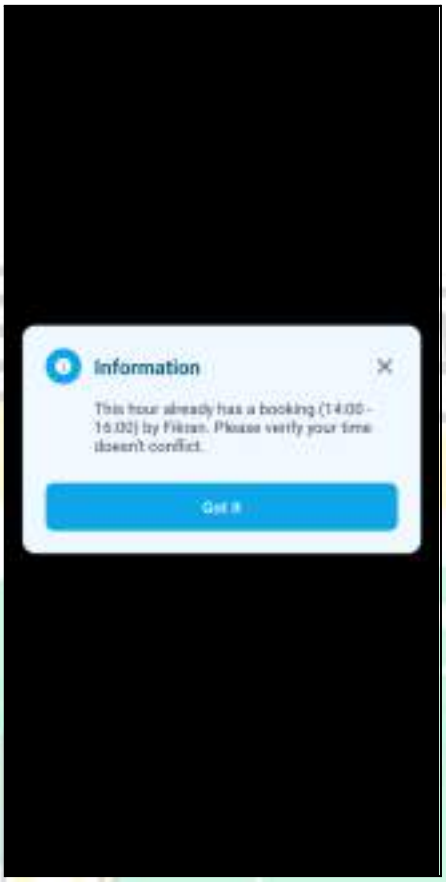
Gambar 5.24 Data *Reschedule Booking* Ruang Rapat Tersimpan di *Database*

Sedangkan untuk hasil pengujian alternatif terhadap fungsional tambah data *booking* ruang rapat dapat dilihat pada Tabel 5.7 berikut.

Tabel 5.7 Hasil Pengujian *Reschedule Booking* Ruang Rapat (Alternatif)

Kasus Hasil Pengujian (Alternatif)	
Data masukan	Tidak mengisi data yang wajib diisi.
Hal yang diharapkan	Sistem menampilkan pesan kesalahan pada *form.*
Pengamatan	Sistem menampilkan pesan kesalahan pada *form.*
Hasil	Sesuai

Pengujian alternatif ini dilakukan dengan mengklik tombol "*Confirm Reschedule*" ketika jadwal yang *reschedule* oleh admin sudah ada yang *booking*, kemudian sistem akan menampilkan pesan kesalahan pada *form*. Hasil pengujian *reschedule booking* ruang rapat (alternatif) dapat dilihat pada Gambar 5.25 berikut.



Gambar 5.25 *Output* Pengujian *Reschedule Booking* Ruang Rapat *(Alternatif)*

5.2.3. *User Acceptance Test* (UAT)

User Acceptance Testing (UAT) adalah pengujian aplikasi terhadap pengguna yang dilakukan sehubungan dengan kebutuhan pengguna terakhir atau end user (Adima et al., 2021). Tujuan dari UAT adalah untuk memverifikasi bahwa semua fitur aplikasi berjalan sesuai harapan dan dapat memberikan manfaat serta mempermudah tugas pengguna (Wahyudi et al., 2023). Pengujian ini melibatkan dua kelompok pengguna internal, yaitu satu Admin dan banyak Karyawan. Umpan balik dari kedua kelompok ini diperoleh melalui kuesioner yang dibagikan, yang masing-masing difokuskan pada Admin dan Karyawan. Metode pengujian yang digunakan dalam penelitian ini adalah UAT menggunakan **skala Likert**. Rensis Likert telah mengembangkan skala pengukuran ini dan mempublikasikannya pada tahun 1932 (Suasapha, 2020). Skala ini menggunakan ukuran ordinal, yang memungkinkan responden untuk memberikan peringkat. Tanggapan untuk setiap item instrumen pada skala ini memiliki kriteria penilaian

dari positif hingga negatif , dengan kata-kata yang digunakan seperti: sangat baik, baik, cukup, kurang baik, dan tidak baik

Hasil UAT dapat dilihat pada **LAMPIRAN H.** Dari lembaran UAT tersebut dapat dilakukan perhitungan persentase kepuasan pengguna dengan menggunakan beberapa kriteria yang dapat dilihat pada Tabel 5.8, dan menggunakan Rumus yang dapat dilihat pada Gambar 5.26.

Tabel 5.8 Skala Likert (Suasapha, 2020)

Skala	Keterangan	Skor	Persentase
SS	Sangat Sesuai	5	84,01% - 100%
S	Sesuai	4	68,01% - 84,00%
CS	Cukup Sesuai	3	52,01% - 68,00%
TS	Tidak Sesuai	2	36,01% - 52,00%
STS	Sangat Tidak Sesuai	1	20,00% - 36,00%

Gambar 5.26 Rumus Pengukuran (Sari, 2016)

Keterangan:

1. Skor actual merupakan pilihan dari semua responden dari kuesioner yang telah diberikan.
2. Skor ideal diasumsikan bahwa semua responden memilih skor tertinggi dari semua jawaban.
3. S = Jumlah frekuensi dikali dengan skor jawaban

Hasil penelitian lembaran UAT yang diberikan oleh masing-masing aktor yaitu sebagai berikut :

1. Penilaian Admin, pada lembaran pengujian aktor ini terdapat 17 kriteria yang dinilai, dari penilaian tersebut didapatkan 13 kriteria dengan kategori

penilaian sangat sesuai atau sangat setuju, 3 kriteria dengan kategori penilaian sesuai atau setuju, dan 1 kriteria dengan kategori penilaian cukup sesuai atau cukup setuju. Penguji juga memberikan hasil akhir penilaian kepuasan secara keseluruhan yaitu **Saya Cukup Puas dengan Sistem yang Dibangun.** Sehingga, didapatkan hasil perhitungan sebagai berikut.

S = (13 × 5) + (3 × 4) + (1 × 3) = 80
Skor Ideal= 17 × 5 = 85
Hasil Persentase = (80 ÷ 85) × 100% = 94.12%

2. Penilaian Karyawan 1, pada lembaran pengujian aktor ini terdapat 18 kriteria yang dinilai, dari penilaian tersebut didapatkan 10 kriteria dengan kategori penilaian sangat sesuai atau sangat setuju, 6 kriteria dengan kategori penilaian sesuai atau setuju, dan 2 kriteria dengan kategori penilaian cukup sesuai atau cukup setuju. Penguji juga memberikan hasil akhir penilaian kepuasan secara keseluruhan yaitu **Saya Sangat Puas dengan Sistem yang Dibangun.** Sehingga, didapatkan hasil perhitungan sebagai berikut.

S = (10 × 5) + (6 × 4) + (2 × 3) = 80
Skor Ideal= 18 × 5 = 85
Hasil Persentase = (80 ÷ 90) × 100% = 88.89%

3. Penilaian Karyawan 2, pada lembaran pengujian aktor ini terdapat 18 kriteria yang dinilai, dari penilaian tersebut didapatkan 14 kriteria dengan kategori penilaian sangat sesuai atau sangat setuju, 2 kriteria dengan kategori penilaian sesuai atau setuju, dan 2 kriteria dengan kategori penilaian cukup sesuai atau cukup setuju. Penguji juga memberikan hasil akhir penilaian kepuasan secara keseluruhan yaitu **Saya Sangat Puas dengan Sistem yang Dibangun.** Sehingga, didapatkan hasil perhitungan sebagai berikut.

S = (14 × 5) + (2 × 4) + (2 × 3) = 84

Skor Ideal= 18 × 5 = 85

Hasil Persentase = (84 ÷ 90) × 100% = 93.33%

Berdasarkan hasil perhitungan persentase UAT atau kepuasan pengguna tersebut didapatkan persentase rata-rata secara keseluruhan yaitu 92.11%, yang dapat diartikan berdasarkan tabel kriteria yang digunakan, persentase tersebut termasuk ke dalam penilaian "sangat sesuai". Sehingga, sistem yang dibangun tersebut **sangat sesuai dengan kebutuhan pengguna** dalam melakukan booking meeting room dan transportasi menggunakan perangkat mobile.

5.3. Analisis Hasil Pengujian

Berdasarkan hasil pengujian yang telah dilakukan sebelumnya, dengan fokus pada ketersediaan dan kesesuaian fungsional sistem yang diuji secara manual, diperoleh hasil yang sesuai antara desain sistem dan output yang diharapkan. Tidak ditemukan kegagalan atau kesalahan pada setiap fungsional yang diuji. Berdasarkan hasil tersebut, dapat disimpulkan bahwa Sistem Informasi *Booking* Ruang Rapat dan Transportasi Berbasis *Mobile* pada PT SISI telah berjalan sesuai dengan fungsional yang dirancang. Hasil pengujian sistem dapat dilihat pada Tabel 5.8 berikut.

Tabel 5.9 Hasil Pengujian

No	Item Uji	User	Detail Pengujian	Hasil
1	Login *back office*	Admin	Login	Sesuai
2	Logout *back office*	Admin	Logout	Sesuai
3	Dashboard	Admin	View	Sesuai
4	Manage Room	Admin	View, Add, Delete, dan Edit	Sesuai
5	Manage Transportasi	Admin	View, Add, Delete, dan Edit	Sesuai
6	Manage User	Admin	View, Add, Delete, dan Edit	Sesuai
7	Booking	Admin	View, Delete dan Export	Sesuai
8	Login aplikasi *mobile*	Karyawan	Login	Sesuai
9	Logout aplikasi *mobile*	Karyawan	Logout	Sesuai
10	Home	Karyawan	View	Sesuai
11	Booking Room	Karyawan	Add, View, Reschedule, dan Cancel	Sesuai
12	Booking Transportasi	Karyawan	Add, View, Reschedule, dan Cancel	Sesuai
13	Room	Karyawan	View	Sesuai
14	Transportasi	Karyawan	View	Sesuai

5.4. Analisa dan Pembahasan Hasil Pengujian

Pengujian sistem *booking* ruang rapat dan transportasi di PT SISI dilakukan menggunakan metode *Black Box Testing*, yaitu pengujian fungsi utama tanpa melihat kode program. Pengujian ini melibatkan dua jenis pengguna, Admin dan Karyawan, dengan total 8 pengujian untuk Admin dan 7 untuk Karyawan. Hasil pengujian menunjukkan seluruh fitur bekerja sesuai rancangan dan memenuhi kebutuhan pengguna.

Pembuatan aplikasi *mobile* menggunakan React Native dilakukan dengan satu program yang bisa digunakan untuk kedua sistem operasi, yaitu Android dan iOS. Cara ini menghilangkan kebutuhan membuat dua program yang berbeda untuk masing-masing sistem. Selain itu, React Native menggunakan elemen-elemen yang sudah ada di dalam sistem operasi Android dan iOS, seperti tombol dan menu, untuk membangun tampilan aplikasi. Dengan cara ini, aplikasi dapat bekerja seperti aplikasi asli pada masing-masing perangkat, memastikan bahwa pengguna dapat dengan mudah mengakses dan menggunakan fitur *booking* tanpa hambatan.

Sistem backend dibangun menggunakan Express.js dan *database* MySQL yang terintegrasi dengan aplikasi *mobile* dan web admin. Pemberitahuan yang dikirim melalui email berfungsi dengan baik, memberikan informasi penting seperti konfirmasi *booking*, pengingat jadwal, dan pembatalan secara langsung. Fitur ini membantu mengurangi risiko bentrok jadwal dan meningkatkan koordinasi antara admin dan karyawan.

Sistem baru ini memiliki beberapa keunggulan jika dibandingkan dengan sistem lama, yang menggunakan pemesanan lewat website, antara lain:

1. Pemberitahuan melalui email dan pengingat otomatis yang menonjol berbeda dengan penelitian sebelumnya, seperti yang dilakukan oleh Ani Rachmaniar et al. (2024) dan Frans Ikorasaki et al. (2024), yang mungkin hanya fokus pada konfirmasi seketika, aplikasi ini secara unik menerapkan pemberitahuan dan pengingat otomatis melalui email. Fitur ini memberi pembaruan status pemesanan secara langsung kepada pengguna.
2. Fungsionalitas *reschedule* dan pembatalan penuh aplikasi ini secara istimewa memiliki fungsionalitas untuk menjadwalkan ulang dan

membatalkan pemesanan. Pengguna dapat mengubah jadwal atau membatalkan pemesanan langsung melalui aplikasi. Ini merupakan respons langsung terhadap kebutuhan yang teridentifikasi dari wawancara dengan admin sistem *booking*.

3. Pengembangan dengan Teknologi Terbaru yang Terintegrasi Aplikasi ini dibuat menggunakan React Native untuk perangkat mobile, Next.js untuk web admin, Express.js sebagai *backend*, dan MySQL sebagai *database*. React Native memungkinkan pembuatan aplikasi mobile yang responsif untuk Android dan iOS dari satu basis kode. Hal ini merupakan keunggulan yang membedakan dibandingkan dengan penelitian sebelumnya seperti "Aplikasi Pemesanan Ruangan Panti Jompo Yayasan Karya Asih Berbasis Android" atau "Aplikasi Pemesanan Tiket Bioskop di Kota Medan Berbasis Android" yang terbatas pada platform Android saja

Beberapa manfaat yang diperoleh pengguna dari sistem ini antara lain:

1. Admin dapat mengelola data ruang dan transportasi secara terstruktur melalui web admin.
2. Karyawan dapat melakukan booking, penjadwalan ulang, dan pembatalan melalui aplikasi mobile dengan proses yang lancar.
3. Informasi yang dikirim secara real-time memastikan pengguna mendapatkan update terbaru sehingga menghindari kesalahan jadwal.
4. Fitur ekspor data ke Excel memudahkan pelaporan dan evaluasi penggunaan fasilitas.

Secara keseluruhan, hasil pengujian membuktikan bahwa aplikasi ini mampu memenuhi kebutuhan PT SISI dengan menyediakan proses booking yang didukung teknologi yang tepat. Pemilihan React Native berperan penting dalam menghasilkan aplikasi mobile yang responsif dan dapat menjangkau berbagai perangkat pengguna.

BAB VI
PENUTUP

Bagian ini menyajikan kesimpulan dan saran dalam laporan tugas akhir ini. Kesimpulan memberikan ringkasan dari hasil penelitian yang diperoleh, sesuai dengan tujuan yang telah ditetapkan sejak awal. Saran disampaikan sebagai harapan-harapan yang dapat menjadi acuan untuk pengembangan penelitian lebih lanjut di masa mendatang.

6.1. Kesimpulan

Berdasarkan penelitian yang telah dilakukan tentang Pembangunan Sistem Informasi *Booking* Ruang Rapat dan Transportasi Berbasis *Mobile* pada PT SISI (Sinergi Informatika Semen Indonesia), dapat disimpulkan bahwa:

1. Sistem aplikasi telah berhasil dibangun sesuai dengan kebutuhan pengguna, menggunakan React Native untuk aplikasi *mobile*, Next.js untuk *website* admin, Express.js untuk pengembangan API, dan MySQL sebagai penyimpanan data.
2. Sistem ini menyelesaikan masalah utama yang dihadapi PT SISI yaitu sulitnya pemesanan langsung, tidak adanya pemberitahuan, dan aplikasi yang lambat di perangkat *mobile*. Dengan sistem baru ini, proses pemesanan menjadi lebih teratur dengan berkurangnya bentrok jadwal serta meningkatnya ketepatan dalam penggunaan fasilitas..
3. Fitur-fitur yang dibuat telah memenuhi kebutuhan pengguna, termasuk pemesanan ruang rapat dan transportasi, mengubah jadwal, membatalkan pesanan, pemberitahuan langsung, pengingat, persetujuan pemesanan, pengelolaan fasilitas, dan ekspor data pemesanan ke excel. Semua fitur ini telah diuji dan berhasil bekerja sesuai dengan kebutuhan yang sudah ditentukan.
4. Berdasarkan hasil pengujian *User Acceptance Testing* (UAT) yang mencapai 92,11%, berdasarkan skala linkert yang digunakan dapat disimpulkan bahwa fitur aplikasi telah sangat sesuai dengan kebutuhan pengguna.

6.2. Saran

Berdasarkan hasil penelitian yang telah dilakukan, beberapa saran untuk pengembangan sistem di masa mendatang adalah mengoptimalkan dukungan aplikasi pada perangkat iOS agar dapat digunakan secara maksimal di berbagai jenis perangkat mobile. Hal ini bertujuan untuk meningkatkan kenyamanan dan fleksibilitas pengguna dalam mengakses sistem. Selain itu, disarankan untuk melaksanakan pelatihan berkala bagi pengguna dan administrator sistem guna memastikan pemahaman yang baik terhadap seluruh fitur yang tersedia, sehingga sistem dapat dimanfaatkan secara optimal dalam mendukung kelancaran operasional perusahaan.

DAFTAR PUSTAKA

Agustini, A., & Kurniawan, W. J. (2020). “Sistem E-Learning Do’a dan Iqro dalam Peningkatan Proses Pembelajaran pada TK Amal Ikhlas.” *Jurnal Mahasiswa Aplikasi Teknologi Komputer Dan Informasi (JMApTeKsi)*, 1(3), 154–159.

Aji, T. B. A., Ajie, H., & Nugraheni, M. (2022). Pengembangan Web Service Aplikasi Manajemen Aset UPT TIK Universitas Negeri Jakarta. JURNAL PINTER, 6(2), 69-75.

Ani Rachmaniar, Desy Diana, Mohamad Saefudin. (2024). *Aplikasi Mobile Reservasi Ruang Meeting dan Event Space Pada Marquee Executive Offices*. JISICOM (Journal of Information System, Informatics and Computing), Vol. 8, No. 1, Hal. 14-28.

Arifina, M. N., & Siahaan, D. (2020). "Structural and Semantic Similarity Measurement of UML Use Case Diagram." *Lontar Komputer, 11*(2). https://doi.org/10.24843/LKJITI.2020.v11.i02.p03

Bachtiar, D. H., Paniran, P., & Suksmadana, I. M. B. (2024). Perancangan Back-end Api pada Aplikasi Mobile Fruityfit Menggunakan Framework Express JS. *Mars: Jurnal Teknik Mesin, Industri, Elektro Dan Ilmu Komputer*, 2(3), 107-117. https://doi.org/10.61132/mars.v2i3.138

Bagwan, K., & Ghule, S. (2019). “Mobile Application Development Using React Native: A Framework to Bridge the Gap between Cross-Platform and Native Applications.” International Journal of Computer Applications, 182(1), 23–29.

Damayanti, D., Sulistiani, H., & Umpu, E. F. G. S. (2021). Analisis dan perancangan sistem informasi akuntansi pengelolaan tabungan siswa pada SD Ar-Raudah Bandarlampung. *Jurnal Teknologi dan Informasi*, 11(1), 40–50. https://doi.org/10.34010/abdujati.v11i1.3392

Fataha, M. K.-165410221. (2022). “APLIKASI PENYEWAAN PERLENGKAPAN PEMETAAN DATA TERPADU KESEJAHTERAAN SOSIAL DI KABUPATEN GARUT.” https://doi.org/10.25126/jtiik.202296098

Fintri Indriyani, Yunita, Muthia, D. A., Surniandari, A., & Sriyadi. (2019). Analisa Perancangan Sistem Informasi. Bina Sarana Informatika Jakarta.

Frans Ikorasaki, Rida Utami, Cindy Paramitha Lubis, Nur Anzelina Hrp, Bintang Wildan Utama, Nandri Marsan Sitinjak. (2024). *Aplikasi Pemesanan Ruangan Panti Jompo Yayasan Karya Asih Berbasis Android*. Jurnal Widya, Vol. 5, No. 1, Hal. 482-493.

Irfan, M., Siregar, H., & Handoko, J. T. (2023). Pengembangan Dan Integrasi Aplikasi Prediksi Jumlah Gagal Produksi PC Meggunakan Metode Triple Exponential Smoothing Pada Sistem Aplikasi Produksi Di PT Tera Data Indonusa,Tbk. Prosiding Seminar Nasional Darmajaya, 1(November 2015), 80–96

Irfansyah, A. (6 Januari 2023). 5 Kelebihan Framework Laravel untuk Pengembangan Web. *Eduparx Blog*. Diakses pada 23 Oktober 2024 dari https://www.eduparx.com/blog/kelebihan-framework-laravel

Ispandi, M., Fahmi, M., Sudarsono, B., Darono, H. E., & Jefi. (2024). Perancangan Sistem Manajemen Pembelajaran dan Forum Diskusi Berbasis Website Menggunakan Framework Next JS di MTs Nurul Fiqri. JATI (Jurnal Mahasiswa Teknik Informatika), 8(3), 3757–3765.

Karthikeyan, R., & Karthick, S. (2023). "Hotel Booking Mobile and Web Application." International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET), 6(6), 1823–1828. http://www.ijmrset.com/

Kausar Bagwan, M. I., & Swati Ghule, P. D. (2019). A Modern Review on LaravelPHP Framework. *IRE Journals*, 2(12).

Komaran, R. M., Andarsyah, R., Lubis, M., & Awangga, R. M. (2023). "Langkah Mudah Membuat Dashboard Rest API." Penerbit Buku Pedia.

Marianko, G., Seferian, A., Leerentveld, T., Jackson, D., & Hanson, A. (2020). "Real-Time Automatic Meeting Room Reservation Based on the Number of Actual Participants." *United States Patent*, US 10,692,020 B2, June 23, 2020.

Martin, H. (2023). Use case adalah dan beberapa jenisnya. ITBox. Diakses pada 18 Mei 2025 dari https://itbox.id/uncategory/use-case-adalah-dan-beberapa-jenisnya/

Maulana, C. A., Riza, Y. S., & Asrin, F. (2023). Aplikasi Berbasis Web untuk Manajemen Ruangan, Presensi, dan Notulensi Rapat Pada Bappeda Kota Pontianak. Jurnal Ilmiah ILKOMINFO – Jurnal Ilmu Komputer dan Informatika, 6(2), 191–201.

Novian, C., Idah, Y. M., & Rifai, Z. (2022). *Pemodelan Proses Bisnis Pengadaan Barang (Stok) Menggunakan Pendekatan Business Process Modelling Notation (BPMN)*. Journal of Information System Management (JOISM), 3(2), 63-69.

Nugraha, S., Prasetijo, A. B., & Eridani, D. (2022). Perancangan Back-End Aplikasi Reservasi Talanoa Kopi and Space Menggunakan Framework Express.js. *Jurnal Teknik Komputer*, 1(3), 126-131. https://doi.org/10.14710/jtk.v1i3.36901

Nurhidayah, L., Salsabillah, A., & Yanti, F. R. (2023). Perancangan Aplikasi Pemesanan Tiket Bioskop di Kota Medan Berbasis Android. JUKTISI (Jurnal Komputer Teknologi Informasi Sistem Komputer), 2(2), 305-314.

Pricillia, T., & Zulfachmi. (2021). *Survey Paper: Perbandingan Metode Pengembangan Perangkat Lunak (Waterfall, Prototype, RAD)*. Bangkit Indonesia, Vol. X, No. 01, Bulan Maret 2021, 6-12.

Rahma, F. A., & Samsudin. (2024). *Android-Based Sports Infrastructure E-Booking Application at Provincial Youth and Sports Office Using Waterfall Method*. Journal of Computer Networks, Architecture and High Performance Computing, 6(4). https://doi.org/10.47709/cnahpc.v6i4.4804

Ramadhan, R. F., & Mukhaiyar, R. (2020). Penggunaan Database Mysql dengan Interface PhpMyAdmin sebagai Pengontrolan Smarthome Berbasis Raspberry Pi. JTEIN: Jurnal Teknik Elektro Indonesia, 1(2), 129-134.

Rasefta, R. S., & Esabella, S. (2020). Sistem Informasi Akademik SMK Negeri 3 Sumbawa Besar Berbasis Web. Jurnal JINTEKS, 2(1), 50–58. ISSN: 2686-3359.

Sasmito, G. W., Wiyono, S., Irwansyah, E., & Suhartono, D. (2022). Transcrop: Media Pemesanan Transportasi Agribisnis Online Berbasis Web. Jurnal Informatika: Jurnal Pengembangan IT (JPIT), 7(1), 8-12.

Setiawan, R. (15 Desember 2021). Apa itu Framework? Developer Wajib Tahu. *Dicoding*. Diakses pada 22 September 2024 dari https://www.dicoding.com/blog/apa-itu-framework/

Sinambela, M. R., Waidah, D. F., Susilo, T., Jaya, N. A., & Friansyah, I. G. (2024). Rancang Bangun Perpustakaan Digital Berbasis Website pada SD Swasta 001 PT. KG Meral Barat di Kabupaten Karimun. Tikar: Jurnal Teknik Informatika Karimun, 5(1), 12–23. https://doi.org/https://doi.org/10.51742/teknik_informatika.v5i1.1231

Sommerville. (2011). *Software Engineering* (9th ed.). Wesley: Addison.

Subhiyakto, E. R., & Astuti, Y. P. (2020). Aplikasi Pembelajaran Class Diagram Berbasis Web untuk Pendidikan Rekayasa Perangkat Lunak. *Simetris: Jurnal Teknik Mesin, Elektro dan Ilmu Komputer*, 11(1), 143–150. https://doi.org/10.24176/simet.v11i1.3787

Sulaeman, H., & Waluyo, A. F. (2023). "Perancangan Aplikasi Manajemen Keuangan Berbasis Mobile Menggunakan React Native Untuk Meningkatkan Literasi Keuangan Individu." KLIK: Kajian Ilmiah Informatika dan Komputer, 4(2), 1021–1031. https://djournals.com/klik

Supit, M. A., Pratasik, S., Kainde, Q. C., & Kumajas, S. (2021). *Pemodelan Proses Bisnis dengan Business Process Management Notation pada Fakultas Teknik Universitas Negeri Manado*. EduTIK: Jurnal Pendidikan Teknologi Informasi dan Komunikasi, 1(6), 630-639.

Telkom University. (2024). "Apa Itu Pengembangan Perangkat Lunak?" docif.telkomuniversity.ac.id. https://docif.telkomuniversity.ac.id/apa-itu-pengembangan-perangkat-lunak/

Wahyudi, A., Suryani, N., & Wibowo, B. (2023). *Pengujian Sistem Pengelolaan Informasi Pengguna di Lingkungan Organisasi*. Jurnal Teknologi dan Sistem Informasi, 12(3), 123-130.

Wahyudi, J., Asbari, M., Sasono, I., Pramono, T., & Novitasari, D. (2022). Database Management in MYSQL. Jurnal Edumaspul, 6(2), 2413-2417.

Wayahdi, M. R., & Ruziq, F. (2023). "Pemodelan Sistem Penerimaan Anggota Baru dengan Unified Modeling Language (UML)." *Jurnal Minfo Polgan, 12*(1). https://doi.org/10.33395/jmp.v12i1.12870

Zaihan, M. A. S., & Yunus, M. A. M. (2022). *Smartphone Technician Service Booking Application (Technician2U)*. Applied Information Technology And Computer Science, 3(1), 660-670. https://doi.org/10.30880/aitcs.2022.03.01.044

Adima, N., Praptono, B., & Sagita, B. H. (2021). Pengembangan program after sales service PT Zatalini Cipta Persada menggunakan aplikasi berbasis web dalam proyek kerjasama dengan PT Pertamina Pemasaran. *e-Proceeding of Engineering*, *8*(2), 2148–2158.

Suasapha, A. H. (2020). Skala Likert untuk penelitian pariwisata; beberapa catatan untuk menyusunnya dengan baik. *Jurnal Kepariwisataan*, *19*(1), 26–37. https://doi.org/10.52352/jpar.v19i1.407

Chamida, M. A., Susanto, A., & Latubessy, A. (2021). Analisa User Acceptance Testing Terhadap Sistem Informasi Pengelolaan Bedah Rumah di Dinas Perumahan Rakyat dan Kawasan Permukiman Kabupaten Jepara**.** *Indonesian Journal of Technology, Informatics and Science (IJTIS)*, Vol. 3, No. 1, hlm. 36–41. DOI: 10.24176/ijtis.v3i1.7531

Sari, T. N. (2016). *Analisis kualitas dan pengembangan sistem informasi akademik berbasis web menggunakan standard ISO 9126*. Jurnal Informatika dan Komputer (JIKO), 1(1), 1–7. Universitas Gadjah Mada.

PT Sinergi Informatika Semen Indonesia. (2024, Juli 17). *Achmad Tholchah: Kepemimpinan berlandaskan kejujuran dan ketekunan.* Diakses dari https://sisi.id/stories/news/achmad-tholchah-kepemimpinan-berlandaskan-kejujuran-dan-ketekunan/

UNIVERSITAS ANDALAS
UNTUK
KEDJAJAAN
BANGSA

LAMPIRAN A

***USE CASE SCENARIO* (LANJUTAN)**

1. Admin dapat *login* website *back office*.

Use Case :	*Login Back Office*
Actor :	Admin
Preconditions :	1. Aktor memiliki akun yang terdaftar pada website *back office*. 2. Aktor telah membuka halaman *login*.
Flow of Event :	1. Aktor menginputkan *email* dan *password*. 2. Aktor memilih tombol "*Login*". 3. Sistem melakukan validasi input *email* dan *password*. 4. Sistem mengautentikasi aktor dan mengarahkan ke halaman *dashboard*.
Alternate Flows :	A1. Aktor menginputkan *email* atau *password* yang salah. a. Sistem menampilkan pesan *error*: "*Email* atau *password* Anda salah." b. Sistem menampilkan kembali *form login* dengan *input field email* yang diisi sebelumnya. c. Aktor memperbaiki *email* atau *password*. d. Kembali ke langkah 2.
Postconditions :	Aktor berhasil *login* dan sistem akan diarahkan ke halaman *dashboard.*

2. Admin Dapat *Logout* Dari Website *Back Office*.

Use Case :	*Logout*
Actor :	Admin
Preconditions :	Aktor telah *login* ke dalam aplikasi
Flow of Event :	1. Aktor klik menu profil di sidebar dan memilih tombol *"Logout"* 2. Sistem mengarahkan Aktor ke halaman *login*
Postconditions :	Aktor berhasil *logout*

3. Admin Dapat Melihat *Dashboard*.

Use Case :	View *Dashboard*
Actor :	Admin
Preconditions :	Aktor telah *login* kedalam website *back office*
Flow of Event :	1. Aktor klik menu "Dashboard" di sidebar 2. Sistem mengarahkan Aktor ke halaman *Dashboard*
Postconditions :	Aktor berhasil melihat dashboard

4. Admin Dapat Melihat Data Ruangan

Use Case :	View List Room
Actor :	Admin
Preconditions :	Aktor telah *login* kedalam website *back office*
Flow of Event :	1. Aktor klik *dropdown manage* di *sidebar* dan memilih menu *"Room"* 2. Sistem menampilkan *list room*
Postconditions :	Aktor berhasil melihat *list room*

5. Admin Dapat Menambah Data Ruangan

Use Case :	Add Room Data
Actor :	Admin
Preconditions :	1. Aktor telah *login* ke dalam website *back office* 2. Aktor berada di halaman *manage room*
Flow of Event :	1. Aktor klik tombol "Add room" 2. Sistem menampilkan modal *form input* untuk menambah *room* baru 3. Aktor mengisi *form* dengan data yang diperlukan 4. Aktor memilih tombol "Simpan" 5. Sistem melakukan validasi terhadap *input* 6. Jika semua validasi berhasil, sistem menyimpan

	data *room* baru ke dalam *database* 7. Sistem menampilkan pemberitahuan "*Room added successfully"*
Alternate Flows	A1. Jika aktor tidak mengisi semua *field* yang diperlukan, sistem menampilkan pesan *error*
Postconditions :	Aktor berhasil menambah data *room*

6. Admin Dapat Menghapus Data Ruangan

Use Case :	Remove Room Data
Actor :	Admin
Preconditions :	1. Aktor telah *login* ke dalam aplikasi 2. Aktor berada di halaman *manage room*
Flow of Event :	1. Aktor klik icon "*Trash*" 2. Sistem menampilkan alert "Apakah anda yakin? Data *room* akan dihapus secara permanen!" 3. Sistem menampilkan pemberitahuan "*Room deleted successfully"*
Postconditions :	Aktor berhasil menghapus data *room*

7. Admin Dapat Mengedit Data Ruangan

Use Case :	*Update Room Data*
Actor :	Admin
Preconditions :	1. Aktor telah *login* kedalam website *back office* 2. Aktor berada di halaman *manage room*
Flow of Event :	1. Aktor klik icon "edit" 2. Sistem menampilkan modal *form input* untuk mengedit *room* 3. Aktor mengedit *form* dengan data yang diperlukan 4. Aktor memilih tombol "Simpan" 5. Sistem melakukan validasi terhadap *input* 6. Jika semua validasi berhasil, sistem menyimpan

	data *room* ke dalam *database* 7. Sistem menampilkan pemberitahuan "*Room updated successfully"* 8. Sistem memindahkan aktor kembali ke halaman *manage room*
Postconditions :	Aktor berhasil mengedit data *room*

8. Admin dapat melihat data transportasi

Use Case :	View List Transport
Actor :	Admin
Preconditions :	Aktor telah *login* kedalam website *back office*
Flow of Event :	1. Aktor klik *dropdown manage* di *sidebar* dan memilih menu *"Transportasi"* 2. Sistem menampilkan *list* transportasi
Postconditions :	Aktor berhasil melihat *list* transportasi

9. Admin dapat menambah data transportasi

Use Case :	Add Transport Data
Actor :	Admin
Preconditions :	1. Aktor telah *login* ke dalam website *back office* 2. Aktor berada di halaman *manage* transportasi
Flow of Event :	1. Aktor klik tombol "*Add* transportasi" 2. Sistem menampilkan modal *form input* untuk menambah transportasi baru 3. Aktor mengisi *form* dengan data yang diperlukan 4. Aktor memilih tombol "Simpan" 5. Sistem melakukan validasi terhadap *input* 6. Jika semua validasi berhasil, sistem menyimpan data transportasi baru ke dalam *database*

	7. Sistem menampilkan pemberitahuan "*Transport data added successfully"*
Alternate Flows	A1. Jika aktor tidak mengisi semua *field* yang diperlukan, sistem menampilkan pesan *error*

10. Admin dapat menghapus data transportasi

Use Case :	Remove Transport Data
Actor :	Admin
Preconditions :	1. Aktor telah *login* ke dalam aplikasi 2. Aktor berada di halaman *manage* transportasi
Flow of Event :	1. Aktor klik icon "*Trash*" 2. Sistem menampilkan alert "Apakah anda yakin? Data transportasi akan dihapus secara permanen!" 3. Sistem menampilkan pemberitahuan "*Transport data deleted successfully"*
Postconditions :	Aktor berhasil menghapus data transportasi

11. Admin dapat mengedit data transportasi

Use Case :	*Update Transport Data*
Actor :	Admin
Preconditions :	1. Aktor telah *login* kedalam website *back office* 2. Aktor berada di halaman *manage* transportasi
Flow of Event :	1. Aktor klik icon "edit" 2. Sistem menampilkan modal *form input* untuk mengedit transportasi 3. Aktor mengedit *form* dengan data yang diperlukan 4. Aktor memilih tombol "Simpan" 5. Sistem melakukan validasi terhadap *input* 6. Jika semua validasi berhasil, sistem menyimpan data transportasi ke dalam *database*

	7. Sistem menampilkan pemberitahuan "*Transport data updated successfully"* 8. Sistem memindahkan aktor kembali ke halaman *manage* transportasi
Postconditions :	Aktor berhasil mengedit data transportasi

12. Admin dapat melihat list users

Use Case :	Melihat Data *User*
Actor :	Admin
Preconditions :	Aktor telah *login* ke dalam website *back office*
Flow of Event :	1. Aktor klik *dropdown manage* di *sidebar* dan memilih menu *"User"* 2. Sistem menampilkan *list user*
Postconditions :	Aktor berhasil melihat *list user*

13. Admin dapat menambah data user

Use Case :	Add *User*
Actor :	Admin
Preconditions :	1. Aktor telah *login* ke dalam website *back office* 2. Aktor berada di halaman *manage user*
Flow of Event :	1. Aktor klik tombol "*Add user*" 2. Sistem menampilkan modal *form input* untuk menambah *user* baru 3. Aktor mengisi *form* dengan data yang diperlukan 4. Aktor memilih tombol "Simpan" 5. Sistem melakukan validasi terhadap *input* 6. Jika semua validasi berhasil, sistem menyimpan data *user* baru ke dalam *database* 7. Sistem menampilkan pemberitahuan "*User added*

	successfully"
Alternate Flows	A1. Jika aktor tidak mengisi semua *field* yang diperlukan, sistem menampilkan pesan *error*

14. Admin dapat menghapus data user

Use Case :	*Remove User*
Actor :	Admin
Preconditions :	1. Aktor telah *login* ke dalam aplikasi 2. Aktor berada di halaman *manage user*
Flow of Event :	1. Aktor klik icon "*Trash*" 2. Sistem menampilkan alert "Apakah anda yakin? Data *user* akan dihapus secara permanen!" 3. Sistem menampilkan pemberitahuan "*User deleted successfully"*
Postconditions :	Aktor berhasil menghapus data *user*

15. Admin dapat mengedit data user

Use Case :	*Update User*
Actor :	Admin
Preconditions :	1. Aktor telah *login* kedalam website *back office* 2. Aktor berada di halaman *manage* transportasi
Flow of Event :	1. Aktor klik icon "edit" 2. Sistem menampilkan modal *form input* untuk mengedit *user* 3. Aktor mengedit *form* dengan data yang diperlukan 4. Aktor memilih tombol "Simpan" 5. Sistem melakukan validasi terhadap *input* 6. Jika semua validasi berhasil, sistem menyimpan data *user* ke dalam *database* 7. Sistem menampilkan pemberitahuan "*User updated successfully"*

	8. Sistem memindahkan aktor kembali ke halaman *manage user*
Postconditions :	Aktor berhasil mengedit data transportasi

16. Admin dapat melihat detail *booking*

Use Case :	Melihat Data Detail *Booking*
Actor :	Admin
Preconditions :	1. Aktor telah *login* ke dalam aplikasi 2. Aktor berada di halaman *list booking*
Flow of Event :	1. Aktor klik *"eye"* di salah satu data *booking* 2. Sistem memindahkan aktor ke halaman detail *booking*
Postconditions :	Aktor berhasil melihat detail *booking*

17. Admin dapat menghapus data *booking*

Use Case :	*Remove Bookings Data*
Actor :	Admin
Preconditions :	1. Aktor telah *login* ke dalam aplikasi 2. Aktor berada di halaman *list booking*
Flow of Event :	1. Aktor klik icon "*Trash*" 2. Sistem menampilkan alert "Apakah anda yakin? Data *booking* akan dihapus secara permanen!" 3. Sistem menampilkan pemberitahuan "*booking deleted successfully"*
Postconditions :	Aktor berhasil menghapus data *booking*

18. Admin dapat mengekspor data *booking*

Use Case :	*Export Data Booking*
Actor :	Admin

Preconditions :	1. Aktor telah *login* ke dalam aplikasi 2. Aktor berada di halaman *list booking*
Flow of Event :	1. Aktor klik tombol “ekspor” 2. Sistem mengekspor dan mengunduh otomatis file
Postconditions :	Aktor berhasil mengekspor data *booking*

19. Karyawan dapat *login* aplikasi *mobile*

Use Case :	*Login*
Actor :	Karyawan
Preconditions :	1. Aktor memiliki akun yang terdaftar pada aplikasi *mobile* 2. Aktor telah membuka halaman *login*.
Flow of Event :	3. Aktor menginputkan *email* dan *password*. 4. Aktor memilih tombol "*Login*". 5. Sistem melakukan validasi input *email* dan *password*. 6. Sistem mengautentikasi aktor dan mengarahkan ke halaman home.
Alternate Flows :	A1. Aktor menginputkan *email* atau *password* yang salah. a. Sistem menampilkan pesan *error*: "*Email* atau *password* Anda salah.” b. Sistem menampilkan kembali *form login* dengan *input field email* yang diisi sebelumnya. c. Aktor memperbaiki *email* atau *password*. d. Kembali ke langkah 2.
Postconditions :	Aktor berhasil *login* dan sistem akan diarahkan ke halaman home.

20. Karyawan Dapat *Logout* Dari Aplikasi *Mobile*

Use Case :	*Logout*

Actor :	Karyawan
Preconditions :	Aktor telah *login* ke dalam aplikasi
Flow of Event :	1. Aktor klik menu profil di navbar dan memilih tombol *"Logout"* 2. Sistem mengarahkan Aktor ke halaman *login*
Postconditions :	Aktor berhasil *logout*

21. Karyawan dapat melihat halaman *home*

Use Case :	View Homepage
Actor :	Karyawan
Preconditions :	Aktor telah *login* ke dalam aplikasi *mobile*
Flow of Event :	1. Aktor klik menu home di navbar 2. Sistem mengarahkan pengguna ke halaman home
Postconditions :	Aktor berhasil melihat halaman *home*

22. Karyawan dapat melihat *profile*

Use Case :	Melihat Halaman *Profile*
Actor :	Karyawan
Preconditions :	Aktor telah *login* ke dalam aplikasi *mobile*
Flow of Event :	1. Aktor klik menu profil di navbar 2. Sistem mengarahkan pengguna ke halaman *profile*
Postconditions :	Aktor berhasil melihat halaman *profile*

23. Karyawan dapat melakukan booking ruang rapat

Use Case :	*Booking Room*
Actor :	Karyawan

Preconditions :	1. Aktor telah *login* ke dalam aplikasi 2. Aktor berada di halaman *booking room*
Flow of Event :	1. Aktor klik opsi "*Room*" di halaman opsi *booking*. 2. Sistem menampilkan halaman *form input* untuk *booking room* baru. 3. Aktor mengisi *form* dengan data *booking* yang diperlukan 4. Aktor klik tombol "*Submit*" . 5. Sistem melakukan validasi terhadap *input*. 6. Jika semua validasi berhasil, sistem menyimpan data baru ke dalam *database*. 7. Sistem menampilkan pemberitahuan "*Booking submitted successfully*". 8. Sistem mengarahkan aktor ke halaman my *booking*
Alternate Flows :	A1. Jika aktor tidak mengisi semua *field* yang diperlukan, sistem menampilkan pesan error.

24. Karyawan dapat melakukan booking transportasi

Use Case :	*Booking Transport*
Actor :	Karyawan
Preconditions :	1. Aktor telah *login* ke dalam aplikasi 2. Aktor berada di halaman *booking* transportasi
Flow of Event :	3. Aktor klik opsi "*Transport*" di halaman opsi *booking*. 4. Sistem menampilkan halaman *form input* untuk *booking* transport baru. 5. Aktor mengisi *form* dengan data *booking* yang diperlukan 6. Aktor klik tombol "*Submit*" . 7. Sistem melakukan validasi terhadap *input*. 8. Jika semua validasi berhasil, sistem menyimpan data baru ke dalam *database*. 9. Sistem menampilkan pemberitahuan "*Booking submitted successfully*". 10. Sistem mengarahkan aktor ke halaman my

	booking
Alternate Flows :	A1. Jika aktor tidak mengisi semua *field* yang diperlukan, sistem menampilkan pesan error.

25. Karyawan dapat melihat *list my booking*

Use Case :	View Data Booking
Actor :	Karyawan
Preconditions :	Aktor telah *login* ke dalam aplikasi
Flow of Event :	1. Aktor klik menu my booking di navbar 2. Sistem mengarahkan pengguna ke halaman *my booking*
Postconditions :	Aktor berhasil melihat halaman *my booking*

26. Karyawan dapat melihat *detail booking*

Use Case :	Melihat *Detail Booking*
Actor :	Karyawan
Preconditions :	1. Aktor telah *login* ke dalam aplikasi 2. Aktor berada di halaman *my booking*
Flow of Event :	3. Aktor klik salah satu data di *list my booking* 4. Sistem mengarahkan pengguna ke halaman *detail booking*
Postconditions :	Aktor berhasil melihat *detail booking*

27. Karyawan dapat melakukan reschedule booking

Use Case :	*Reschedule Booking*
Actor :	Karyawan
Preconditions :	1. Aktor telah *login* ke dalam aplikasi 2. Aktor berada di halaman *my booking*
Flow of Event :	3. Aktor pilih data *booking* dengan status “Pending” 4. Sistem menampilkan halaman *detail booking*.

	5. Aktor klik tombol "*reschedule*" 6. Sistem akan menampilkan halaman *form reschedule* 7. Aktor memilih jadwal atau waktu yang ingin di *reschedule* 8. Aktor klik tombol "*Save*" 9. Sistem melakukan validasi terhadap *input*, jika semua validasi berhasil, sistem menyimpan data baru ke dalam *database*. 10. Sistem menampilkan pemberitahuan "*Booking updated successfully*". 11. Sistem mengarahkan aktor kembali ke halaman *my booking*
Alternate Flows :	e. A1. Jika aktor memilih tanggal dan waktu yang sudah lewat, sistem menampilkan pesan error
Postconditions :	Aktor berhasil melakukan r*eschedule booking*

28. Karyawan dapat melakukan cancel booking

Use Case :	*Cancel Booking*
Actor :	Karyawan
Preconditions :	1. Aktor telah *login* ke dalam aplikasi 2. Aktor berada di halaman *my booking*
Flow of Event :	3. Aktor pilih booking dengan status "*pending*" 4. Sistem menampilkan halaman *detail booking.* 5. Aktor memilih tombol "*Cancel*". 6. Sistem menampilkan pemberitahuan *"are u sure ?*". 7. Aktor klik tombol "*Yes*". 8. Sistem mengarahkan aktor kembali ke halaman *my booking.*
Postconditions :	Aktor berhasil melakukan *cancel booking*

29. Karyawan dapat melihat halaman *explore*

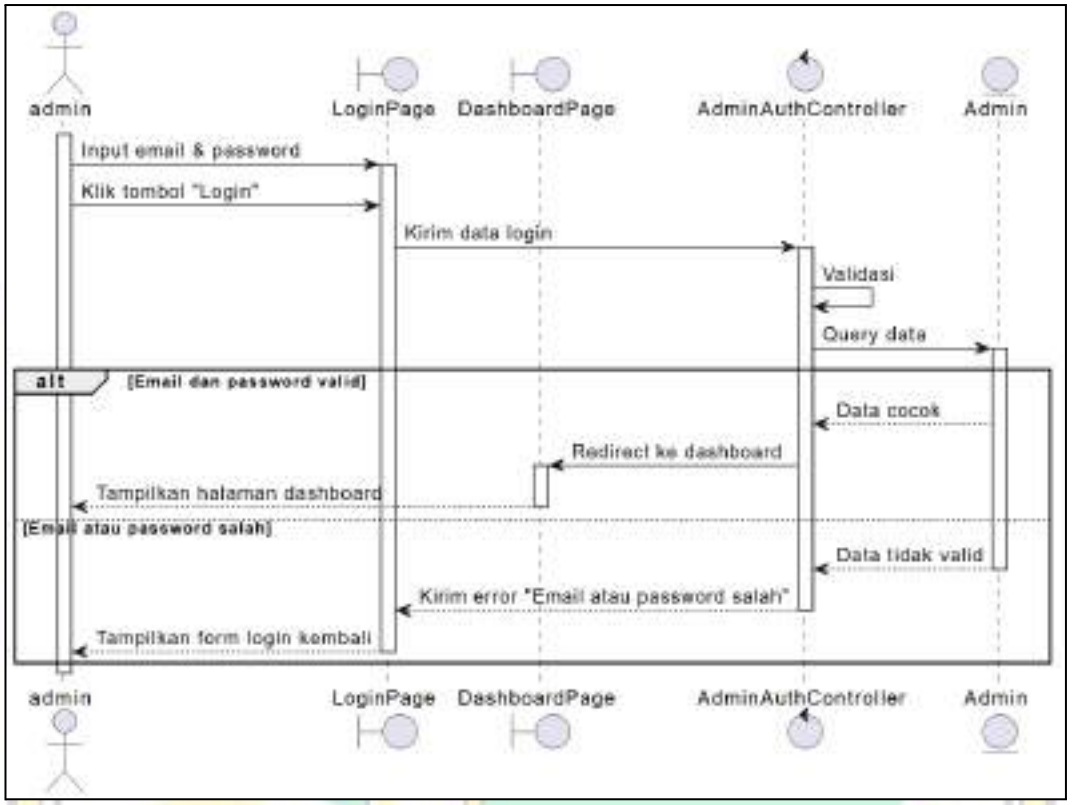
Use Case :	Melihat Halaman *Explore*
Actor :	Karyawan

Preconditions :	Aktor telah *login* ke dalam aplikasi *mobile*
Flow of Event :	1. Aktor klik menu *explore* di navbar 2. Sistem mengarahkan pengguna ke halaman *explore*
Postconditions :	Aktor berhasil melihat halaman *explore*

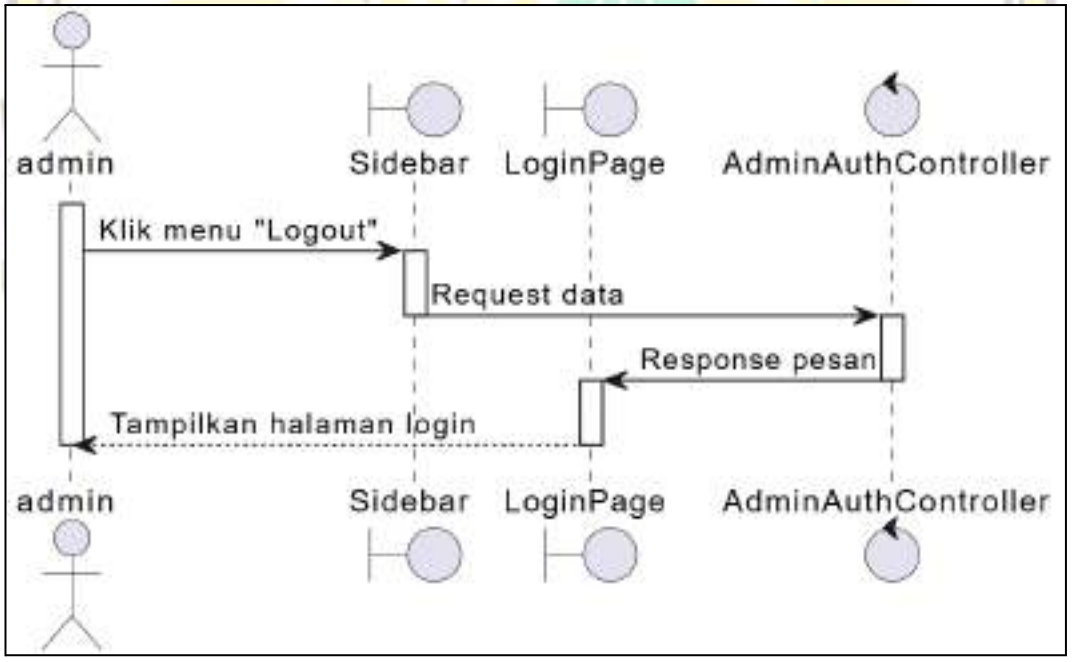
LAMPIRAN B

***SEQUENCE DIAGRAM* (LANJUTAN)**

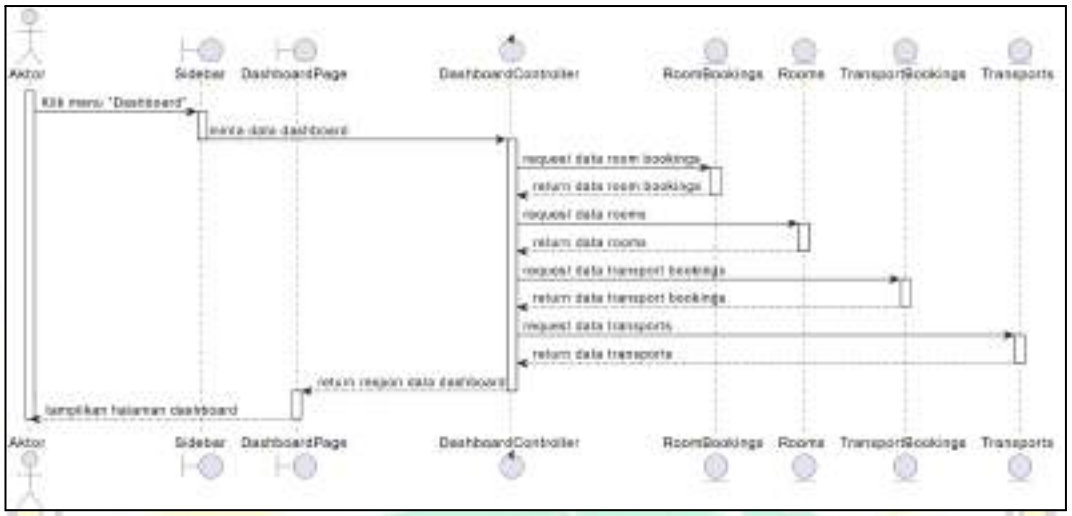
1. Admin dapat login *back office*



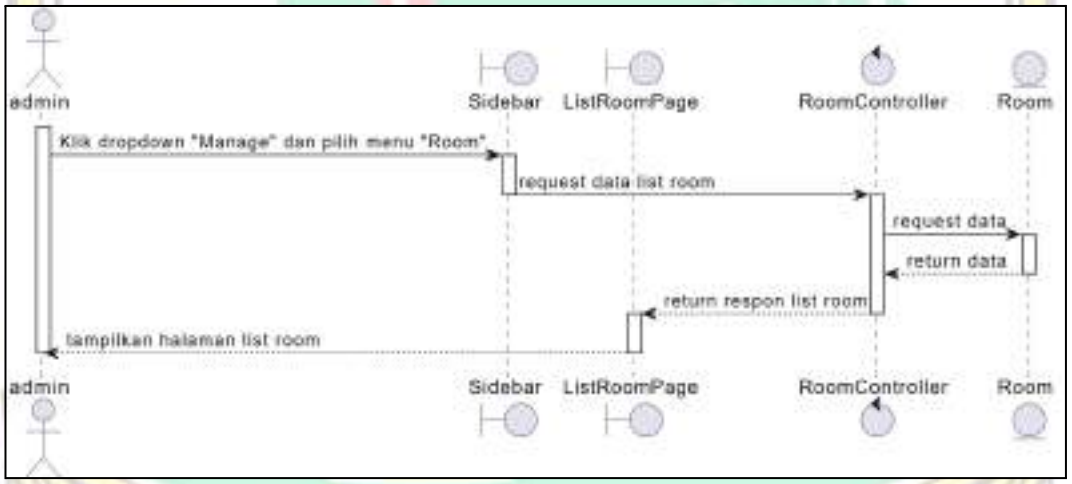
2. Admin dapat logout *back office*



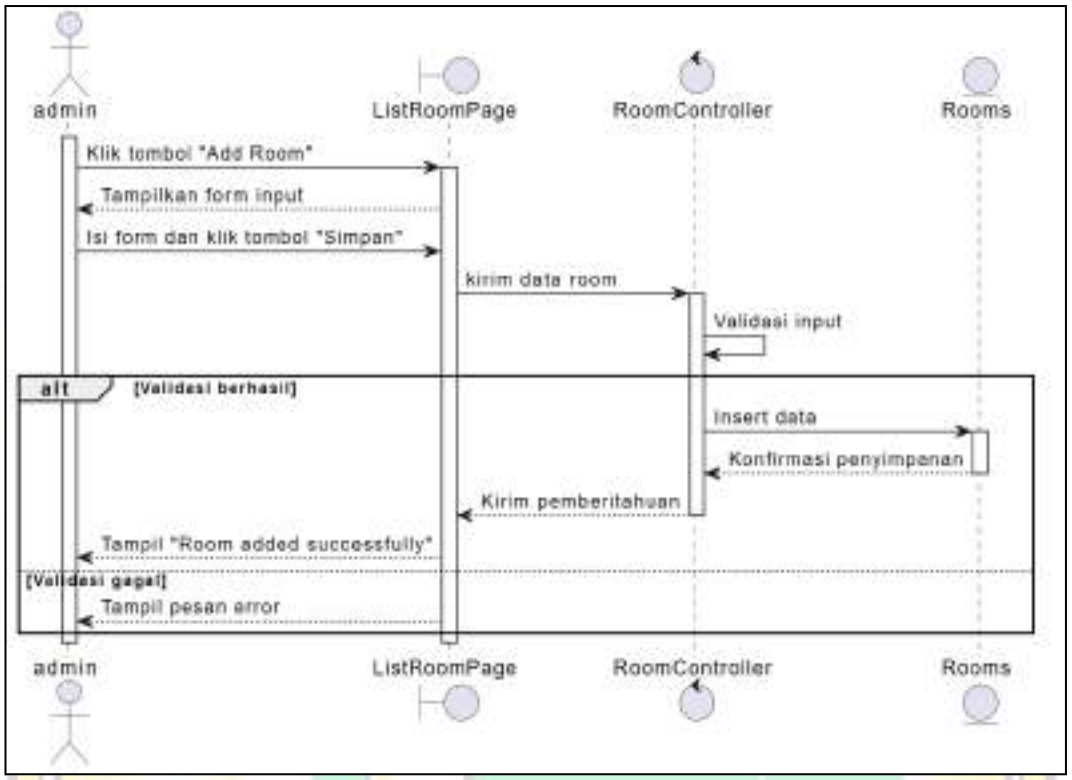
3. Admin dapat melihat *dashboard*



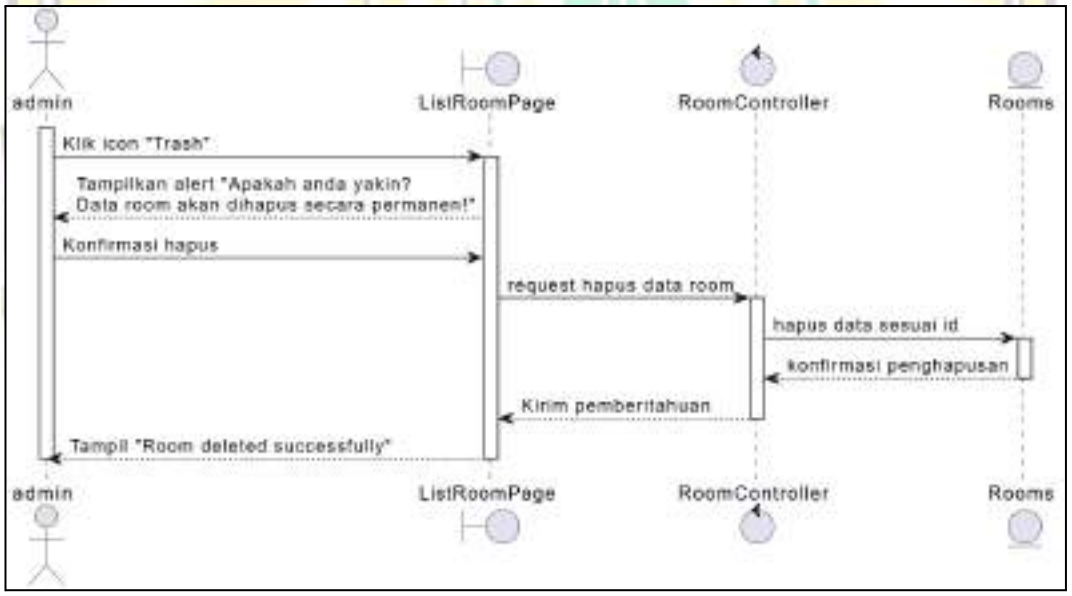
4. Admin dapat melihat data ruangan



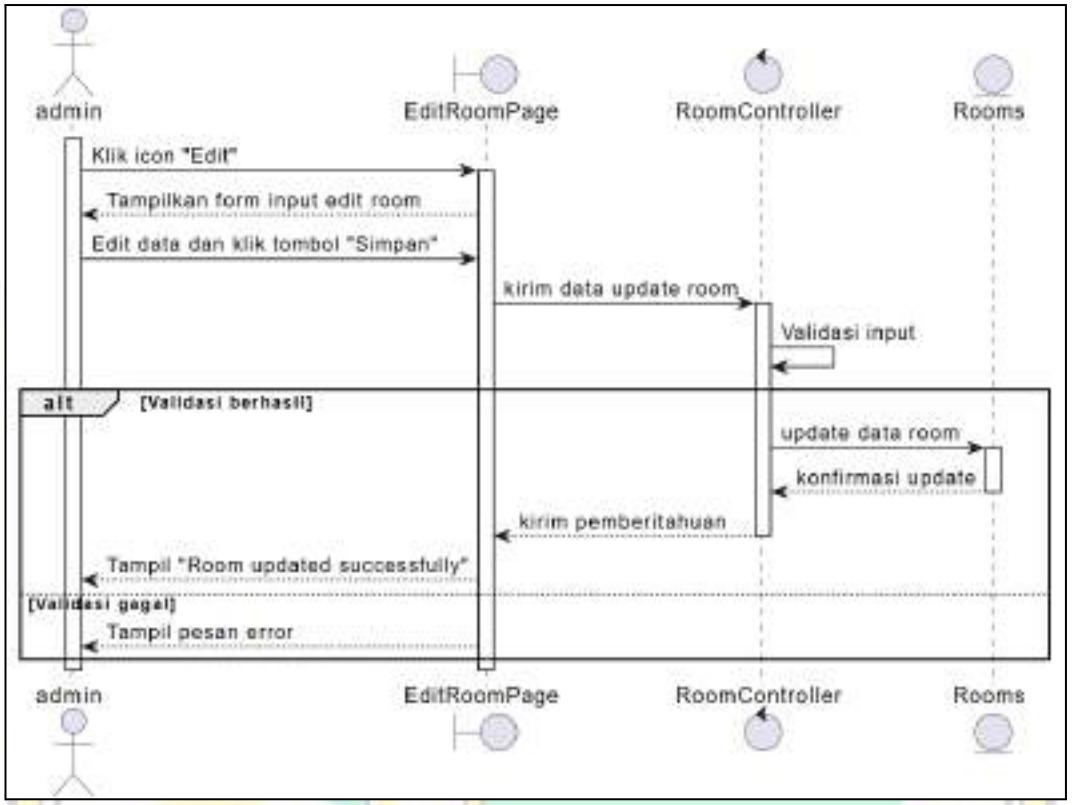
5. Admin dapat menambah data ruangan



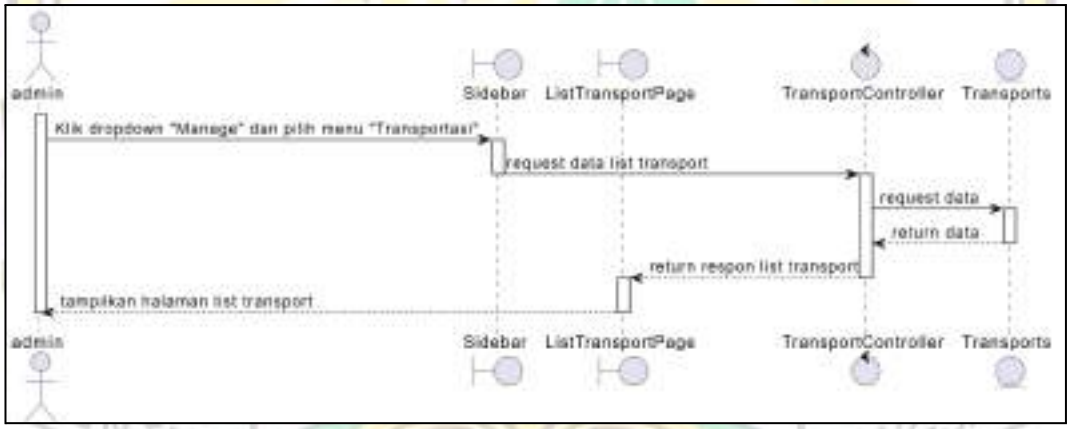
6. Admin dapat menghapus data ruangan



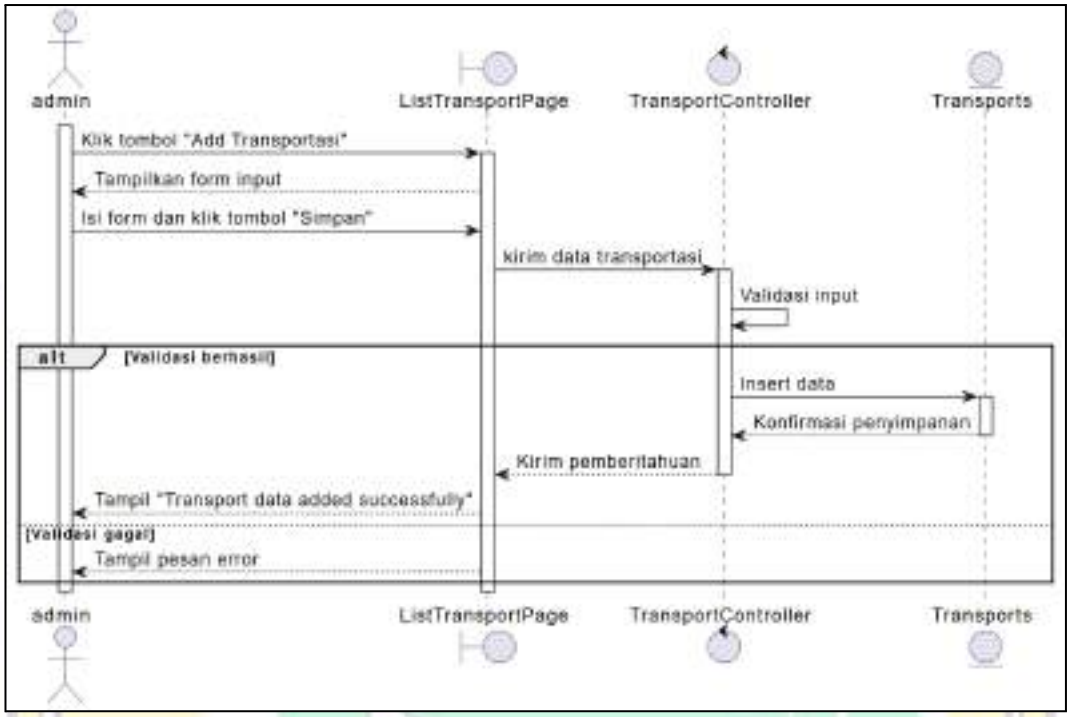
7. Admin dapat mengedit data ruangan



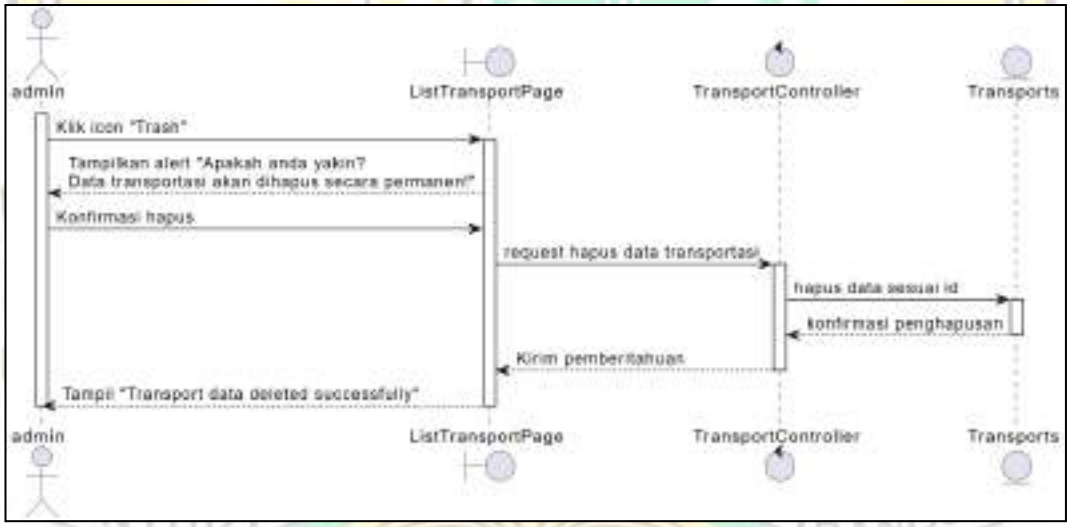
8. Admin dapat melihat data transportasi



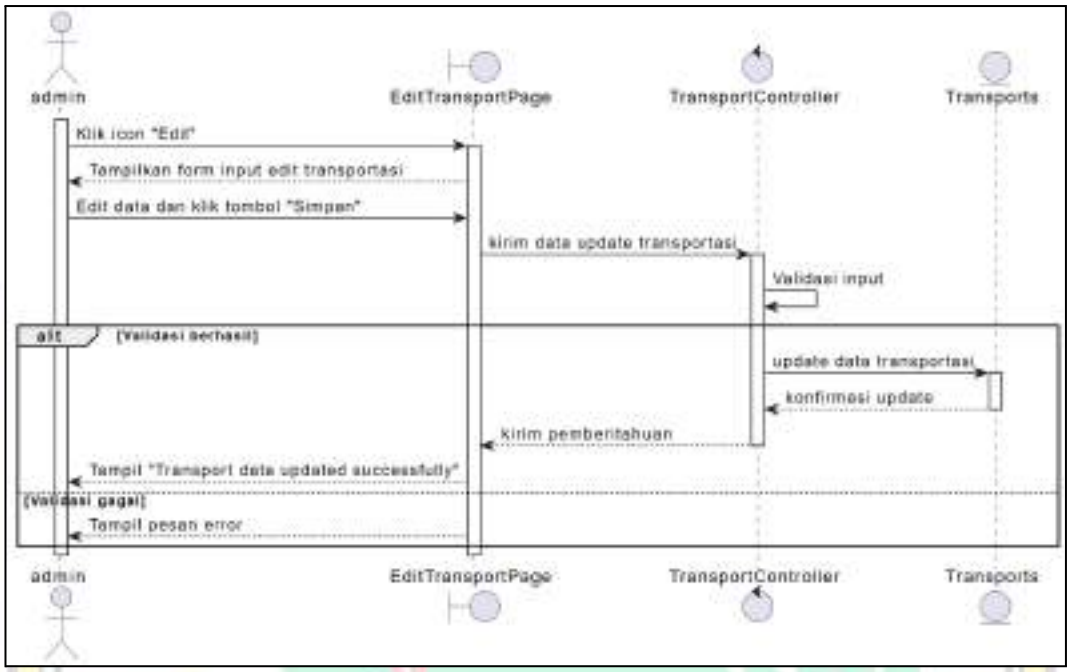
9. Admin dapat menambah data transportasi



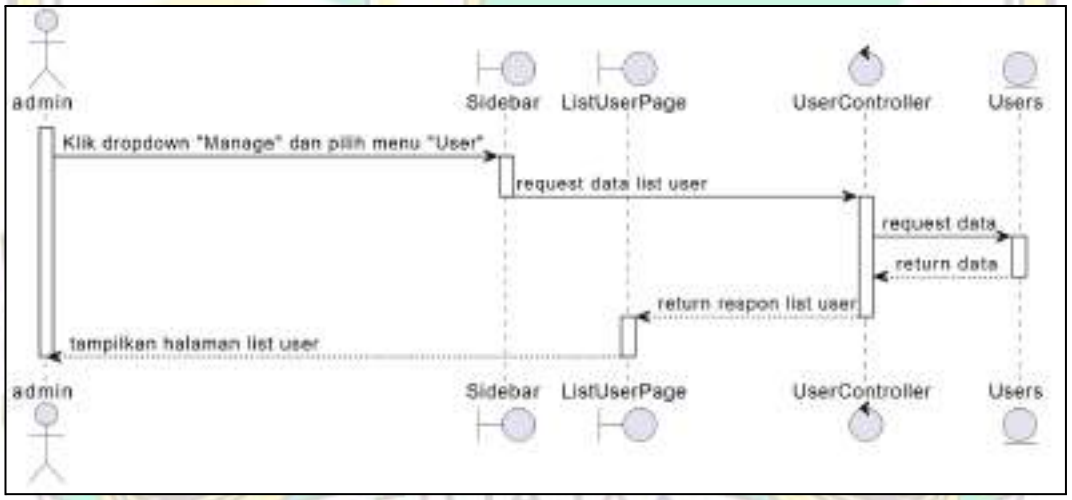
10. Admin dapat menghapus data transportasi



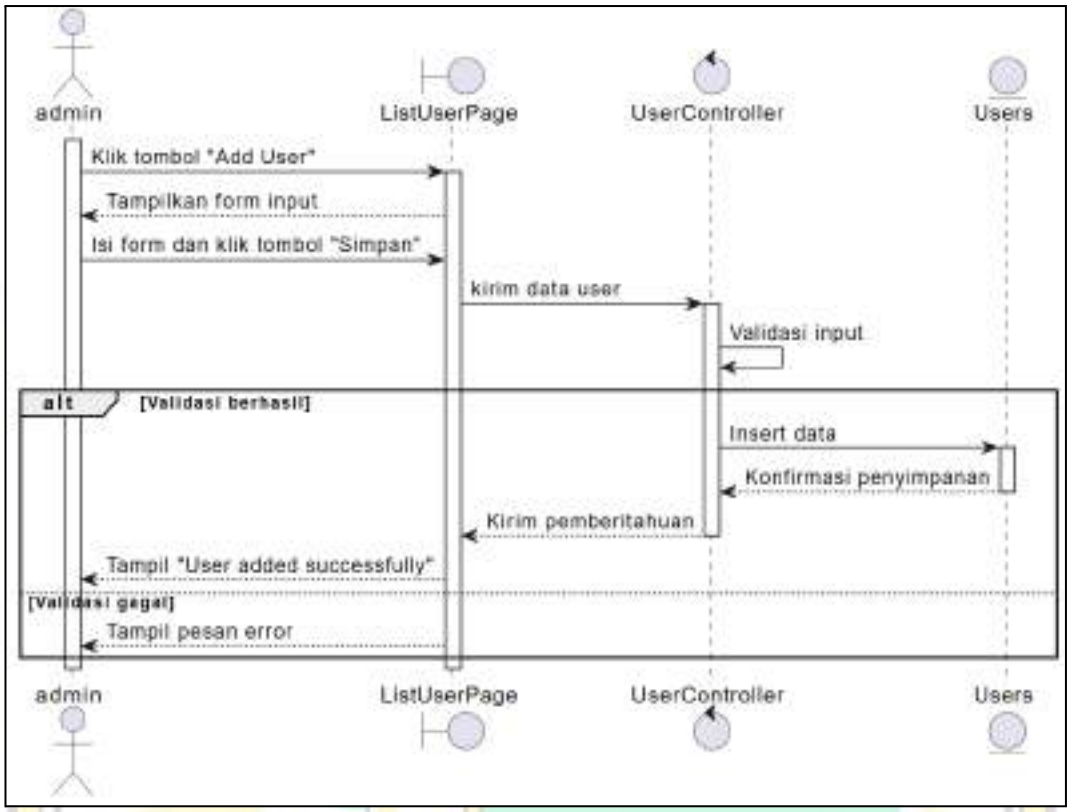
11. Admin dapat mengedit data transportasi



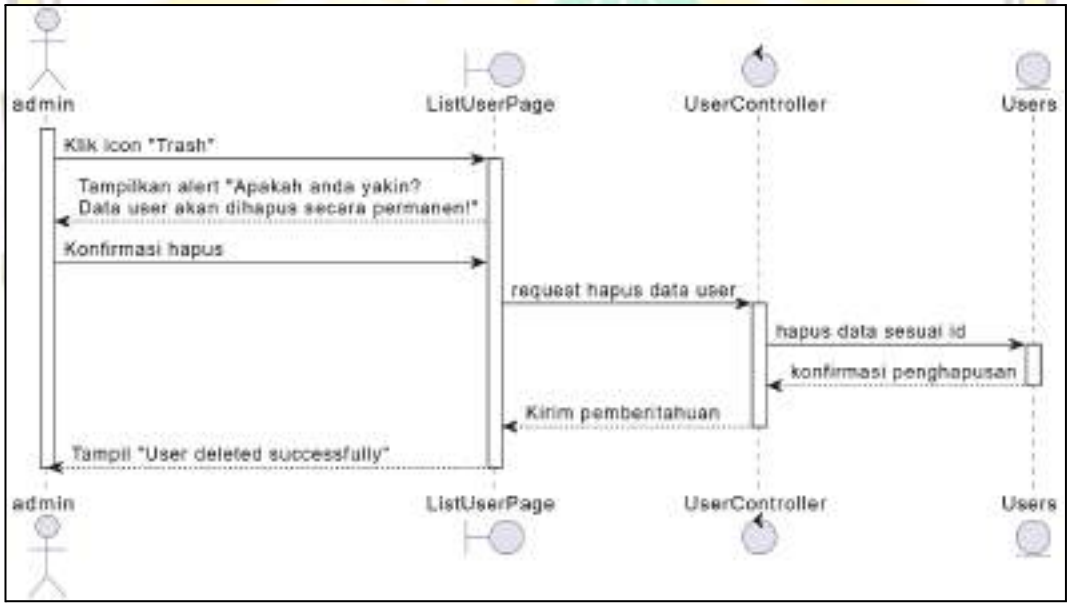
12. Admin dapat melihat data *user*



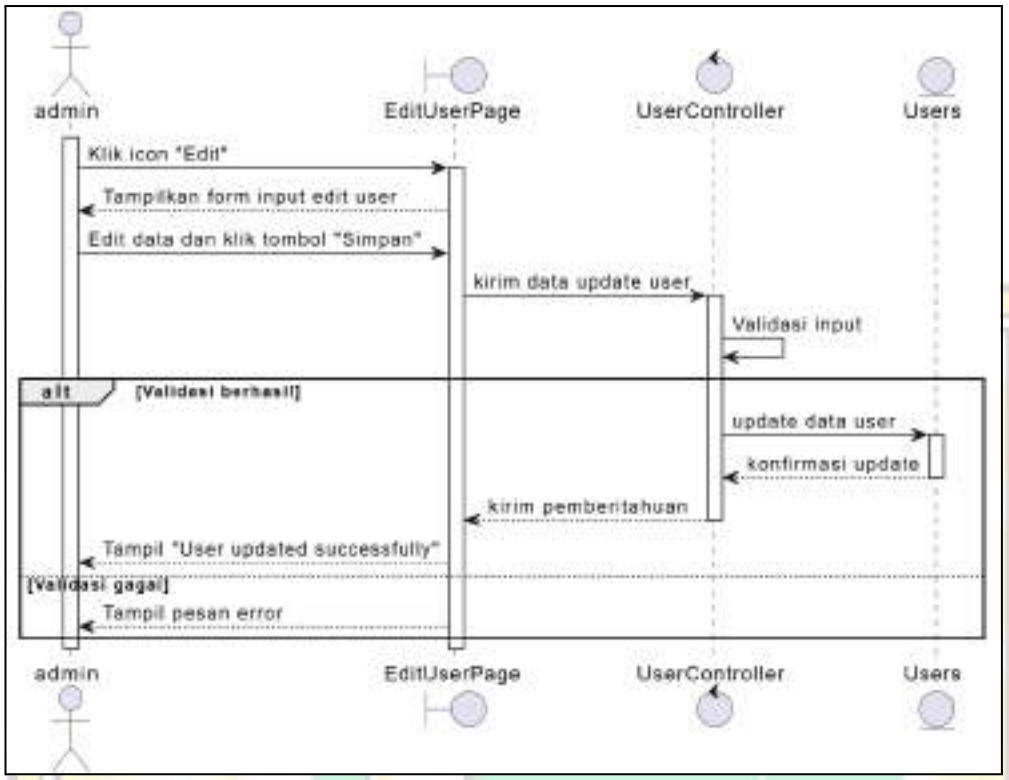
13. Admin dapat menambah data *user*



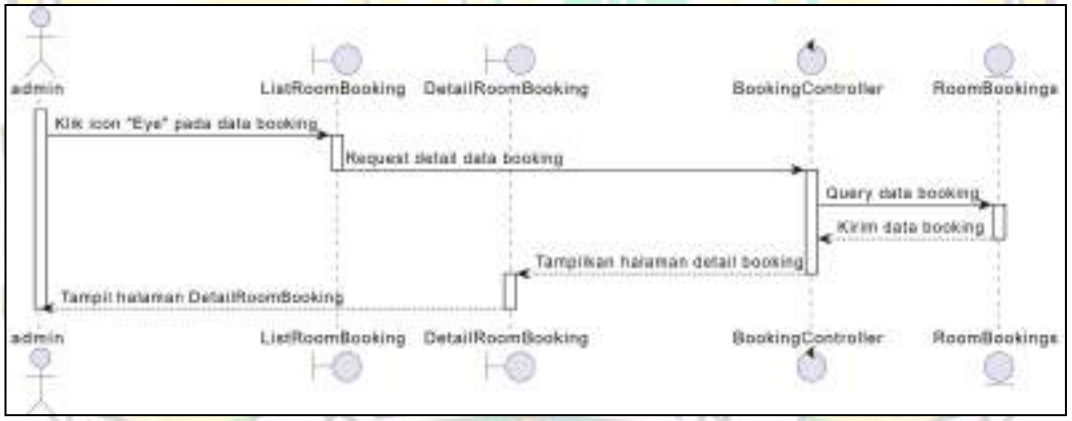
14. Admin dapat menghapus data *user*



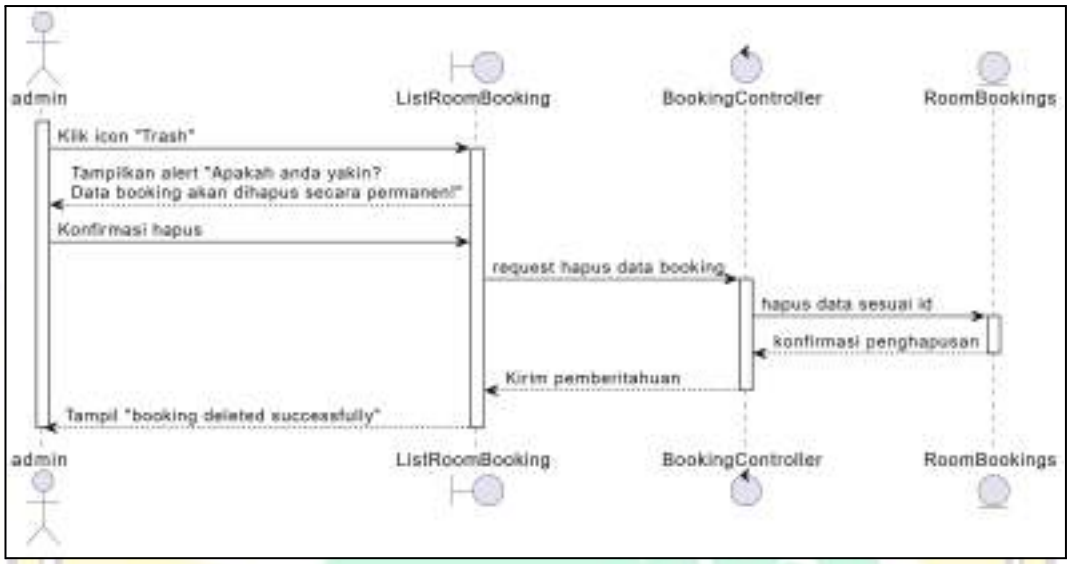
15. Admin dapat mengedit data *user*



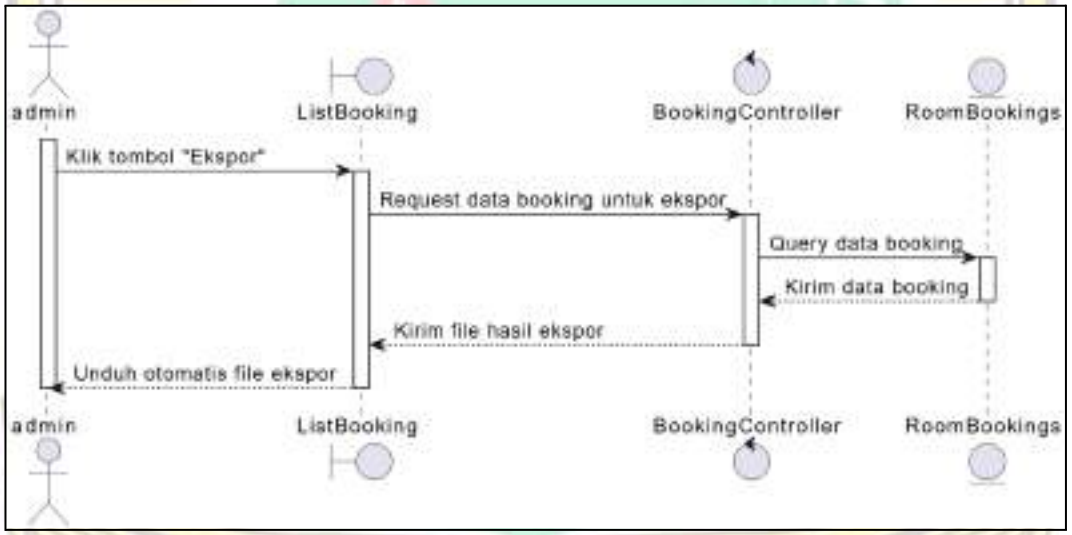
16. Admin dapat melihat detail *booking*



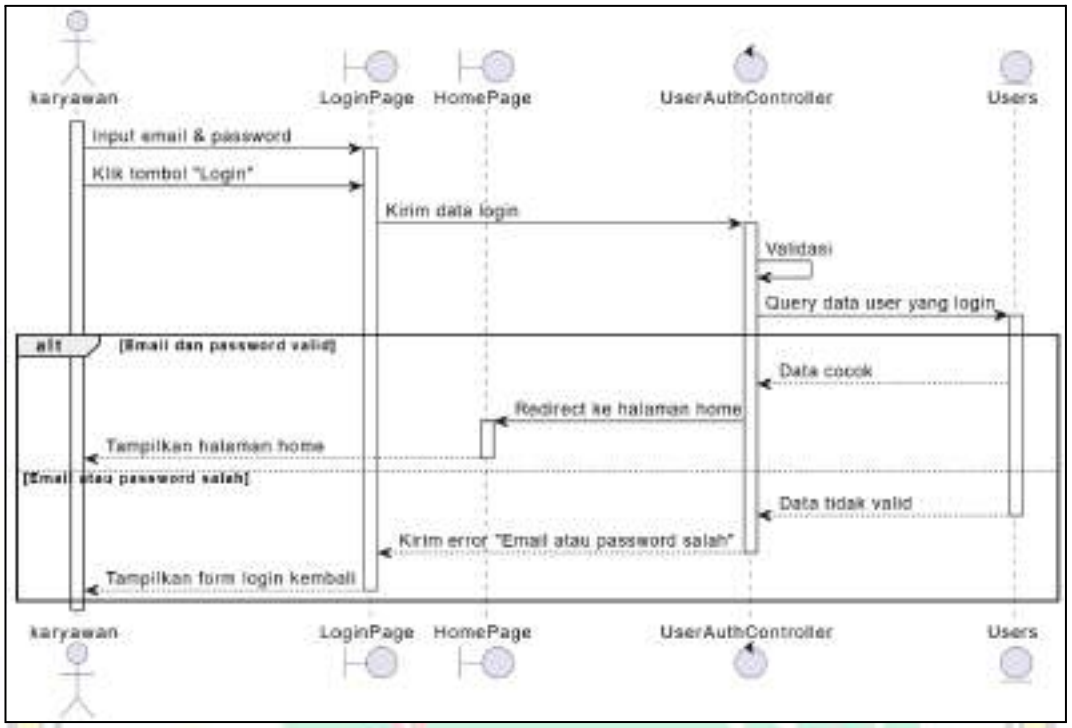
17. Admin dapat menghapus data *booking*



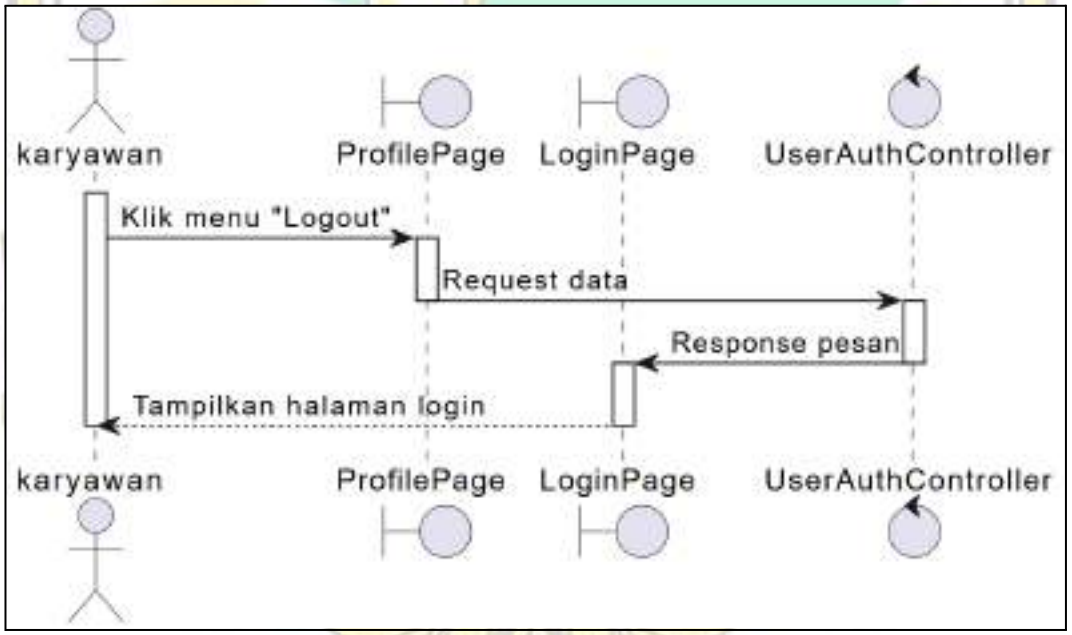
18. Admin dapat mengekspor data *booking*



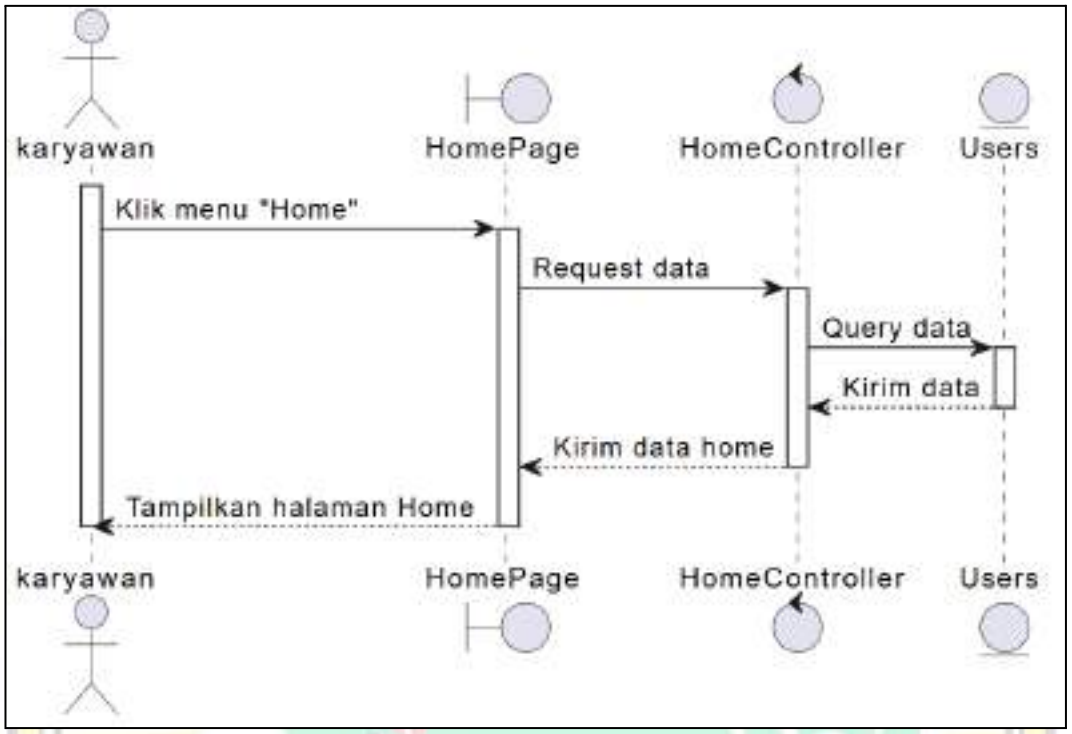
19. Karyawan dapat login aplikasi *mobile*



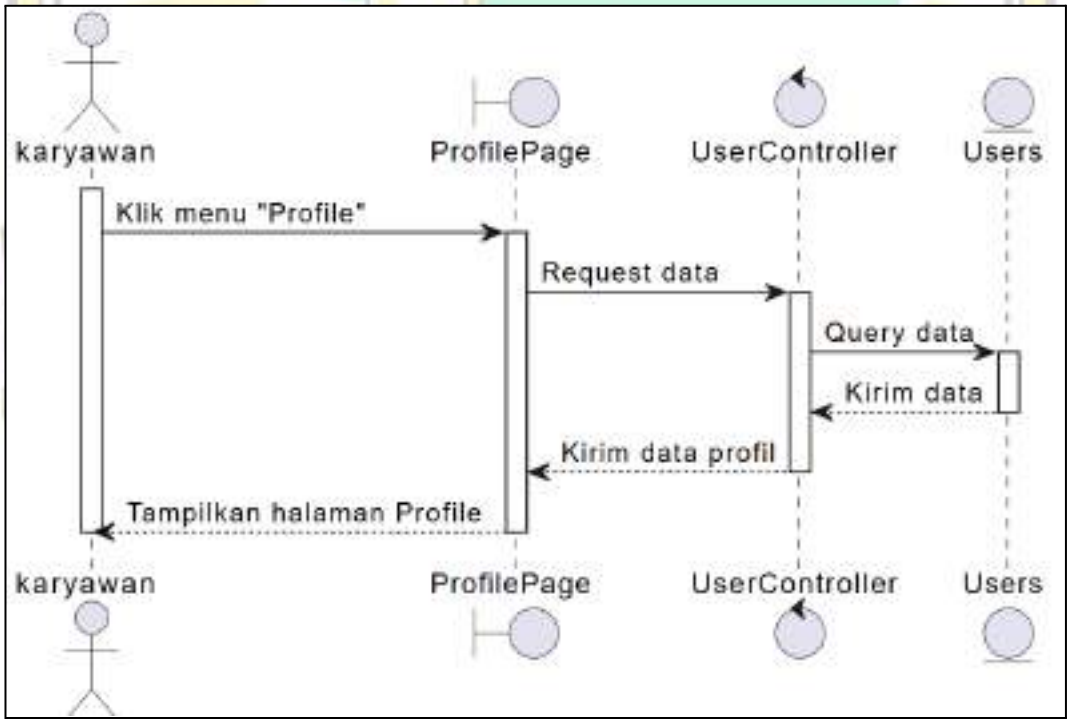
20. Karyawan dapat logout aplikasi *mobile*



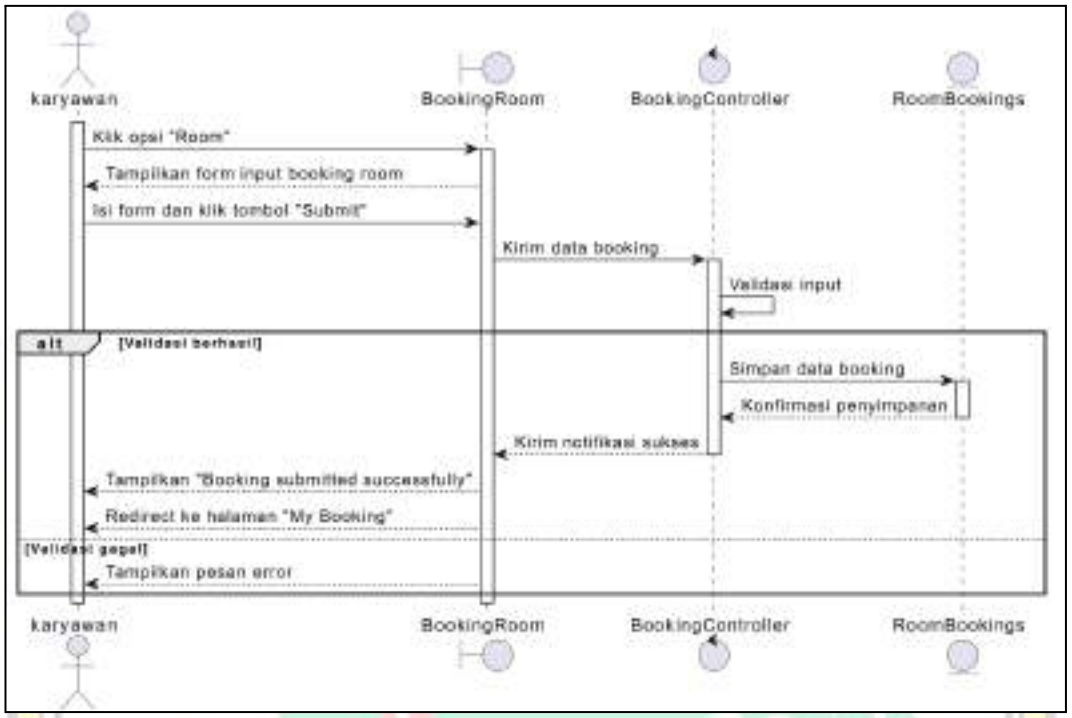
21. Karyawan dapat melihat halaman *home*



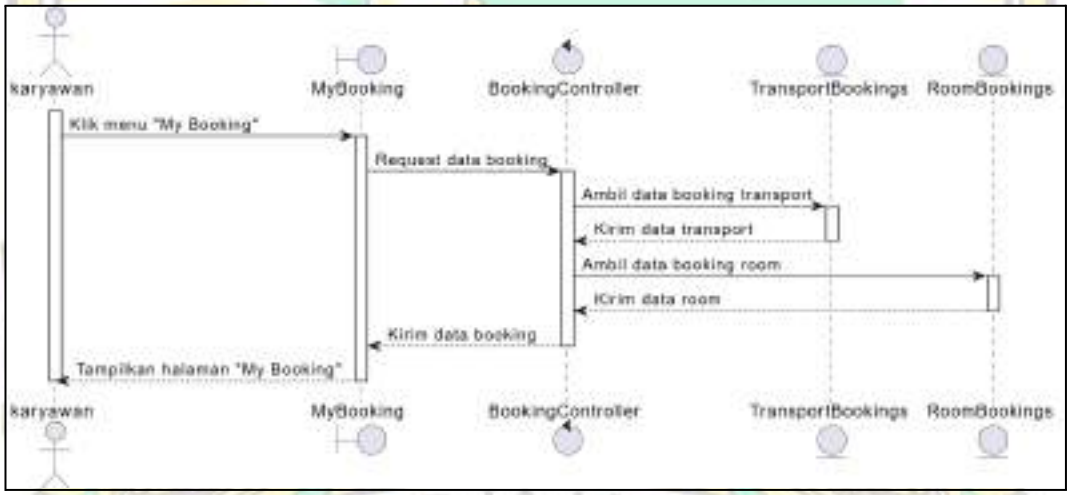
22. Karyawan dapat melihat *profile*



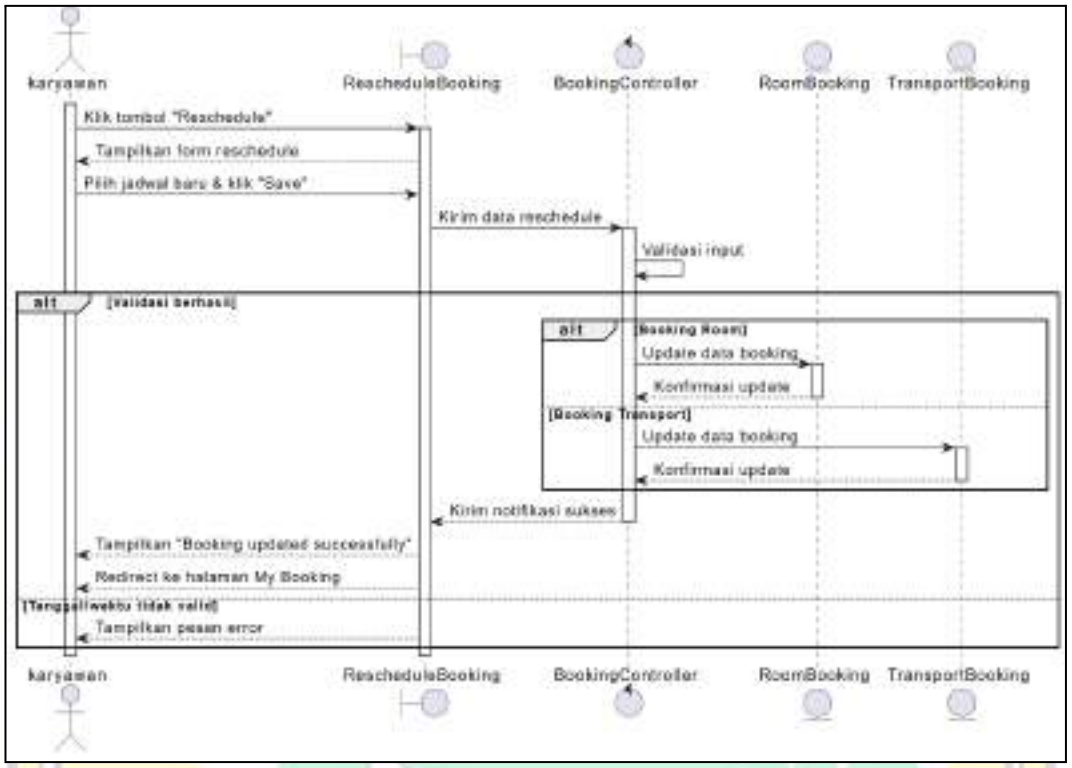
23. Karyawan dapat melakukan *booking* ruang rapat



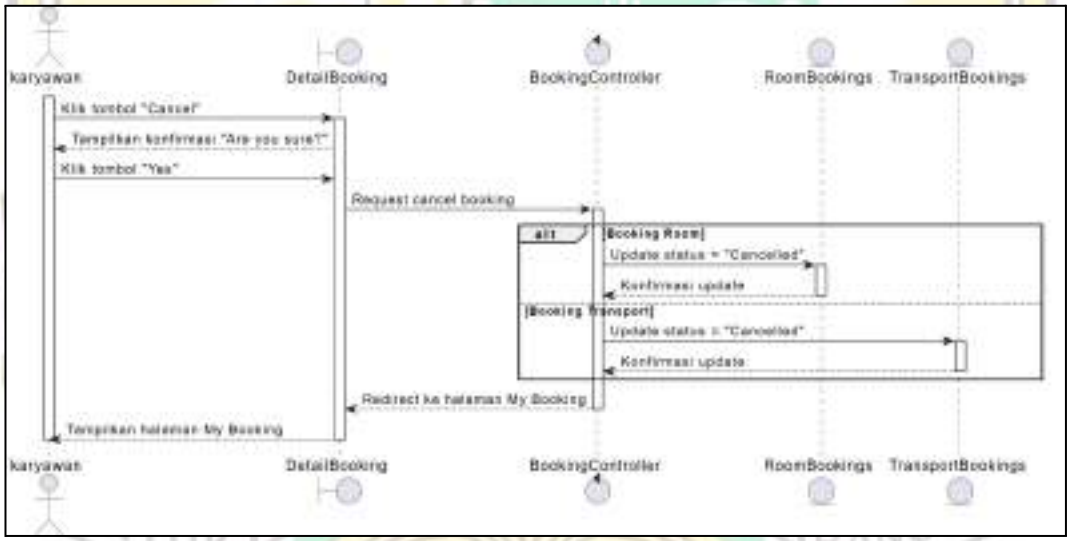
24. Karyawan dapat melihat data *my booking*



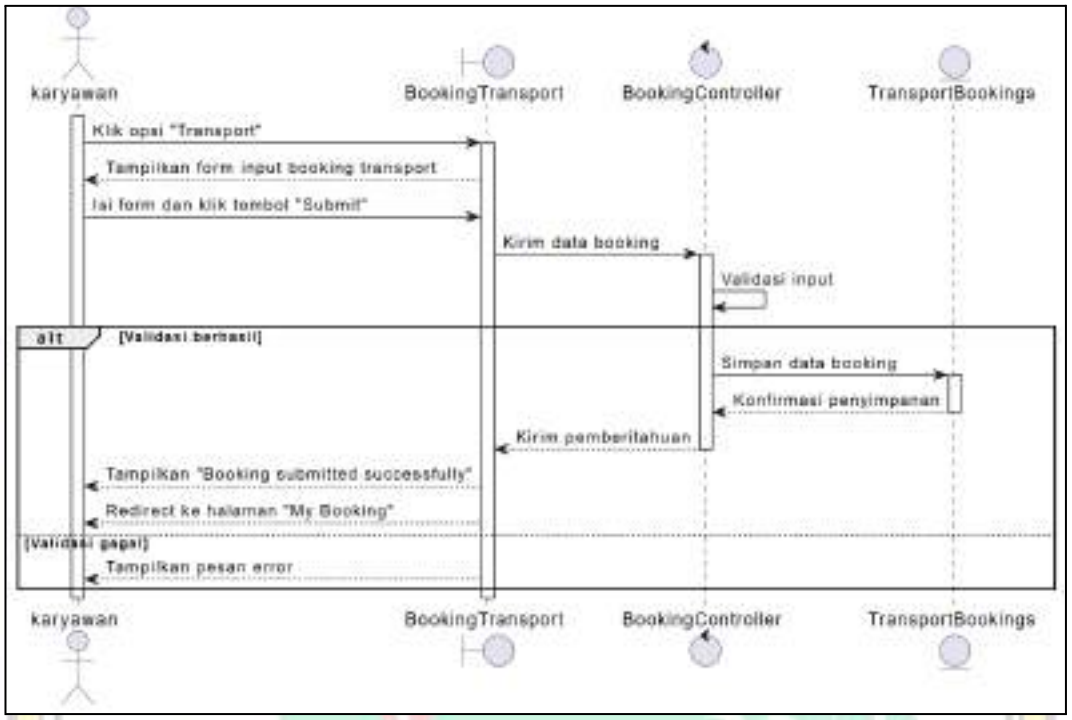
25. Karyawan dapat melakukan *reschedule booking*



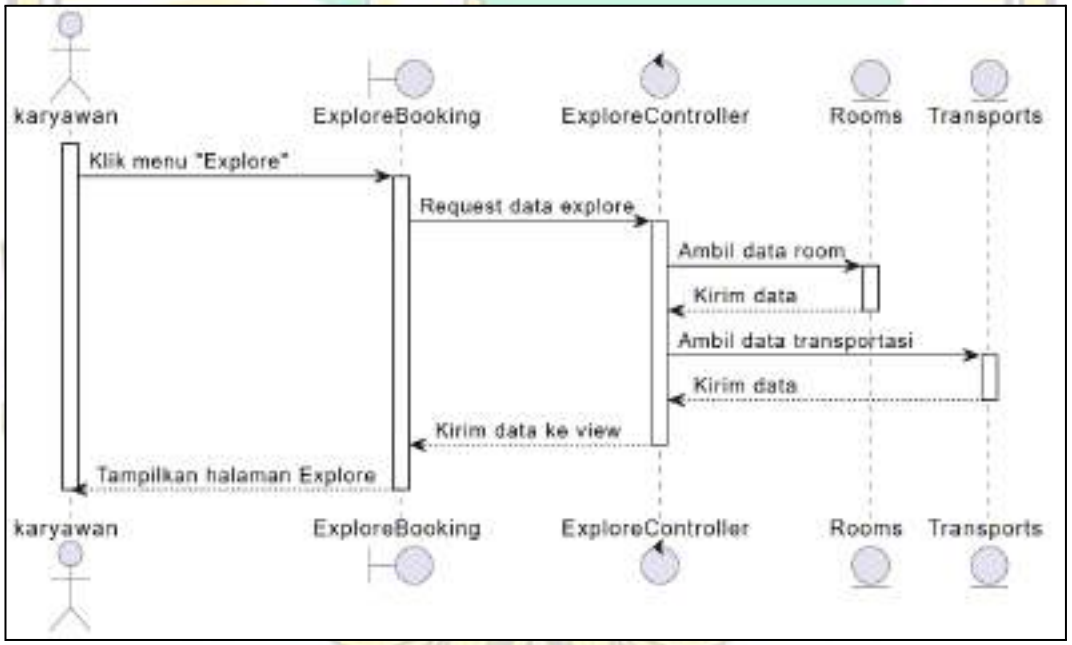
26. Karyawan dapat melakukan *cancel booking*



27. Karyawan dapat melakukan *booking* transportasi



28. Karyawan dapat melihat data *explore*



LAMPIRAN C

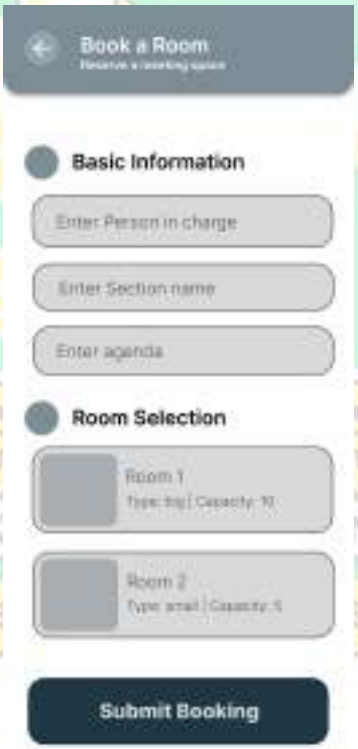
PERANCANGAN ANTARMUKA (LANJUTAN)

1. Rancangan User Interface Halaman Login

2. Rancangan User Interface Halaman Home

3. Rancangan User Interface Halaman Explore

4. Rancangan User Interface Halaman Form Booking



5. Rancangan User Interface Halaman Detail Booking

6. Rancangan User Interface Halaman My Booking

7. Rancangan User Interface Halaman Detail Fasilitas

8. Rancangan User Interface Halaman Reschedule

9. Rancangan User Interface Halaman Profile

LAMPIRAN D

KODE PROGRAM (LANJUTAN)

1. Backend Controller User

```
// backend/src/controllers/userController.ts

import { Request, Response } from 'express';

import path from 'path';
import fs from 'fs';
import UserService from '../services/userService';

class UserController {

public static async importFromExcel(req: Request, res: Response) {
  try {
    // Validasi ketersediaan file
    if (!req.file) {
      return res.status(400).json({
        success: false,
        error: 'Tidak ada file yang diunggah'
      });
    }

    // Validasi ekstensi file
    const fileExtension = path.extname(req.file.originalname).toLowerCase();
    if (![.xlsx', '.xls'].includes(fileExtension)) {
      // Hapus file jika tidak valid
      fs.unlinkSync(req.file.path);
      return res.status(400).json({
        success: false,
        error: 'Format file harus .xlsx atau .xls'
      });
    }

    // Validasi ukuran file (max 5MB)
    const fileSizeInMB = req.file.size / (1024 * 1024);
    if (fileSizeInMB > 5) {
      fs.unlinkSync(req.file.path);
      return res.status(400).json({
        success: false,
        error: 'Ukuran file terlalu besar. Maksimal 5MB'
      });
    }

    // Process file Excel dan import data pengguna
    const importResults = await UserService.importFromExcel(req.file.path);

    // Hapus file temporary setelah selesai
    fs.unlinkSync(req.file.path);

    // Response hasil import
    return res.status(200).json({
      success: true,
      message: 'Berhasil mengimpor pengguna dari Excel',
      data: {
        totalProcessed: importResults.totalProcessed,
        successCount: importResults.successCount,
        failedCount: importResults.failedCount,
```

```
        errors: importResults.errors
      }
    });
  } catch (error: any) {
    // Pastikan file temporary dihapus jika terjadi error
    if (req.file && fs.existsSync(req.file.path)) {
      fs.unlinkSync(req.file.path);
    }

    return res.status(500).json({
      success: false,
      error: error.message || 'Terjadi kesalahan saat mengimpor data'
    });
  }
}

public static async getAll(req: Request, res: Response) {
  try {
    const page = parseInt(req.query.page as string) || 1;
    const limit = parseInt(req.query.limit as string) || 10;
    const search = req.query.search as string || '';

    const result = await UserService.getAll(page, limit, search);
    res.status(200).json({
      message: 'Berhasil mendapatkan daftar pengguna',
      data: result.users,
      pagination: {
        total: result.total,
        page,
        limit,
        totalPages: Math.ceil(result.total / limit)
      }
    });
  } catch (error: any) {
    res.status(500).json({ error: error.message });
  }
}

// Get user by ID
public static async getById(req: Request, res: Response) {
  try {
    const userId = parseInt(req.params.id);
    const user = await UserService.getById(userId);
    res.status(200).json({
      message: 'Berhasil mendapatkan data pengguna',
      data: user
    });
  } catch (error: any) {
    res.status(error.message === 'Pengguna tidak ditemukan' ? 404 : 500).json({
      error: error.message
    });
  }
}

// Create new user (admin only)
public static async create(req: Request, res: Response) {
  try {
```

```
    const { name, email, password, phone } = req.body;
    const result = await UserService.create({ name, email, password, phone });
    res.status(201).json({
      message: 'Pengguna berhasil dibuat',
      data: {
        id: result.id,
        name: result.name,
        email: result.email,
        phone: result.phone,
        createdAt: result.createdAt
      }
    });
  } catch (error: any) {
    res.status(400).json({ error: error.message });
  }
}

// Update user
public static async update(req: Request, res: Response) {
  try {
    const userId = parseInt(req.params.id);
    const { name, email, phone } = req.body;

    // Check if user has permission (admin or own account)
    const requestingUserId = (req as any).user.id;


    const result = await UserService.update(userId, { name, email, phone });
    res.status(200).json({
      message: 'Pengguna berhasil diperbarui',
      data: {
        id: result.id,
        name: result.name,
        email: result.email,
        phone: result.phone,
        updatedAt: result.updatedAt
      }
    });
  } catch (error: any) {
    res.status(error.message === 'Pengguna tidak ditemukan' ? 404 : 400).json({
      error: error.message
    });
  }
}

// Delete user
public static async delete(req: Request, res: Response) {
  try {
    const userId = parseInt(req.params.id);

    await UserService.delete(userId);
    res.status(200).json({
      message: 'Pengguna berhasil dihapus'
    });
  } catch (error: any) {
    res.status(error.message === 'Pengguna tidak ditemukan' ? 404 : 500).json({
      error: error.message
```

```
    });
  }
 }

 // Get current user profile
 public static async getProfile(req: Request, res: Response) {
  try {
    const userId = (req as any).user.id;
    const user = await UserService.getById(userId);
    res.status(200).json({
      message: 'Berhasil mendapatkan profil',
      data: user
    });
  } catch (error: any) {
    res.status(500).json({ error: error.message });
  }
 }

 // Update current user profile
 public static async updateProfile(req: Request, res: Response) {
  try {
    const userId = (req as any).user.id;
    const { name, phone } = req.body;

    const result = await UserService.updateProfile(userId, { name, phone });
    res.status(200).json({
      message: 'Profil berhasil diperbarui',
      data: {
        id: result.id,
        name: result.name,
        email: result.email,
        phone: result.phone,
        updatedAt: result.updatedAt
      }
    });
  } catch (error: any) {
    res.status(400).json({ error: error.message });
  }
 }
}

export default UserController;
```

2. Backend Controller Admin

```
// src/controllers/adminController.ts
import { Request, Response } from 'express';
import Admin from '../models/Admin';
import bcrypt from 'bcryptjs';

/**
 * Get All Admins
 */
export const getAllAdmins = async (req: Request, res: Response) => {
  try {
```

```
    const admins = await Admin.findAll({
      attributes: ['admin_id', 'username', 'email', 'createdAt', 'updatedAt'],
    });

    return res.status(200).json({ admins });
  } catch (error) {
    console.error('Error fetching admins:', error);
    return res.status(500).json({ message: 'Internal server error.' });
  }
};

/**
 * Get Admin by ID
 */
export const getAdminById = async (req: Request, res: Response) => {
  const { admin_id } = req.params;

  try {
    const admin = await Admin.findByPk(admin_id, {
      attributes: ['admin_id', 'username', 'email', 'createdAt', 'updatedAt'],
    });

    if (!admin) {
      return res.status(404).json({ message: 'Admin tidak ditemukan.' });
    }

    return res.status(200).json({ admin });
  } catch (error) {
    console.error('Error fetching admin:', error);
    return res.status(500).json({ message: 'Internal server error.' });
  }
};

/**
 * Update Admin
 */
export const updateAdmin = async (req: Request, res: Response) => {
  const { admin_id } = req.params;
  const { username, email, password } = req.body;

  try {
    const admin = await Admin.findByPk(admin_id);
    if (!admin) {
      return res.status(404).json({ message: 'Admin tidak ditemukan.' });
    }

    if (username) admin.username = username;
    if (email) admin.email = email;
    if (password) {
      const salt = await bcrypt.genSalt(10);
      admin.password = await bcrypt.hash(password, salt);
    }

    await admin.save();

    return res.status(200).json({
      message: 'Admin berhasil diperbarui.',
```

```
      admin: {
        admin_id: admin.admin_id,
        username: admin.username,
        email: admin.email,
        createdAt: admin.createdAt,
        updatedAt: admin.updatedAt,
      },
    });
  } catch (error) {
    console.error('Error updating admin:', error);
    return res.status(500).json({ message: 'Internal server error.' });
  }
};

/**
 * Delete Admin
 */
export const deleteAdmin = async (req: Request, res: Response) => {
  const { admin_id } = req.params;

  try {
    const admin = await Admin.findByPk(admin_id);
    if (!admin) {
      return res.status(404).json({ message: 'Admin tidak ditemukan.' });
    }

    await admin.destroy();

    return res.status(200).json({ message: 'Admin berhasil dihapus.' });
  } catch (error) {
    console.error('Error deleting admin:', error);
    return res.status(500).json({ message: 'Internal server error.' });
  }
};
```

3. Backend Controller User Auth

```
import { Request, Response } from 'express';
import AuthService from '../services/authService';

class AuthController {

  public static async editProfile(req: Request, res: Response) {
    try {
      const userId = (req as any).user.id; // Assuming the user ID is added to the request by an
authentication middleware
      const { name, email, phone, password } = req.body;

      // Call the AuthService to update the user's profile
      const result = await AuthService.editProfile(userId, { name, email, phone, password });

      res.status(200).json({
        message: 'Profil berhasil diperbarui',
        data: result,
      });
```

```
    } catch (error: any) {
      res.status(400).json({ error: error.message });
    }
  }
  // Handler untuk registrasi
  public static async register(req: Request, res: Response) {
    try {
      const { name, email, password, phone } = req.body;
      const result = await AuthService.register({ name, email, password, phone });
      res.status(201).json({
        message: 'Registrasi berhasil',
        data: {
          user: {
            id: result.user.id,
            name: result.user.name,
            email: result.user.email,
            phone: result.user.phone,
            createdAt: result.user.createdAt,
          },
          token: result.token,
        },
      });
    } catch (error: any) {
      res.status(400).json({ error: error.message });
    }
  }

  // Handler untuk login
  public static async login(req: Request, res: Response) {
    try {
      const { email, password } = req.body;
      const result = await AuthService.login({ email, password });
      res.status(200).json({
        message: 'Login berhasil',
        data: {
          user: {
            id: result.user.id,
            name: result.user.name,
            email: result.user.email,
            phone: result.user.phone,
            createdAt: result.user.createdAt,
          },
          token: result.token,
        },
      });
    } catch (error: any) {
      res.status(400).json({ error: error.message });
    }
  }

  // Handler untuk mengganti password
  public static async changePassword(req: Request, res: Response) {
    try {
      const { currentPassword, newPassword } = req.body;
      const userId = (req as any).user.id; // Assuming the user ID is added to the request by an
authentication middleware
      const result = await AuthService.changePassword(userId, currentPassword, newPassword);
```

```
    res.status(200).json(result);
   } catch (error: any) {
    res.status(400).json({ error: error.message });
   }
  }
}

export default AuthController;
```

4. Backend Controller Admin Auth

```
// src/controllers/adminAuthController.ts
import { Request, Response } from 'express';
import Admin from '../models/Admin';
import bcrypt from 'bcryptjs';
import jwt from 'jsonwebtoken';
import dotenv from 'dotenv';
import { generateToken } from '../utils/generateToken';
dotenv.config();


export const editProfile = async (req: Request, res: Response) => {
 const { username, email, password, admin_id: requestAdminId } = req.body;

 // Get admin_id from middleware OR from request body as fallback
 const admin_id = req.admin_id || requestAdminId;

 // Debug information
 console.log("Edit profile request:", {
  admin_id_from_middleware: req.admin_id,
  admin_id_from_request: requestAdminId,
  admin_id_used: admin_id
 });

 // Ensure admin_id is available
 if (!admin_id) {
  return res.status(400).json({ message: 'ID Admin tidak tersedia. Silakan login kembali.' });
 }

 try {
  // Cari Admin berdasarkan ID
  const admin = await Admin.findByPk(admin_id);
  if (!admin) {
   return res.status(404).json({
    message: 'Admin tidak ditemukan.',
    details: { admin_id }
   });
  }

  // Jika email baru disertakan, cek apakah sudah ada yang menggunakan email tersebut
  if (email && email !== admin.email) {
   const existingAdmin = await Admin.findOne({ where: { email } });
   if (existingAdmin && existingAdmin.admin_id !== admin_id) {
    return res.status(400).json({ message: 'Email tersebut sudah terdaftar.' });
   }
```

```
    admin.email = email;
  }

  // Jika password baru disertakan, hash password baru
  if (password) {
    const salt = await bcrypt.genSalt(10);
    admin.password = await bcrypt.hash(password, salt);
  }

  // Update username jika ada perubahan
  if (username) {
    admin.username = username;
  }

  // Simpan perubahan
  await admin.save();

  // Buat token baru
  const token = jwt.sign(
    { id: admin.admin_id, role: 'admin' },
    process.env.JWT_SECRET as string,
    { expiresIn: '1h' }
  );

  return res.status(200).json({
    message: 'Profil admin berhasil diperbarui.',
    token,
    admin: {
      id: admin.admin_id,
      username: admin.username,
      email: admin.email,
    },
  });
 } catch (error) {
  console.error('Error during profile update:', error);
  return res.status(500).json({
    message: 'Internal server error.',
    details: error.message
  });
 }
};

/**
 * Register Admin
 */
export const adminRegister = async (req: Request, res: Response) => {
 const { username, email, password } = req.body;

 try {
  // Cek apakah Admin sudah ada
  const existingAdmin = await Admin.findOne({ where: { email } });
  if (existingAdmin) {
    return res.status(400).json({ message: 'Admin dengan email tersebut sudah terdaftar.' });
  }

  // Hash password
  const salt = await bcrypt.genSalt(10);
```

```
    const hashedPassword = await bcrypt.hash(password, salt);

    // Buat Admin baru
    const admin = await Admin.create({
      username,
      email,
      password: hashedPassword,
    });

    // Buat token
    const token = jwt.sign(
      { id: admin.admin_id, role: 'admin' },
      process.env.JWT_SECRET as string,
      { expiresIn: '1h' }
    );

    return res.status(201).json({
      message: 'Admin berhasil dibuat.',
      token,
      admin: {
        id: admin.admin_id,
        username: admin.username,
        email: admin.email,
      },
    });
  } catch (error) {
    console.error('Error during admin registration:', error);
    return res.status(500).json({ message: 'Internal server error.' });
  }
};

/**
 * Admin Login
 */
export const adminLogin = async (req: Request, res: Response) => {
    const { email, password } = req.body;

    try {
      // Cari Admin berdasarkan email
      const admin = await Admin.findOne({ where: { email } });
      if (!admin) {
        return res.status(400).json({ message: 'Email atau password salah.' });
      }

      // Cek password
      const isMatch = await bcrypt.compare(password, admin.password);
      if (!isMatch) {
        return res.status(400).json({ message: 'Email atau password salah.' });
      }

      // Buat token dengan peran 'admin'
      const token = generateToken({ id: admin.admin_id, role: 'admin' });

      return res.status(200).json({
        message: 'Login berhasil.',
        token,
        admin: {
```

```
        id: admin.admin_id,
        username: admin.username,
        email: admin.email,
      },
    });
  } catch (error) {
    console.error('Error during admin login:', error);
    return res.status(500).json({ message: 'Internal server error.' });
  }
};
```

5. Backend Controller Room

```
// RoomController.ts - Updated with consistent path handling
import { Request, Response } from 'express';
import RoomService from '../services/RoomService';
import upload from '../utils/multerConfig';
import fs from 'fs';
import path from 'path';
import dotenv from 'dotenv';

// Load environment variables
dotenv.config();

// Get server URL from environment variables or use a default
const SERVER_URL = process.env.SERVER_URL;

// Use the same uploads directory path as in app.js and multerConfig.ts
const uploadsDir = path.join(__dirname, '../utils/uploads');

class RoomController {
  // Create Room with image upload
  public static async createRoom(req: Request, res: Response) {
    upload.single('image')(req, res, async (err: any) => {
      if (err) {
        return res.status(400).json({ error: err.message });
      }

      try {
        let imageUrl = null;

        if (req.file) {
          // Log file details for debugging
          console.log(`File uploaded: ${JSON.stringify(req.file)}`);
          console.log(`File saved at: ${req.file.path}`);

          // Verify file exists after saving
          if (fs.existsSync(req.file.path)) {
            console.log(`Verified file exists at: ${req.file.path}`);

            // Create a proper URL with the server domain
            imageUrl = `/uploads/${req.file.filename}`;
            console.log(`Image URL created: ${imageUrl}`);
          } else {
            console.error(`WARNING: File does not exist at path: ${req.file.path}`);
```

```
          return res.status(500).json({ error: 'Failed to save image file' });
        }
      }

      const roomData = { ...req.body, image: imageUrl };
      const room = await RoomService.createRoom(roomData);
      res.status(201).json(room);
    } catch (error: any) {
      // If there was an error and we uploaded a file, clean it up
      if (req.file && req.file.path) {
        try {
          fs.unlinkSync(req.file.path);
          console.log(`Deleted file after error: ${req.file.path}`);
        } catch (unlinkErr) {
          console.error('Failed to clean up file after error:', unlinkErr);
        }
      }
      res.status(400).json({ error: error.message });
    }
  });
}

// Get all rooms
public static async getAllRooms(req: Request, res: Response) {
  try {
    const rooms = await RoomService.getAllRooms();

    // Make sure all image URLs are properly formatted
    const formattedRooms = rooms.map(room => {
      if (room.image) {
        // Check if image URL is already a full URL
        if (!room.image.startsWith('http')) {
          room.image = `${SERVER_URL}${room.image.startsWith('/') ? '' : '/'}${room.image}`;
        }

        // Check if image file exists on server (for debugging)
        try {
          const imageName = room.image.split('/').pop();
          const localPath = path.join(uploadsDir, imageName);
          const exists = fs.existsSync(localPath);
          console.log(`Image ${room.image} exists on server: ${exists ? 'YES' : 'NO'}, local path:
${localPath}`);
        } catch (e) {
          console.error(`Error checking image existence: ${e.message}`);
        }
      }
      return room;
    });

    res.status(200).json(formattedRooms);
  } catch (error: any) {
    res.status(400).json({ error: error.message });
  }
}

// Get room by ID
public static async getRoomById(req: Request, res: Response) {
```

```
  try {
    const room = await RoomService.getRoomById(Number(req.params.id));
    if (room) {
      // Ensure the image URL is properly formatted
      if (room.image && !room.image.startsWith('http')) {
        room.image = `${SERVER_URL}${room.image.startsWith('/') ? '' : '/'}${room.image}`;
      }

      res.status(200).json(room);
    } else {
      res.status(404).json({ error: 'Room not found' });
    }
  } catch (error: any) {
    res.status(400).json({ error: error.message });
  }
}

// Update Room with image upload
public static async updateRoom(req: Request, res: Response) {
  try {
    const existingRoom = await RoomService.getRoomById(Number(req.params.id));

    if (!existingRoom) {
      return res.status(404).json({ error: 'Room not found' });
    }

    upload.single('image')(req, res, async (err: any) => {
      if (err) {
        return res.status(400).json({ error: err.message });
      }

      try {
        const roomData: any = { ...req.body };

        // Handle image upload if a new file was provided
        if (req.file) {
          console.log(`New file uploaded for update: ${JSON.stringify(req.file)}`);

          // Verify the new file exists
          if (fs.existsSync(req.file.path)) {
            console.log(`Verified new file exists at: ${req.file.path}`);

            // Use full URL with server domain
            roomData.image = `/uploads/${req.file.filename}`;
            console.log(`New image URL: ${roomData.image}`);

            // Try to delete the old image if it exists
            if (existingRoom.image) {
              try {
                const oldFilename = existingRoom.image.split('/').pop();
                if (oldFilename) {
                  const oldImagePath = path.join(uploadsDir, oldFilename);
                  console.log(`Trying to delete old image: ${oldImagePath}`);

                  if (fs.existsSync(oldImagePath)) {
                    fs.unlinkSync(oldImagePath);
                    console.log(`Successfully deleted old image: ${oldImagePath}`);
```

```
              } else {
                console.warn(`Old image file not found: ${oldImagePath}`);
              }
            }
          } catch (unlinkErr) {
            console.error('Error deleting old image file:', unlinkErr);
          }
        }
      } else {
        console.error(`WARNING: New file does not exist at: ${req.file.path}`);
        return res.status(500).json({ error: 'Failed to save uploaded image' });
      }
    }

    const updatedRoom = await RoomService.updateRoom(Number(req.params.id),
roomData);
    if (updatedRoom) {
      res.status(200).json(updatedRoom);
    } else {
      if (req.file && req.file.path) {
        fs.unlinkSync(req.file.path);
        console.log(`Deleted new file after update failure: ${req.file.path}`);
      }
      res.status(404).json({ error: 'Room not found or update failed' });
    }
  } catch (error: any) {
    // Cleanup if error occurs during processing
    if (req.file && req.file.path) {
      try {
        fs.unlinkSync(req.file.path);
        console.log(`Deleted file after error: ${req.file.path}`);
      } catch (unlinkErr) {
        console.error('Failed to clean up file after error:', unlinkErr);
      }
    }
    res.status(400).json({ error: error.message });
  }
 });
 } catch (error: any) {
  res.status(400).json({ error: error.message });
 }
}

// Delete Room
public static async deleteRoom(req: Request, res: Response) {
 try {
  // First get the room to find its image
  const room = await RoomService.getRoomById(Number(req.params.id));

  if (!room) {
   return res.status(404).json({ error: 'Room not found' });
  }

  // Delete the image file if it exists
  if (room.image) {
   try {
    const filename = room.image.split('/').pop();
```

```
        if (filename) {
          const imagePath = path.join(uploadsDir, filename);
          console.log(`Attempting to delete image: ${imagePath}`);

          if (fs.existsSync(imagePath)) {
            fs.unlinkSync(imagePath);
            console.log(`Successfully deleted image: ${imagePath}`);
          } else {
            console.warn(`Image file not found for deletion: ${imagePath}`);
          }
        }
      } catch (unlinkErr) {
        console.error('Error deleting image file:', unlinkErr);
      }
    }

    // Delete the room from database
    const deleted = await RoomService.deleteRoom(Number(req.params.id));

    if (deleted) {
      res.status(200).json({ message: 'Room deleted successfully' });
    } else {
      res.status(500).json({ error: 'Failed to delete room' });
    }
  } catch (error: any) {
    res.status(400).json({ error: error.message });
  }
}

// Helper method to ensure uploads directory exists
private static ensureUploadsDir() {
  if (!fs.existsSync(uploadsDir)) {
    console.log(`RoomController: Creating uploads directory at ${uploadsDir}`);
    fs.mkdirSync(uploadsDir, { recursive: true, mode: 0o755 });
  }
}
}

// Ensure uploads directory exists when controller is loaded
RoomController.ensureUploadsDir();

export default RoomController;
```

6. Backend Controller Transport

```
// TransportController.ts
import { Request, Response } from 'express';
import TransportService from '../services/TransportService';
import upload from '../utils/multerConfig';
import fs from 'fs';
import path from 'path';
import dotenv from 'dotenv';

// Load environment variables
dotenv.config();
```

```
// Get server URL from environment variables or use a default
const SERVER_URL = process.env.SERVER_URL;

// IMPORTANT: This path must match the one in app.js and multerConfig.ts
const uploadsDir = path.join(__dirname, '../utils/uploads');

class TransportController {
  // Create Transport with image upload
  public static async createTransport(req: Request, res: Response) {
    upload.single('image')(req, res, async (err: any) => {
      if (err) {
        return res.status(400).json({ error: err.message });
      }

      try {
        let imageUrl = null;

        if (req.file) {
          // Log file details for debugging
          console.log(`File uploaded: ${JSON.stringify(req.file)}`);
          console.log(`File saved at: ${req.file.path}`);

          // Verify file exists after saving
          if (fs.existsSync(req.file.path)) {
            console.log(`Verified file exists at: ${req.file.path}`);

            // Make sure URL doesn't have double slashes
            // Store the relative URL path in the database, not the full server URL
            imageUrl = `/uploads/${req.file.filename}`;
            console.log(`Image URL created: ${imageUrl}`);
          } else {
            console.error(`WARNING: File does not exist at path: ${req.file.path}`);
            return res.status(500).json({ error: 'Failed to save image file' });
          }
        }

        const transportData = { ...req.body, image: imageUrl };
        const transport = await TransportService.createTransport(transportData);

        // For the response, we can add the full URL
        if (transport && transport.image) {
          // Keep the original relative path in the database record
          // But return the full URL in the API response
          const fullImageUrl = `${SERVER_URL}${transport.image}`;
          console.log(`Transport created with full image URL: ${fullImageUrl}`);

          // Create a new object with the full URL for the response
          const responseTransport = {
            ...transport.toJSON(),
            image: fullImageUrl
          };

          res.status(201).json(responseTransport);
        } else {
          res.status(201).json(transport);
        }
```

```
    } catch (error: any) {
      // If there was an error and we uploaded a file, clean it up
      if (req.file && req.file.path) {
        try {
          fs.unlinkSync(req.file.path);
          console.log(`Deleted file after error: ${req.file.path}`);
        } catch (unlinkErr) {
          console.error('Failed to clean up file after error:', unlinkErr);
        }
      }
      res.status(400).json({ error: error.message });
    }
  });
 }

 // Get all transports
 public static async getAllTransports(req: Request, res: Response) {
  try {
    const transports = await TransportService.getAllTransports();

    // Make sure all image URLs are properly formatted with full server URL
    const formattedTransports = transports.map(transport => {
      const transportJson = transport.toJSON();

      if (transportJson.image) {
        // Check if image URL is already a full URL
        if (!transportJson.image.startsWith('http')) {
          transportJson.image = `${SERVER_URL}${transportJson.image.startsWith('/') ? '' :
'/'}${transportJson.image}`;
        }

        // Debug: check if image file exists on server
        try {
          // Extract the filename from the path
          const imageFilename = transportJson.image.split('/').pop();
          // Construct the file path based on where the file should be
          const localPath = path.join(uploadsDir, imageFilename);
          const exists = fs.existsSync(localPath);
          console.log(`Image ${transportJson.image} exists on server: ${exists ? 'YES' : 'NO'},
local path: ${localPath}`);
        } catch (e) {
          console.error(`Error checking image existence: ${e.message}`);
        }
      }
      return transportJson;
    });

    res.status(200).json(formattedTransports);
  } catch (error: any) {
    res.status(400).json({ error: error.message });
  }
 }

 // Get transport by ID
 public static async getTransportById(req: Request, res: Response) {
  try {
    const transport = await TransportService.getTransportById(Number(req.params.id));
```

```
      if (transport) {
        // Ensure the image URL is properly formatted
        if (transport.image && !transport.image.startsWith('http')) {
          transport.image = `${SERVER_URL}${transport.image.startsWith('/') ? '' :
'/'}${transport.image}`;
        }

        res.status(200).json(transport);
      } else {
        res.status(404).json({ error: 'Transport not found' });
      }
    } catch (error: any) {
      res.status(400).json({ error: error.message });
    }
  }

  // Update Transport with image upload
  public static async updateTransport(req: Request, res: Response) {
    try {
      const existingTransport = await
TransportService.getTransportById(Number(req.params.id));

      if (!existingTransport) {
        return res.status(404).json({ error: 'Transport not found' });
      }

      upload.single('image')(req, res, async (err: any) => {
        if (err) {
          return res.status(400).json({ error: err.message });
        }

        try {
          const transportData: any = { ...req.body };

          // Handle image upload if a new file was provided
          if (req.file) {
            console.log(`New file uploaded for update: ${JSON.stringify(req.file)}`);

            // Verify the new file exists
            if (fs.existsSync(req.file.path)) {
              console.log(`Verified new file exists at: ${req.file.path}`);
              transportData.image = `${SERVER_URL}/uploads/${req.file.filename}`;
              console.log(`New image URL: ${transportData.image}`);

              // Try to delete the old image if it exists
              if (existingTransport.image) {
                try {
                  const oldFilename = existingTransport.image.split('/').pop();
                  if (oldFilename) {
                    const oldImagePath = path.join(uploadsDir, oldFilename);
                    console.log(`Trying to delete old image: ${oldImagePath}`);

                    if (fs.existsSync(oldImagePath)) {
                      fs.unlinkSync(oldImagePath);
                      console.log(`Successfully deleted old image: ${oldImagePath}`);
                    } else {
                      console.warn(`Old image file not found: ${oldImagePath}`);
```

```
            }
           }
          } catch (unlinkErr) {
           console.error('Error deleting old image file:', unlinkErr);
          }
         }
        } else {
         console.error(`WARNING: New file does not exist at: ${req.file.path}`);
         return res.status(500).json({ error: 'Failed to save uploaded image' });
        }
       }

       const updatedTransport = await
TransportService.updateTransport(Number(req.params.id), transportData);
       if (updatedTransport) {
        res.status(200).json(updatedTransport);
       } else {
        if (req.file && req.file.path) {
         fs.unlinkSync(req.file.path);
         console.log(`Deleted new file after update failure: ${req.file.path}`);
        }
        res.status(404).json({ error: 'Transport not found or update failed' });
       }
      } catch (error: any) {
       // Cleanup if error occurs during processing
       if (req.file && req.file.path) {
        try {
         fs.unlinkSync(req.file.path);
         console.log(`Deleted file after error: ${req.file.path}`);
        } catch (unlinkErr) {
         console.error('Failed to clean up file after error:', unlinkErr);
        }
       }
       res.status(400).json({ error: error.message });
      }
     });
    } catch (error: any) {
     res.status(400).json({ error: error.message });
    }
   }

   // Delete Transport
   public static async deleteTransport(req: Request, res: Response) {
    try {
     // First get the transport to check for image
     const transport = await TransportService.getTransportById(Number(req.params.id));

     if (!transport) {
      return res.status(404).json({ error: 'Transport not found' });
     }

     // Delete the image file if it exists
     if (transport.image) {
      try {
       const filename = transport.image.split('/').pop();
       if (filename) {
        const imagePath = path.join(uploadsDir, filename);
```

```
          console.log(`Attempting to delete image: ${imagePath}`);

          if (fs.existsSync(imagePath)) {
            fs.unlinkSync(imagePath);
            console.log(`Successfully deleted image: ${imagePath}`);
          } else {
            console.warn(`Image file not found for deletion: ${imagePath}`);
          }
        }
      } catch (unlinkErr) {
        console.error('Error deleting image file:', unlinkErr);
      }
    }

    const deleted = await TransportService.deleteTransport(Number(req.params.id));
    if (deleted) {
      res.status(200).json({ message: 'Transport deleted successfully' });
    } else {
      res.status(404).json({ error: 'Transport not found' });
    }
  } catch (error: any) {
    res.status(400).json({ error: error.message });
  }
 }

 // Helper method to ensure uploads directory exists
 private static ensureUploadsDir() {
  if (!fs.existsSync(uploadsDir)) {
    console.log(`TransportController: Creating uploads directory at ${uploadsDir}`);
    fs.mkdirSync(uploadsDir, { recursive: true, mode: 0o755 });
  }
 }
}

// Ensure uploads directory exists when controller is loaded
TransportController.ensureUploadsDir();

export default TransportController;
```

7. Backend Controller Room Booking

```
import { Request, Response } from 'express';
import RoomBookingService from '../services/RoomBookingService';

class RoomBookingController {
 public static async createBooking(req: Request, res: Response) {
  try {
    const booking = await RoomBookingService.createBooking({
      ...req.body,
      user_id: (req as any).user.id,
    });
    res.status(201).json(booking);
  } catch (error: any) {
    res.status(400).json({ error: error.message });
  }
```

```
  }

  public static async getAllBookings(req: Request, res: Response) {
    try {
      const bookings = await RoomBookingService.getAllBookings();
      res.status(200).json(bookings);
    } catch (error: any) {
      res.status(400).json({ error: error.message });
    }
  }

  public static async getBookingById(req: Request, res: Response) {
    try {
      const booking = await RoomBookingService.getBookingById(Number(req.params.id));
      if (booking) {
        res.status(200).json(booking);
      } else {
        res.status(404).json({ error: 'Booking not found' });
      }
    } catch (error: any) {
      res.status(400).json({ error: error.message });
    }
  }

  public static async updateBooking(req: Request, res: Response) {
    try {
      const bookingId = Number(req.params.id);
      const updateData = req.body;

      console.log(`Updating room booking ${bookingId} to status: ${updateData.status}`);
      const updatedBooking = await RoomBookingService.updateBooking(bookingId,
updateData);

      if (updatedBooking) {
        // Get the user's email
        let userEmail = await RoomBookingService.getUserEmailByBookingId(bookingId);

        // If email found, send notification
        if (userEmail) {
          const subject = `Room Booking Status: ${updateData.status.toUpperCase()}`;
          const body = `Dear User,\n\nYour booking for Room #${updatedBooking.room_id} on
${new Date(updatedBooking.booking_date).toLocaleDateString()} has been
${updateData.status.toLowerCase()}.\n\n${
            updateData.notes ? `Notes:\n${updateData.notes}\n\n` : ''
          }Thank you for using our booking system.`;

          console.log("Email parameters:", { to: userEmail, subject, bodyPreview:
body.substring(0, 50) + "..." });

          // Send email notification
          const emailResult = await RoomBookingService.sendEmail(userEmail, subject, body);
          console.log(`Email sending result: ${emailResult ? "SUCCESS" : "FAILED"}`);

          return res.status(200).json({
            ...updatedBooking.toJSON(),
            emailSent: emailResult,
            emailRecipient: userEmail
```

```
        });
      } else {
        console.warn(`No email address found for booking ${bookingId}`);
        return res.status(200).json({
          ...updatedBooking.toJSON(),
          emailSent: false,
          emailError: "No email address found"
        });
      }
    } else {
      return res.status(404).json({ error: "Booking not found" });
    }
  } catch (error) {
    console.error("Error updating booking:", error);
    return res.status(500).json({
      error: error.message || "Internal server error",
      stack: process.env.NODE_ENV === 'development' ? error.stack : undefined
    });
  }
 }

 public static async deleteBooking(req: Request, res: Response) {
  try {
    const deleted = await RoomBookingService.deleteBooking(Number(req.params.id));
    if (deleted) {
      res.status(200).json({ message: 'Booking deleted successfully' });
    } else {
      res.status(404).json({ error: 'Booking not found' });
    }
  } catch (error: any) {
    res.status(400).json({ error: error.message });
  }
 }
}

export default RoomBookingController;
```

8. Backend Controller Transport Booking

```
import { Request, Response } from 'express';
import TransportBookingService from '../services/TransportBookingService';

class TransportBookingController {
 public static async createBooking(req: Request, res: Response) {
  try {
    const booking = await TransportBookingService.createBooking({
      ...req.body,
      user_id: (req as any).user.id,
    });
    res.status(201).json(booking);
  } catch (error: any) {
    res.status(400).json({ error: error.message });
  }
 }
```

```
  public static async getAllBookings(req: Request, res: Response) {
    try {
      const bookings = await TransportBookingService.getAllBookings();
      res.status(200).json(bookings);
    } catch (error: any) {
      res.status(400).json({ error: error.message });
    }
  }

  public static async getBookingById(req: Request, res: Response) {
    try {
      const booking = await TransportBookingService.getBookingById(Number(req.params.id));
      if (booking) {
        res.status(200).json(booking);
      } else {
        res.status(404).json({ error: 'Booking not found' });
      }
    } catch (error: any) {
      res.status(400).json({ error: error.message });
    }
  }

  public static async updateBooking(req: Request, res: Response) {
    try {
      const bookingId = Number(req.params.id);
      const updateData = req.body;

      console.log(`Updating transport booking ${bookingId} to status: ${updateData.status}`);
      const updatedBooking = await TransportBookingService.updateBooking(bookingId,
updateData);

      if (updatedBooking) {
        let userEmail = await TransportBookingService.getUserEmailByBookingId(bookingId);

        if (userEmail) {
          const subject = `Transport Booking Status: ${updateData.status.toUpperCase()}`;
          const body = `Dear User,\n\nYour booking for Transport
#${updatedBooking.transport_id} on ${new
Date(updatedBooking.booking_date).toLocaleDateString()} has been
${updateData.status.toLowerCase()}.\n\n${
            updateData.notes ? `Notes:\n${updateData.notes}\n\n` : ''
          }Thank you for using our booking system.`;

          console.log("Email parameters:", { to: userEmail, subject, bodyPreview:
body.substring(0, 50) + "..." });

          const emailResult = await TransportBookingService.sendEmail(userEmail, subject,
body);
          console.log(`Email sending result: ${emailResult ? "SUCCESS" : "FAILED"}`);

          return res.status(200).json({
            ...updatedBooking.toJSON(),
            emailSent: emailResult,
            emailRecipient: userEmail
          });
        } else {
          console.warn(`No email address found for transport booking ${bookingId}`);
```

```
        return res.status(200).json({
          ...updatedBooking.toJSON(),
          emailSent: false,
          emailError: "No email address found"
        });
      }
    } else {
      return res.status(404).json({ error: "Booking not found" });
    }
  } catch (error) {
    console.error("Error updating booking:", error);
    return res.status(500).json({
      error: error.message || "Internal server error",
      stack: process.env.NODE_ENV === 'development' ? error.stack : undefined
    });
  }
 }

 public static async deleteBooking(req: Request, res: Response) {
  try {
    const deleted = await TransportBookingService.deleteBooking(Number(req.params.id));
    if (deleted) {
      res.status(200).json({ message: 'Booking deleted successfully' });
    } else {
      res.status(404).json({ error: 'Booking not found' });
    }
  } catch (error: any) {
    res.status(400).json({ error: error.message });
  }
 }
}

export default TransportBookingController;
```

9. Backend Middleware User Auth

```
// src/middlewares/authMiddleware.ts
import { Request, Response, NextFunction } from 'express';
import { verifyToken } from '../utils/token';
import User from '../models/User';

const authMiddleware = async (req: Request, res: Response, next: NextFunction) => {
  const authHeader = req.headers.authorization;
  if (!authHeader) {
    return res.status(401).json({ error: 'Authorization header missing' });
  }

  const token = authHeader.split(' ')[1];
  if (!token) {
    return res.status(401).json({ error: 'Token missing' });
  }

  try {
    const decoded: any = verifyToken(token);
    const user = await User.findByPk(decoded.id);
```

```
    if (!user) {
      return res.status(401).json({ error: 'Invalid token' });
    }

    (req as any).user = user;
    next();
  } catch (error) {
    return res.status(401).json({ error: 'Invalid token' });
  }
};

export default authMiddleware;
```

10. Backend Middleware Admin Auth

```
// src/middlewares/adminAuthMiddleware.ts
import { Request, Response, NextFunction } from 'express';
import jwt from 'jsonwebtoken';
import dotenv from 'dotenv';

dotenv.config();

// Extend the Request interface to include admin_id
declare global {
  namespace Express {
    interface Request {
      admin_id?: number;
    }
  }
}

export const adminAuthMiddleware = (req: Request, res: Response, next: NextFunction) => {
  // Get token from header
  const authHeader = req.headers.authorization;
  const token = authHeader && authHeader.split(' ')[1]; // Bearer TOKEN format

  // Check if token exists
  if (!token) {
    return res.status(401).json({ message: 'Tidak ada token, otorisasi ditolak.' });
  }

  try {
    // Verify token
    const decoded = jwt.verify(token, process.env.JWT_SECRET as string) as { id: number, role:
string };

    // Check if user role is admin
    if (decoded.role !== 'admin') {
      return res.status(403).json({ message: 'Akses ditolak. Hanya admin yang diizinkan.' });
    }

    // Add admin_id to request object
    req.admin_id = decoded.id;

    // Log for debugging
```

```
    console.log(`Authenticated admin with ID: ${decoded.id}`);

    next();
  } catch (error) {
    console.error('Token verification failed:', error);
    return res.status(401).json({ message: 'Token tidak valid.' });
  }
};

// This middleware is specifically for the profile editing endpoint
export const profileAuthMiddleware = (req: Request, res: Response, next: NextFunction) => {
  // Get token from header
  const authHeader = req.headers.authorization;
  const token = authHeader && authHeader.split(' ')[1]; // Bearer TOKEN format

  // Check if token exists
  if (!token) {
    return res.status(401).json({ message: 'Tidak ada token, otorisasi ditolak.' });
  }

  try {
    // Verify token
    const decoded = jwt.verify(token, process.env.JWT_SECRET as string) as { id: number, role:
string };

    // Check if user role is admin
    if (decoded.role !== 'admin') {
      return res.status(403).json({ message: 'Akses ditolak. Hanya admin yang diizinkan.' });
    }

    // Add admin_id to request object
    req.admin_id = decoded.id;

    // Log for debugging
    console.log(`Admin profile access with ID: ${decoded.id}`);

    next();
  } catch (error) {
    console.error('Token verification failed:', error);
    return res.status(401).json({ message: 'Token tidak valid.' });
  }
};
```

11. Backend Route Admin auth

```
// src/routes/adminAuthRoutes.ts
import express from 'express';
import { adminLogin, adminRegister, editProfile } from '../controllers/adminAuthController';
import { body } from 'express-validator';
import rateLimit from 'express-rate-limit';
import { validate } from '../middlewares/validateMiddleware';
import { profileAuthMiddleware } from '../middlewares/adminAuthMiddleware';

const router = express.Router();
```

```
// Rate limiter untuk login Admin
const loginLimiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 menit
  max: 5, // Maksimal 5 permintaan per IP
  message: 'Terlalu banyak percobaan login dari IP ini. Silakan coba lagi setelah 15 menit.',
  standardHeaders: true,
  legacyHeaders: false,
});

// Validasi dan rate limiter untuk registrasi Admin
router.post(
  '/register',
  [
    body('username').notEmpty().withMessage('Username tidak boleh kosong.'),
    body('email').isEmail().withMessage('Email tidak valid.'),
    body('password').isLength({ min: 6 }).withMessage('Password harus minimal 6 karakter.'),
  ],
  validate,
  adminRegister
);

// Validasi dan rate limiter untuk login Admin
router.post(
  '/login',
  loginLimiter,
  [
    body('email').isEmail().withMessage('Email tidak valid.'),
    body('password').notEmpty().withMessage('Password tidak boleh kosong.'),
  ],
  validate,
  adminLogin
);

// UPDATED: Apply middleware to ensure admin_id is available for the profile update
router.put(
  '/profile',
  profileAuthMiddleware, // Use the dedicated middleware for profile
  [
    body('username').optional().notEmpty().withMessage('Username tidak boleh kosong.'),
    body('email').optional().isEmail().withMessage('Email tidak valid.'),
    body('password').optional().isLength({ min: 6 }).withMessage('Password harus minimal 6
karakter.'),
  ],
  validate,
  editProfile
);

export default router;
```

12. Backend Route User Auth

```
import { Router } from 'express';
import AuthController from '../controllers/authController';
import authMiddleware from '../middlewares/authMiddleware';
```

```
const router = Router();

// Rute publik
router.post('/register', AuthController.register);
router.post('/login', AuthController.login);

// Rute yang dilindungi (contoh)
router.get('/profile', authMiddleware, (req, res) => {
  // Akses user yang telah terotentikasi
  res.status(200).json({ message: 'Akses profil berhasil', user: (req as any).user });
});

// Rute untuk mengganti password
router.post('/change-password', authMiddleware, AuthController.changePassword);
router.put('/edit-profile', authMiddleware, AuthController.editProfile);


export default router;
```

13. Backend Route User

```
// backend/src/routes/userRoutes.ts
import { Router } from 'express';
import UserController from '../controllers/userController';
import authMiddleware from '../middlewares/authMiddleware';
import adminMiddleware from '../middlewares/adminMiddleware';
import multer from 'multer';
import fs from 'fs';
import path from 'path';

const router = Router();

// Memastikan direktori uploads ada
const uploadDir = path.join(__dirname, '../../uploads');
if (!fs.existsSync(uploadDir)) {
  fs.mkdirSync(uploadDir, { recursive: true });
}

// Konfigurasi multer untuk penanganan file Excel
const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, uploadDir); // Menggunakan direktori yang sudah dipastikan ada
  },
  filename: function (req, file, cb) {
    cb(null, `${Date.now()}-${file.originalname}`);
  }
});

// Validasi file Excel yang diupload
const fileFilter = (req: any, file: Express.Multer.File, cb: multer.FileFilterCallback) => {
  const fileExtension = path.extname(file.originalname).toLowerCase();
  if (['.xlsx', '.xls'].includes(fileExtension)) {
    cb(null, true);
  } else {
    cb(new Error('Format file harus .xlsx atau .xls'));
```

```
  }
};

const upload = multer({
  storage: storage,
  fileFilter: fileFilter,
  limits: {
    fileSize: 5 * 1024 * 1024 // Batas ukuran file 5MB
  }
});

router.get('/profile', authMiddleware, UserController.getProfile);
router.put('/profile', authMiddleware, UserController.updateProfile);
router.get('/', authMiddleware, adminMiddleware, UserController.getAll);
router.get('/:id', authMiddleware, adminMiddleware, UserController.getById);
router.post('/', authMiddleware, adminMiddleware, UserController.create);
router.put('/:id', authMiddleware, UserController.update); // With permission check inside
router.delete('/:id', authMiddleware, adminMiddleware, UserController.delete);

// Import user dari Excel - hanya admin
router.post('/import',
  authMiddleware,
  adminMiddleware,
  upload.single('excelFile'),
  UserController.importFromExcel
);

export default router;
```

14. Backend Route Admin

```
// src/routes/adminRoutes.ts
import express from 'express';
import { getAllAdmins, getAdminById, updateAdmin, deleteAdmin } from
'../controllers/adminController';
import { body } from 'express-validator';
import { validate } from '../middlewares/validateMiddleware';

const router = express.Router();

/**
 * @route GET /api/admins
 * @desc Get all admins
 * @access Protected (Admin only)
 */
router.get('/', getAllAdmins);

/**
 * @route GET /api/admins/:admin_id
 * @desc Get admin by ID
 * @access Protected (Admin only)
 */
router.get('/:admin_id', getAdminById);

/**
```

```
 * @route PUT /api/admins/:admin_id
 * @desc Update admin by ID
 * @access Protected (Admin only)
 */
router.put(
  '/:admin_id',
  [
    body('username').optional().notEmpty().withMessage('Username tidak boleh kosong.'),
    body('email').optional().isEmail().withMessage('Email tidak valid.'),
    body('password').optional().isLength({ min: 6 }).withMessage('Password harus minimal 6
karakter.'),
  ],
  validate,
  updateAdmin
);
/**
 * @route DELETE /api/admins/:admin_id
 * @desc Delete admin by ID
 * @access Protected (Admin only)
 */
router.delete('/:admin_id', deleteAdmin);

export default router;
```

15. Backend Route Room

```
// src/routes/roomRoutes.ts
import { Router } from 'express';
import RoomController from '../controllers/RoomController';
import authMiddleware from '../middlewares/authMiddleware';
import adminMiddleware from '../middlewares/adminMiddleware';

const router = Router();

router.post('/', authMiddleware, adminMiddleware, RoomController.createRoom);
router.get('/', authMiddleware, RoomController.getAllRooms);
router.get('/:id', authMiddleware, RoomController.getRoomById);
router.put('/:id', authMiddleware, adminMiddleware, RoomController.updateRoom);
router.delete('/:id', authMiddleware, adminMiddleware, RoomController.deleteRoom);

export default router;
```

16. Backend Route Transport

```
// src/routes/transportRoutes.ts
import { Router } from 'express';
import authMiddleware from '../middlewares/authMiddleware';
import adminMiddleware from '../middlewares/adminMiddleware';
import TransportController from '../controllers/transportController';

const router = Router();

router.post('/', authMiddleware, adminMiddleware, TransportController.createTransport);
router.get('/', authMiddleware, TransportController.getAllTransports);
```

```
router.get('/:id', authMiddleware, TransportController.getTransportById);
router.put('/:id', authMiddleware, adminMiddleware, TransportController.updateTransport);
router.delete('/:id', authMiddleware, adminMiddleware, TransportController.deleteTransport);

export default router;
```

17. Backend Route Room Booking

```
// src/routes/roomBookingRoutes.ts
import { Router } from 'express';
import combinedAuthMiddleware from '../middlewares/combinedAuthMiddleware';
import adminMiddleware from '../middlewares/adminMiddleware';
import RoomBookingController from '../controllers/roomBookingController';

const router = Router();
router.post('/', combinedAuthMiddleware, RoomBookingController.createBooking);
router.get('/', combinedAuthMiddleware, RoomBookingController.getAllBookings);
router.get('/:id', combinedAuthMiddleware, RoomBookingController.getBookingById);
router.put('/:id', combinedAuthMiddleware, RoomBookingController.updateBooking);
router.delete('/:id', adminMiddleware, RoomBookingController.deleteBooking);
export default router;
```

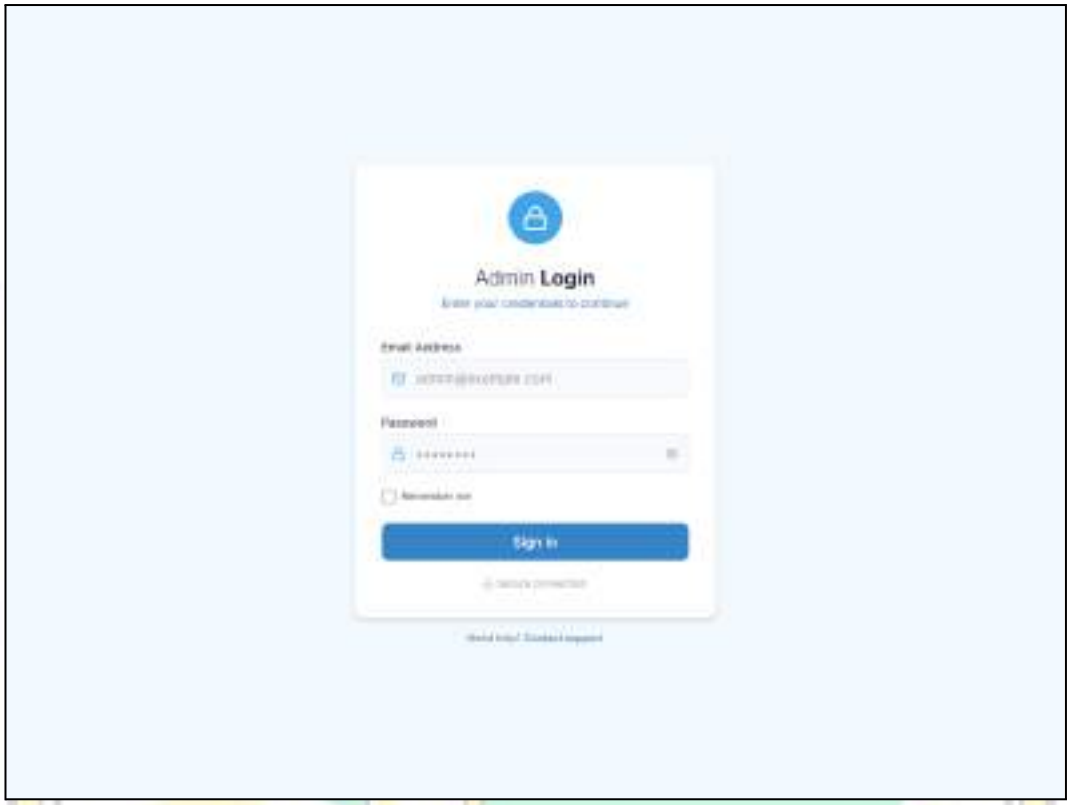
18. Backend Route Transport Booking

```
import { Router } from 'express';
import authMiddleware from '../middlewares/authMiddleware';
import adminMiddleware from '../middlewares/adminMiddleware';
import TransportBookingController from '../controllers/transportBookingController';

const router = Router();

router.post('/', authMiddleware, TransportBookingController.createBooking);
router.get('/', authMiddleware, TransportBookingController.getAllBookings);
router.get('/:id', authMiddleware, TransportBookingController.getBookingById);
router.put('/:id', authMiddleware, TransportBookingController.updateBooking);
router.delete('/:id', authMiddleware, adminMiddleware,
TransportBookingController.deleteBooking);

export default router;
```

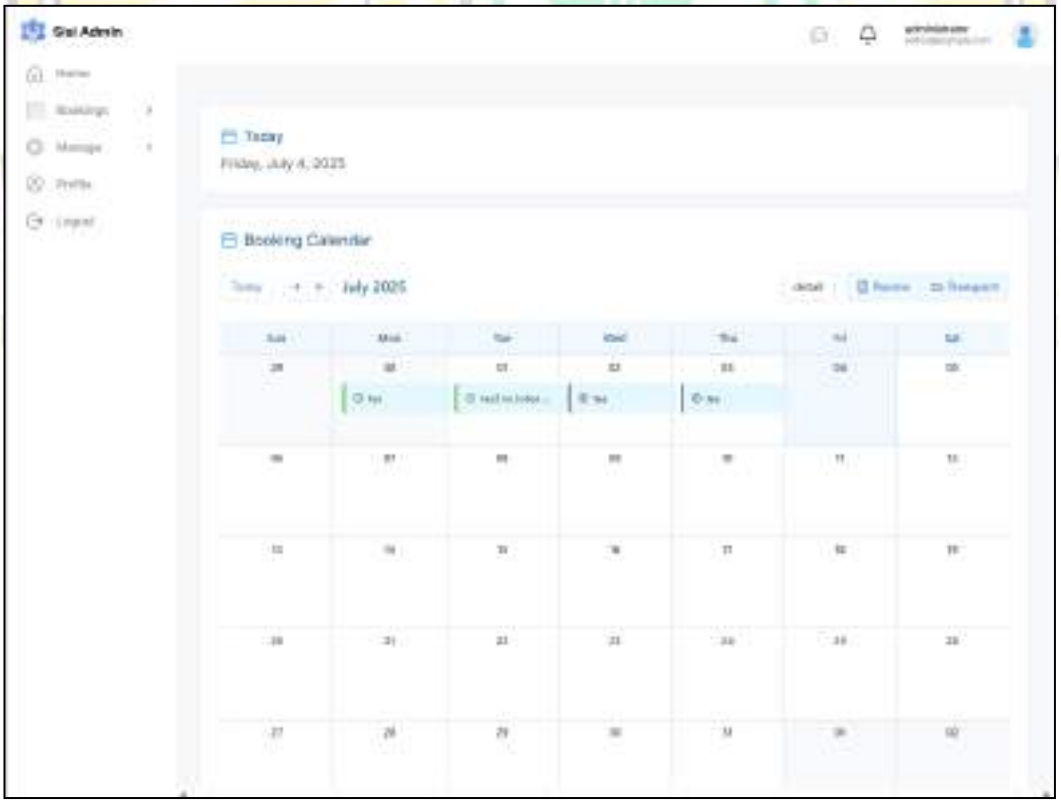
LAMPIRAN E

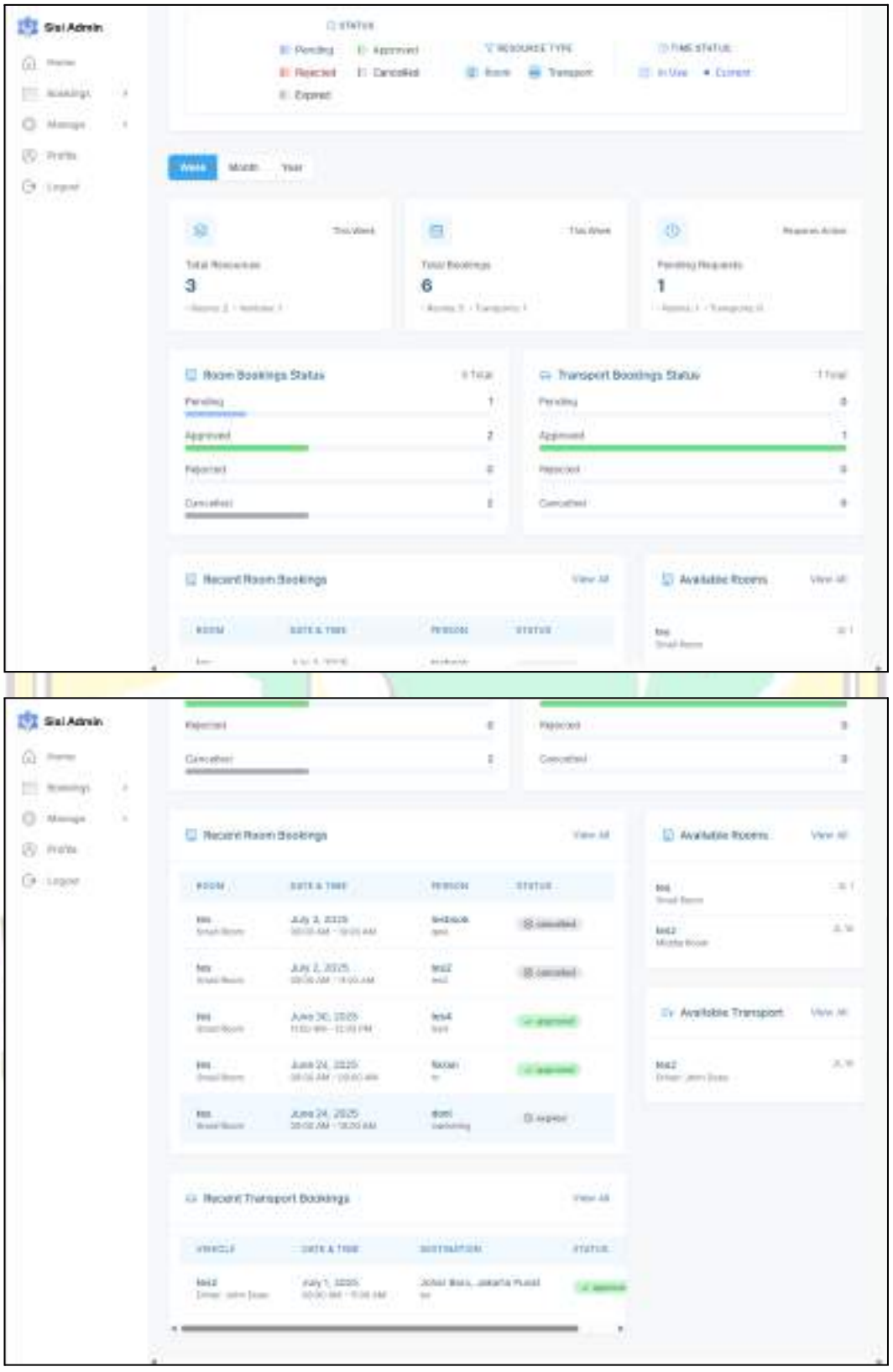
IMPLEMENTASI ANTARMUKA (LANJUTAN)

1. Rancangan User Interface Halaman Login Back Office

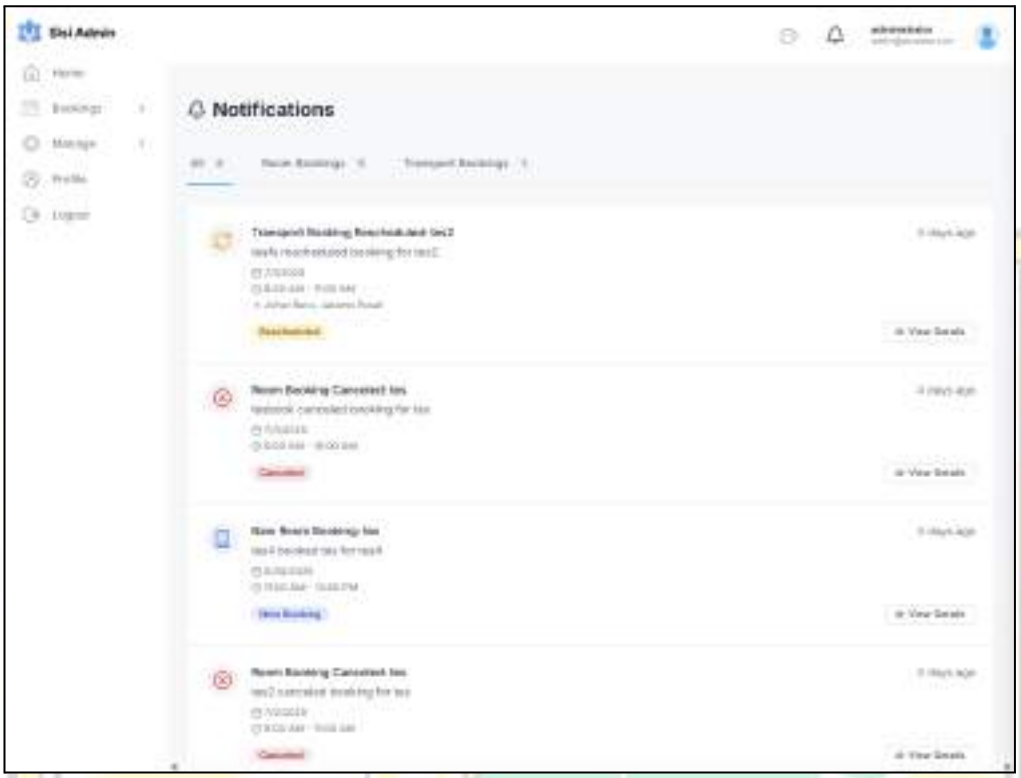


2. Rancangan User Interface Halaman Dashboard

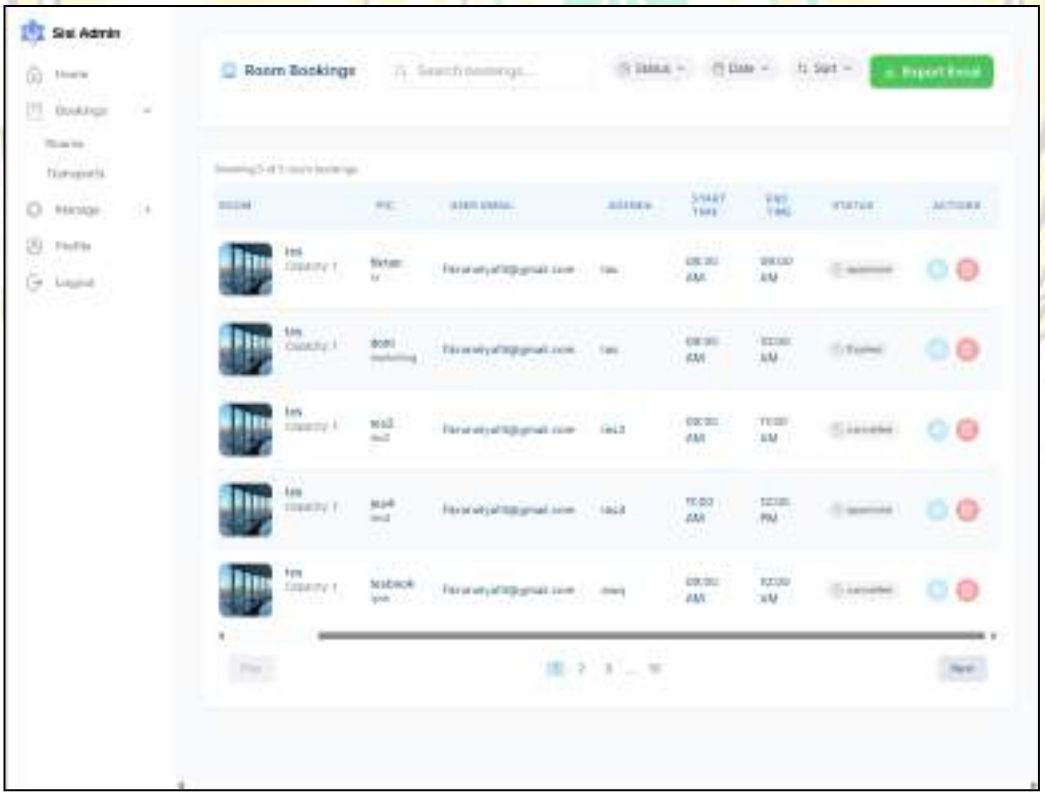



Sisi Admin
Week
Month
Year
3
6
1
Room Bookings Status
Transport Bookings Status
Recent Room Bookings
View All
Available Rooms
Available Transport
Recent Transport Bookings

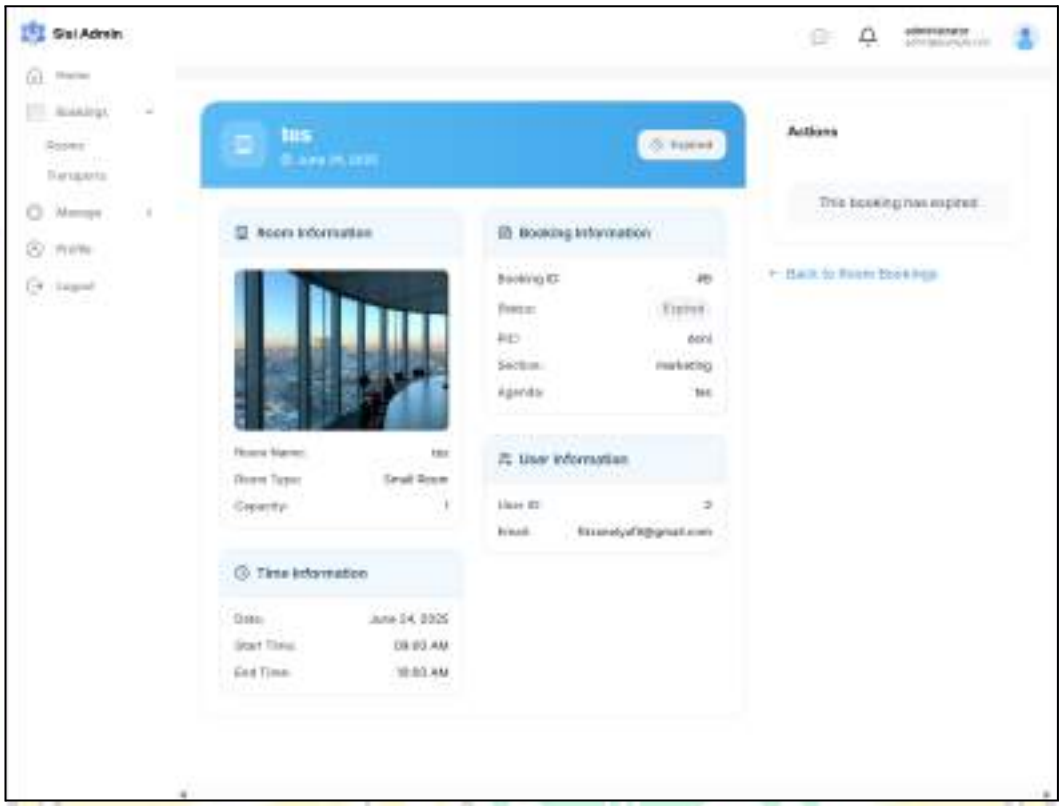
3. Rancangan User Interface Halaman Notifikasi



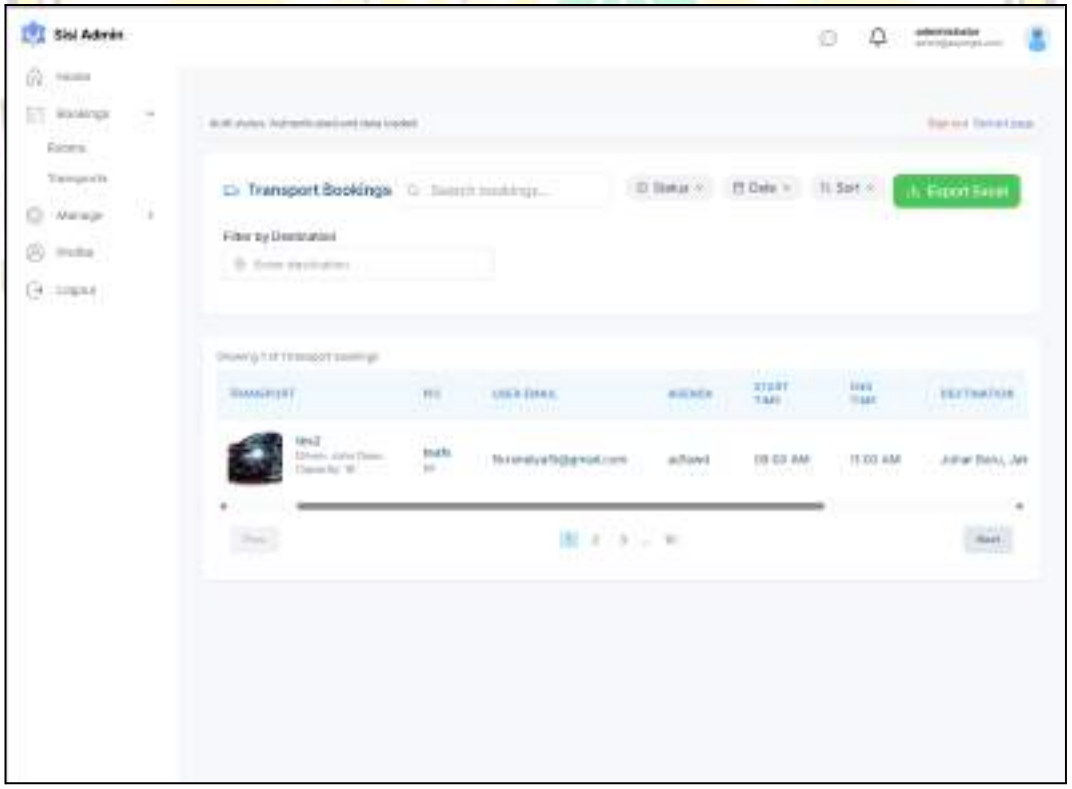
4. Rancangan User Interface Halaman List Booking Meeting Room



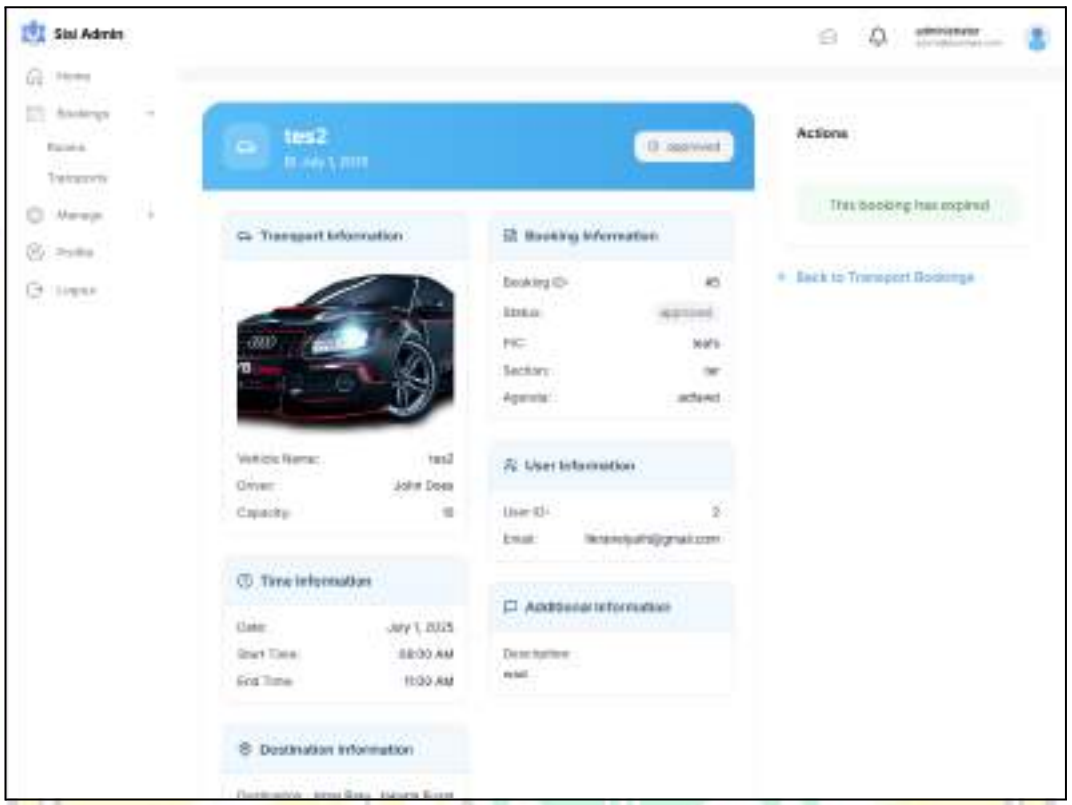
5. Rancangan User Interface Halaman Detail Booking Meeting Room



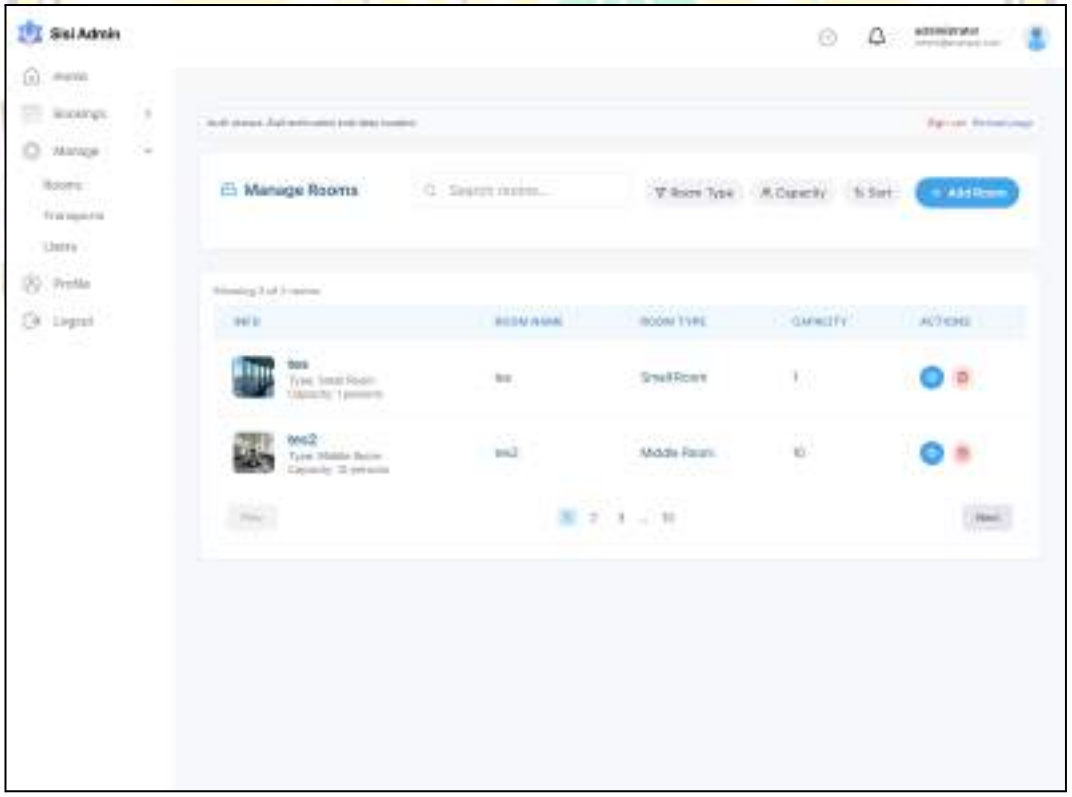
6. Rancangan User Interface Halaman List Booking Transportasi

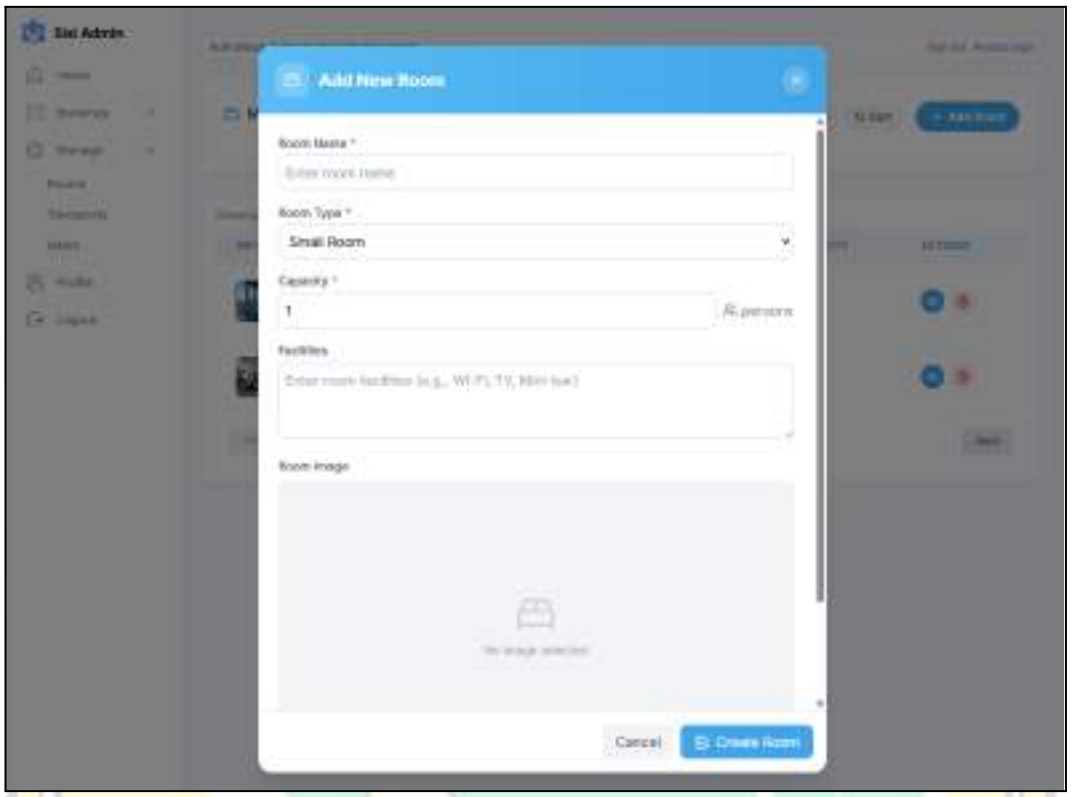


7. Rancangan User Interface Halaman Detail Booking Transportasi

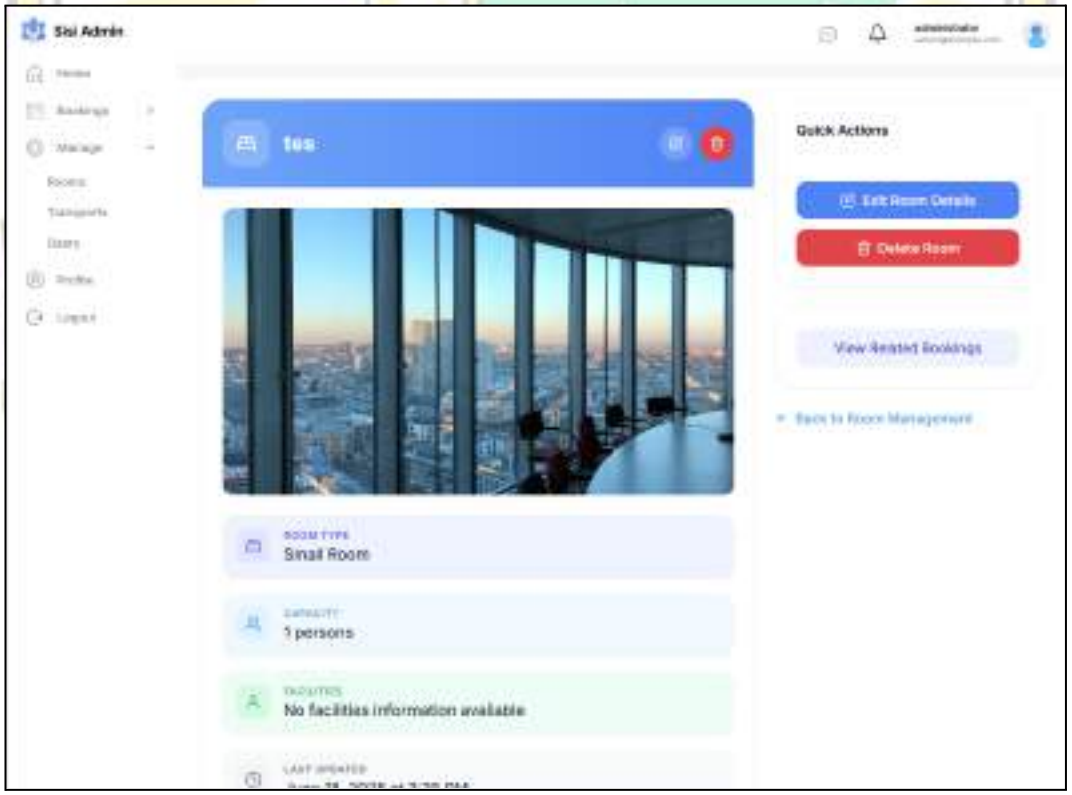


8. Rancangan User Interface Halaman List Room

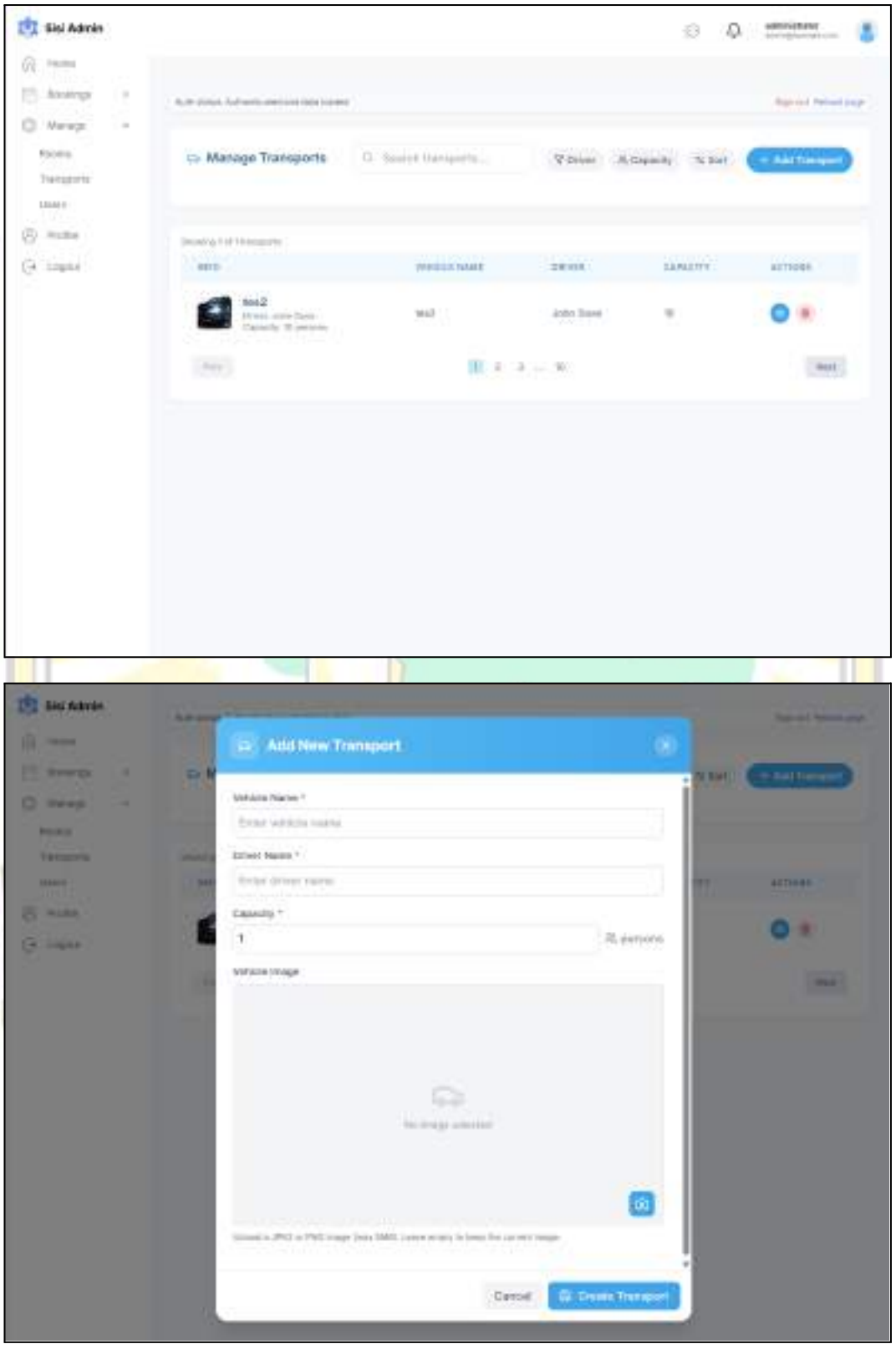




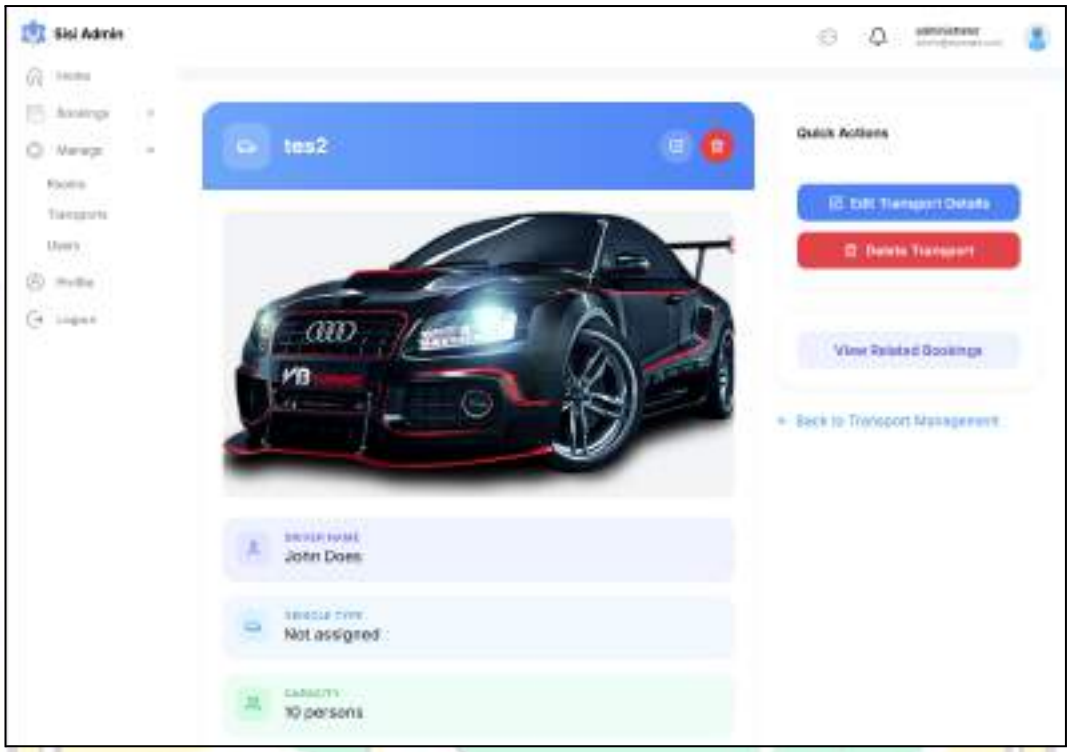
9. Rancangan User Interface Halaman Detail Room



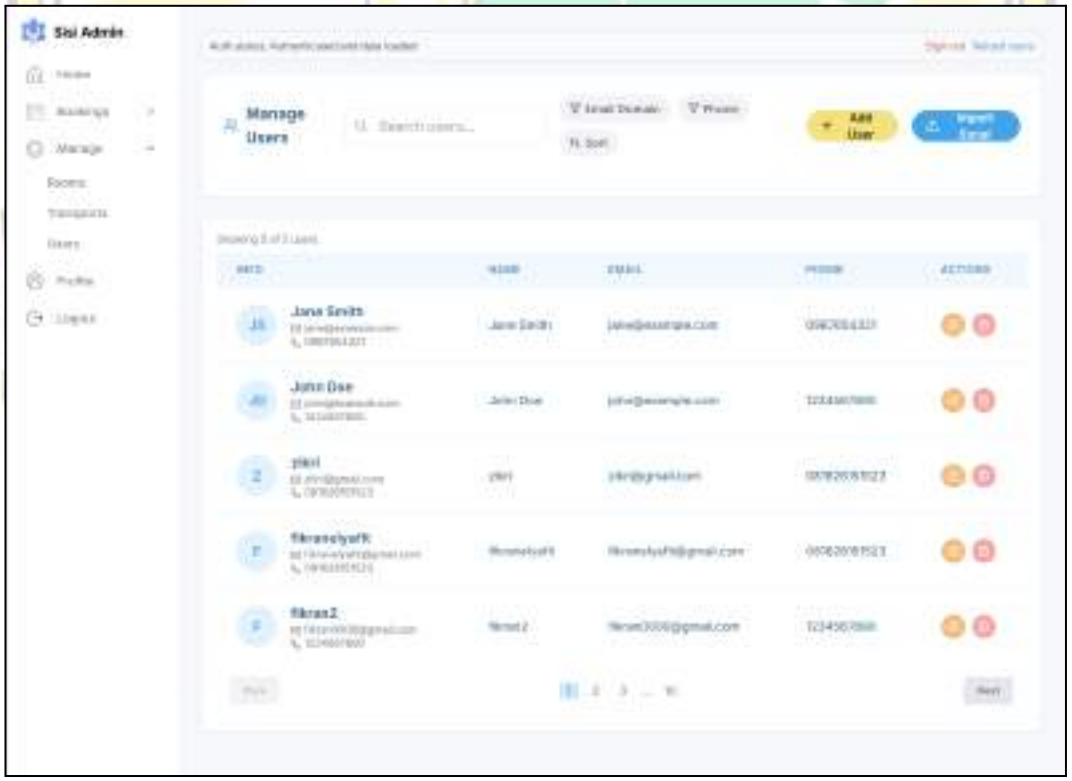
10. Rancangan User Interface Halaman List Transportasi

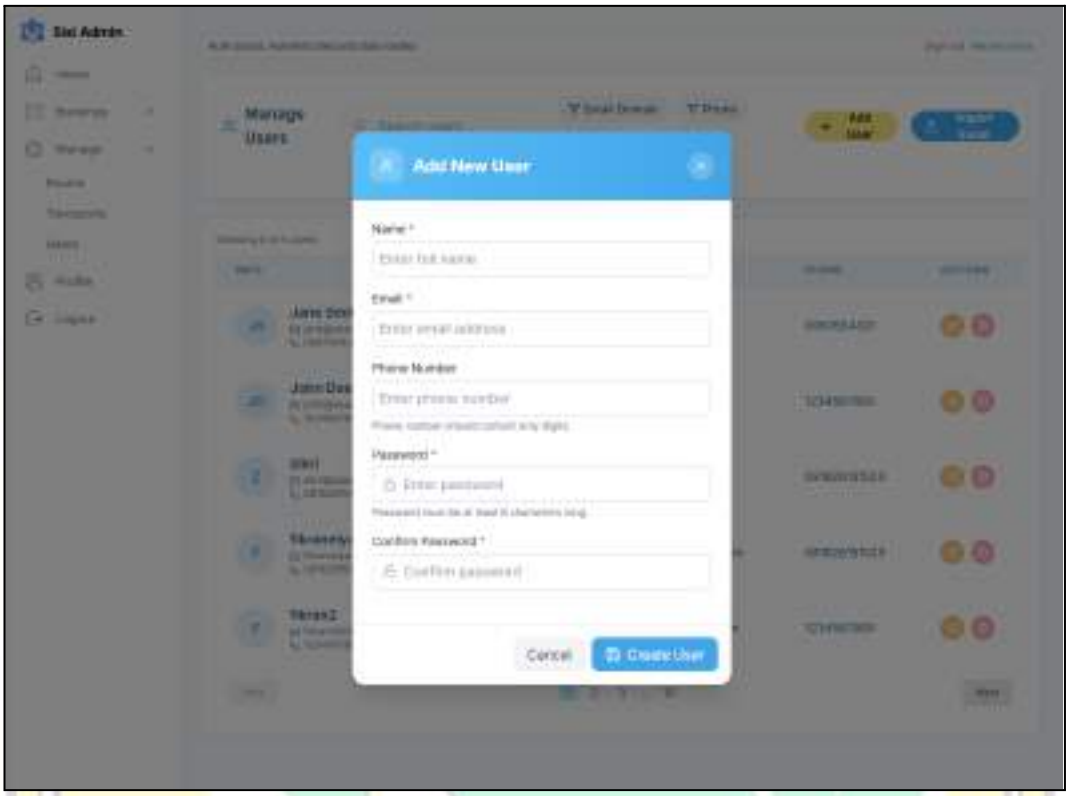


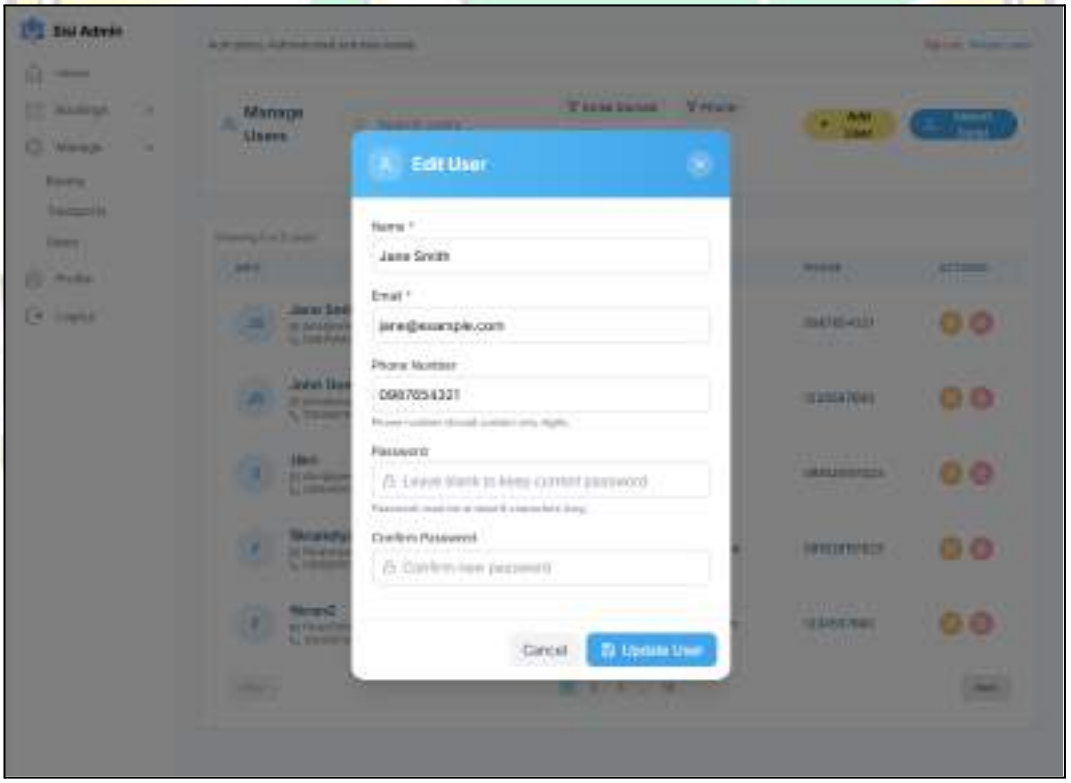
11. Rancangan User Interface Halaman Detail Transportasi

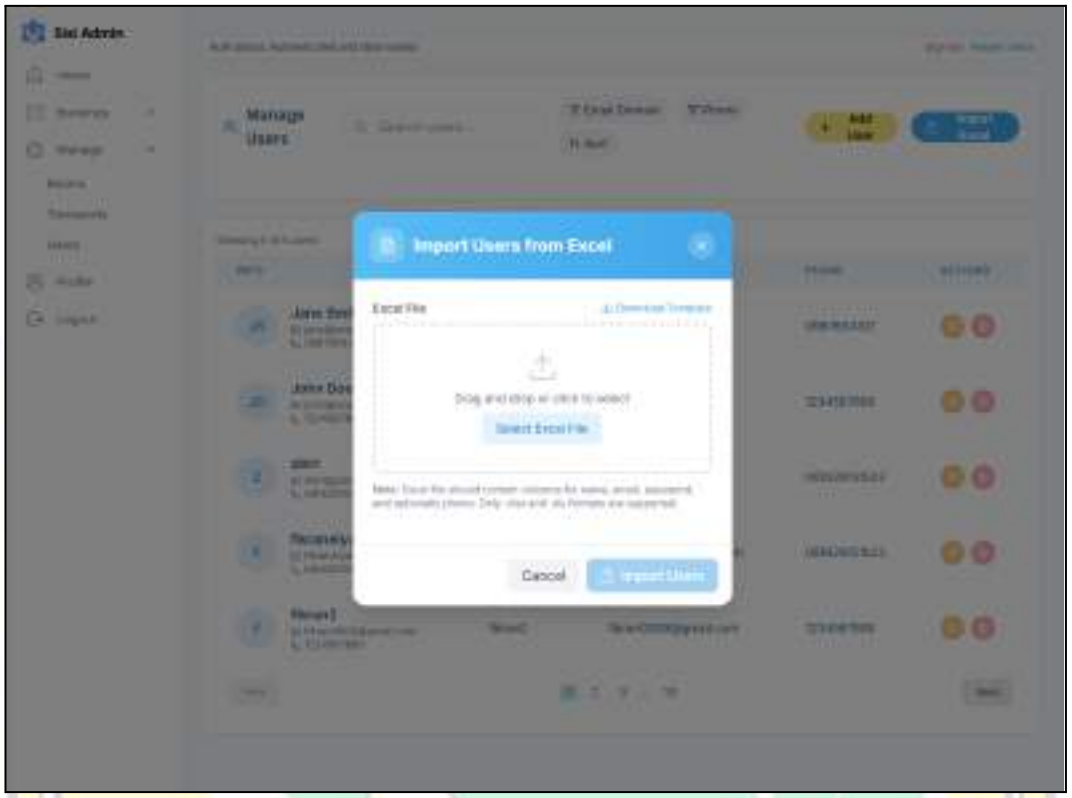


12. Rancangan User Interface Halaman List User

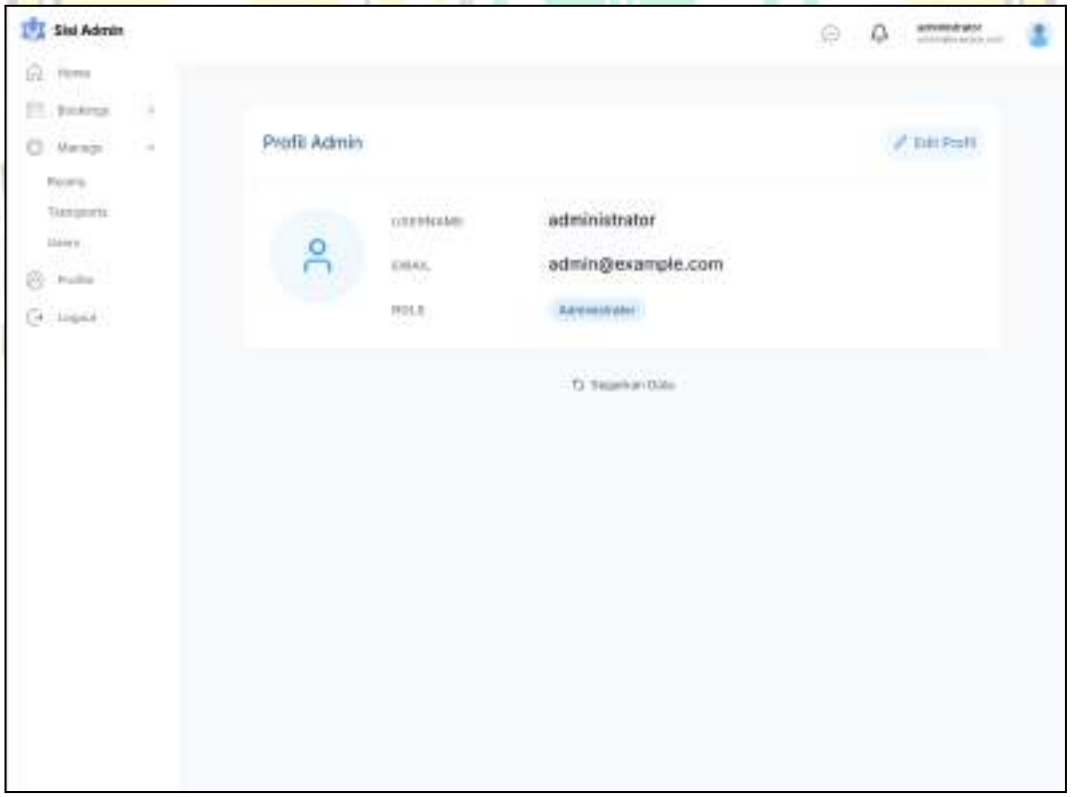



Add New User
Name *
Email *
Phone Number
Password *
Confirm Password *
Cancel
Create User

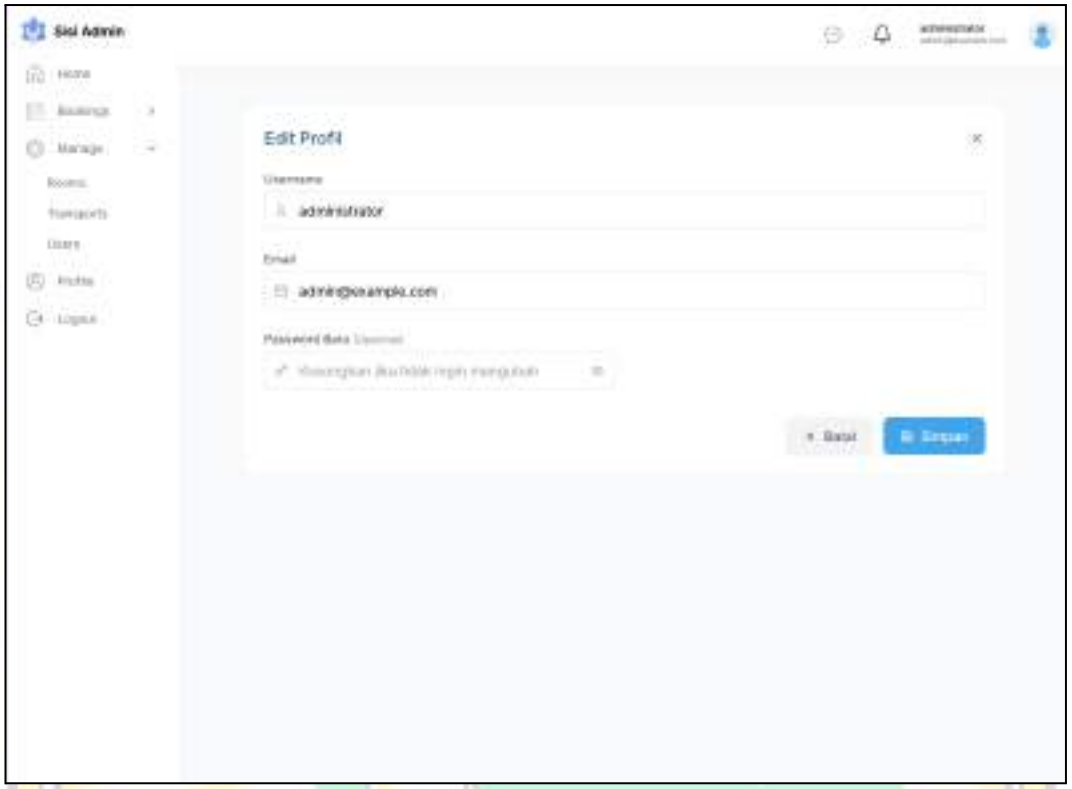

Edit User
Name *
Jane Smith
Email *
jane@example.com
Phone Number
0987654321
Password
Confirm Password
Cancel
Update User



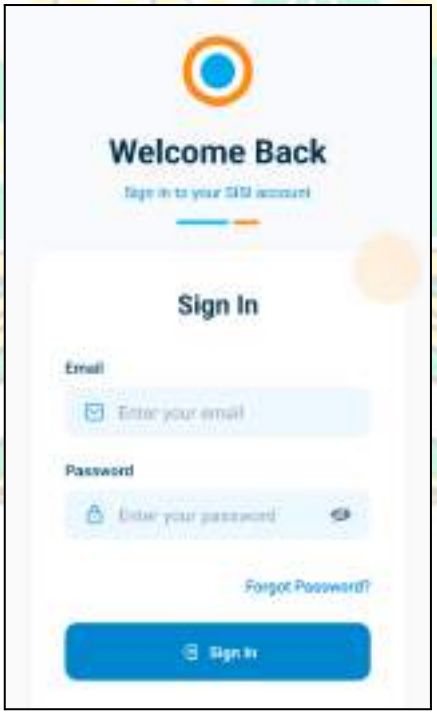
13. Rancangan User Interface Halaman Profile



14. Rancangan User Interface Halaman Edit Profile



15. Rancangan User Interface Halaman Login Aplikasi Mobile

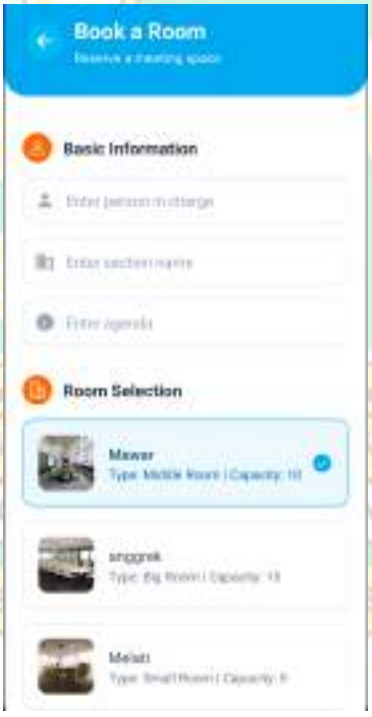


16. Rancangan User Interface Halaman Home

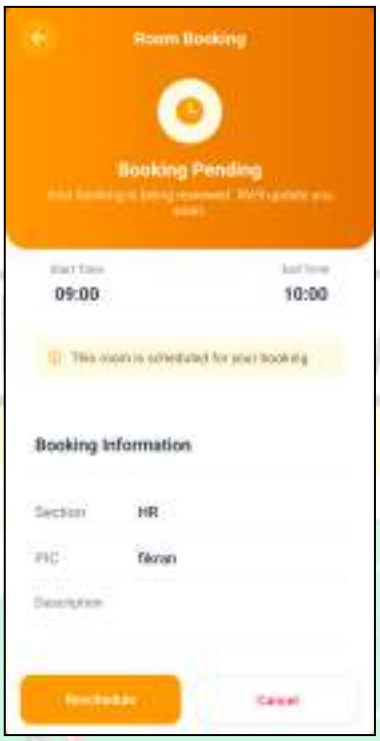
17. Rancangan User Interface Halaman Explore

18. Rancangan User Interface Halaman Detail Explore

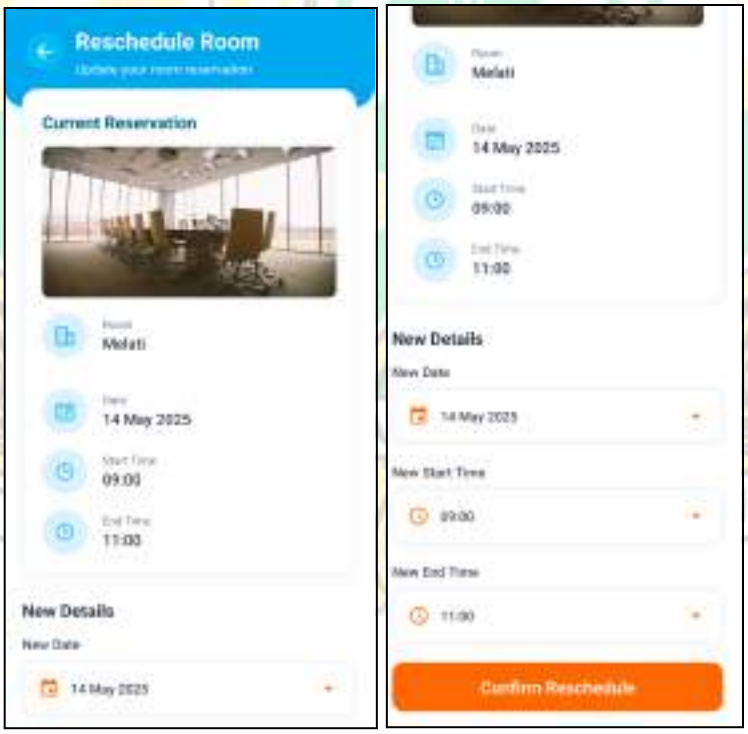
19. Rancangan User Interface Halaman Booking



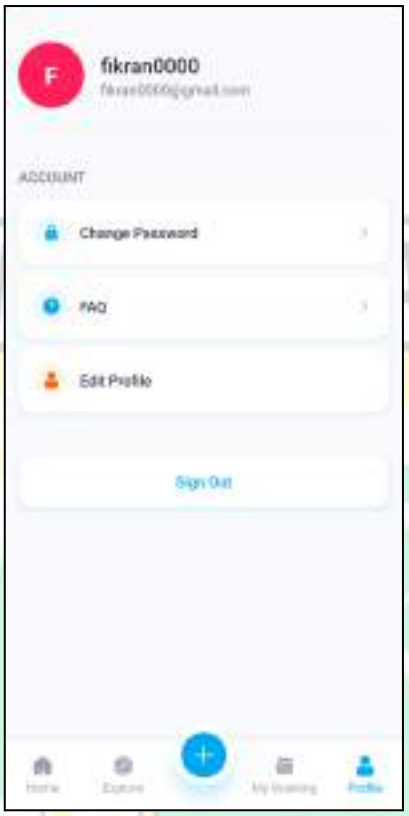
20. Rancangan User Interface Halaman Detail Booking



21. Rancangan User Interface Halaman Reschedule



22. Rancangan User Interface Halaman Profile

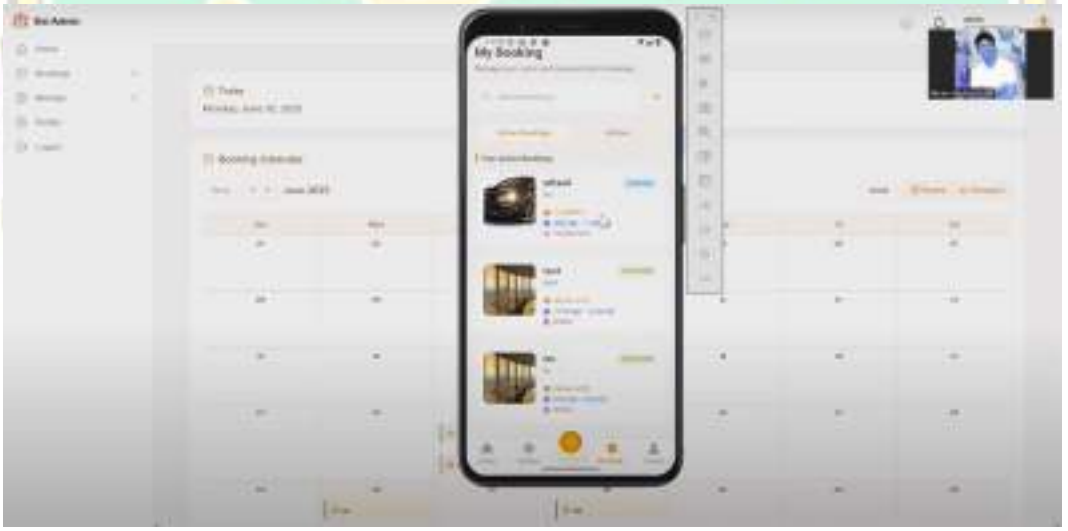


LAMPIRAN F

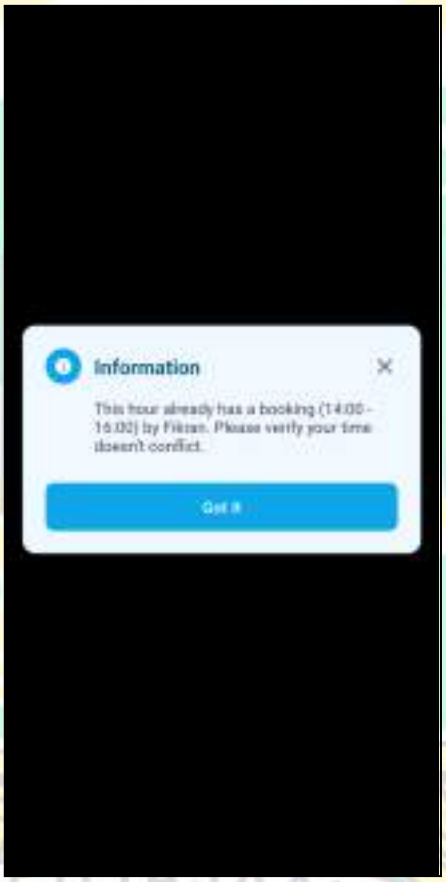
PENGUJIAN SISTEM (LANJUTAN)

1. Pengujian Pada *Booking* Ruang Rapat dan Transportasi

Tipe Alur Pengujian	Berhasil
Hasil Yang Diharapkan	Pengguna berhasil melakukan *booking* ruang rapat dan transportasi melalui aplikasi
Pengamatan	Sistem merekam data *booking* yang akan di konfirmasi oleh admin
Hasil Akhir	Sesuai

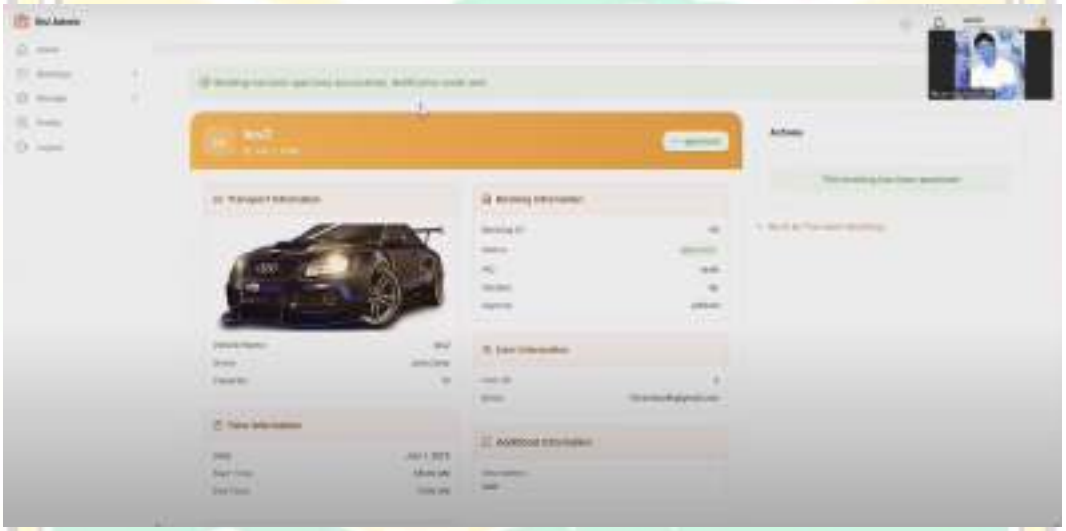


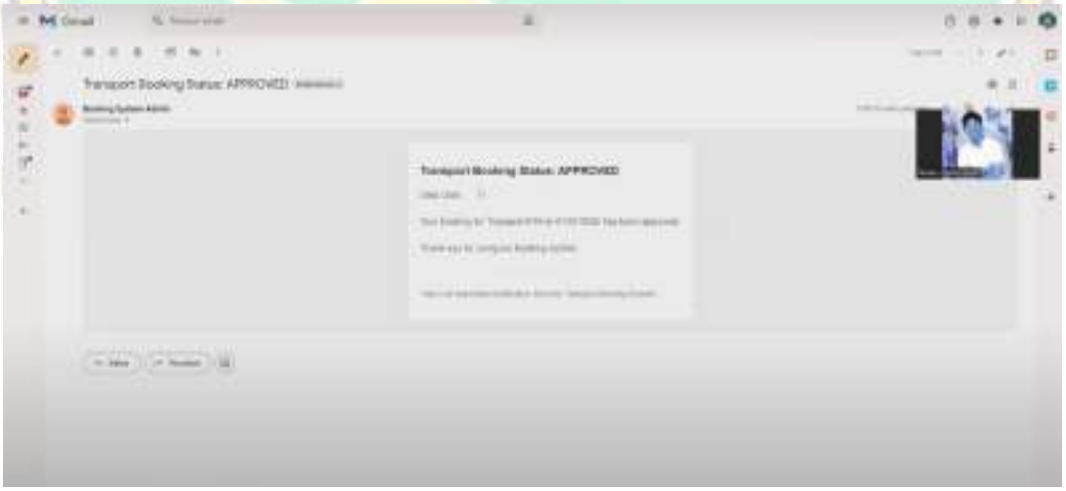
Tipe Alur Pengujian	Alternatif
Hasil Yang Diharapkan	Pengguna tidak bisa melakukan booking jika jadwal bentrok dengan peminjam lain
Pengamatan	Sistem mengembalikan pesan kesalahan kepada pengguna bahwa tanggal dan waktu yang dipilih bentrok dengan peminjam lain
Hasil Akhir	Sesuai



2. Pengujian Pada Penerimaan dan Penolakan *Booking*

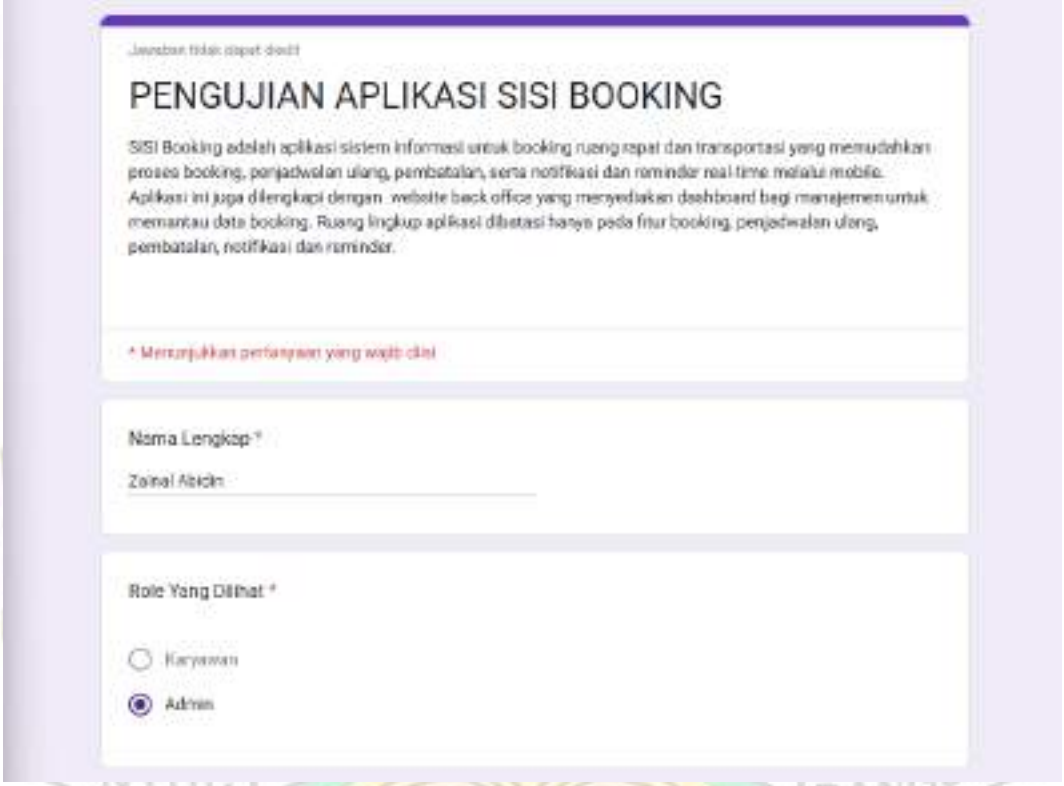
Tipe Alur Pengujian	Berhasil
Hasil Yang Diharapkan	Admin berhasil melakukan perubahan status *approve* pada data booking
Pengamatan	Sistem merekam data booking yang sudah dikonfirmasi oleh admin dan email konfirmasi terikirim ke pengguna
Hasil Akhir	Sesuai

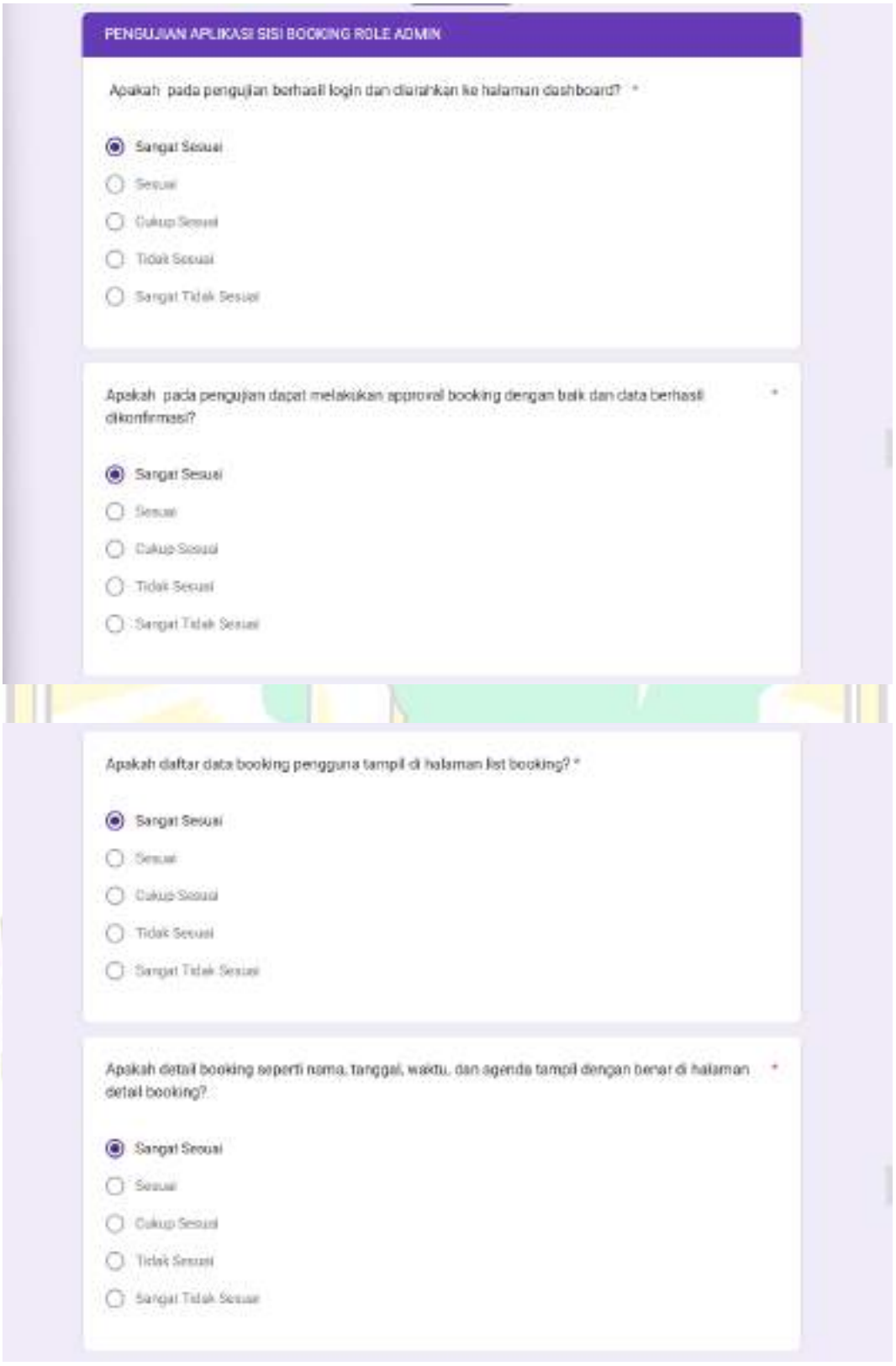


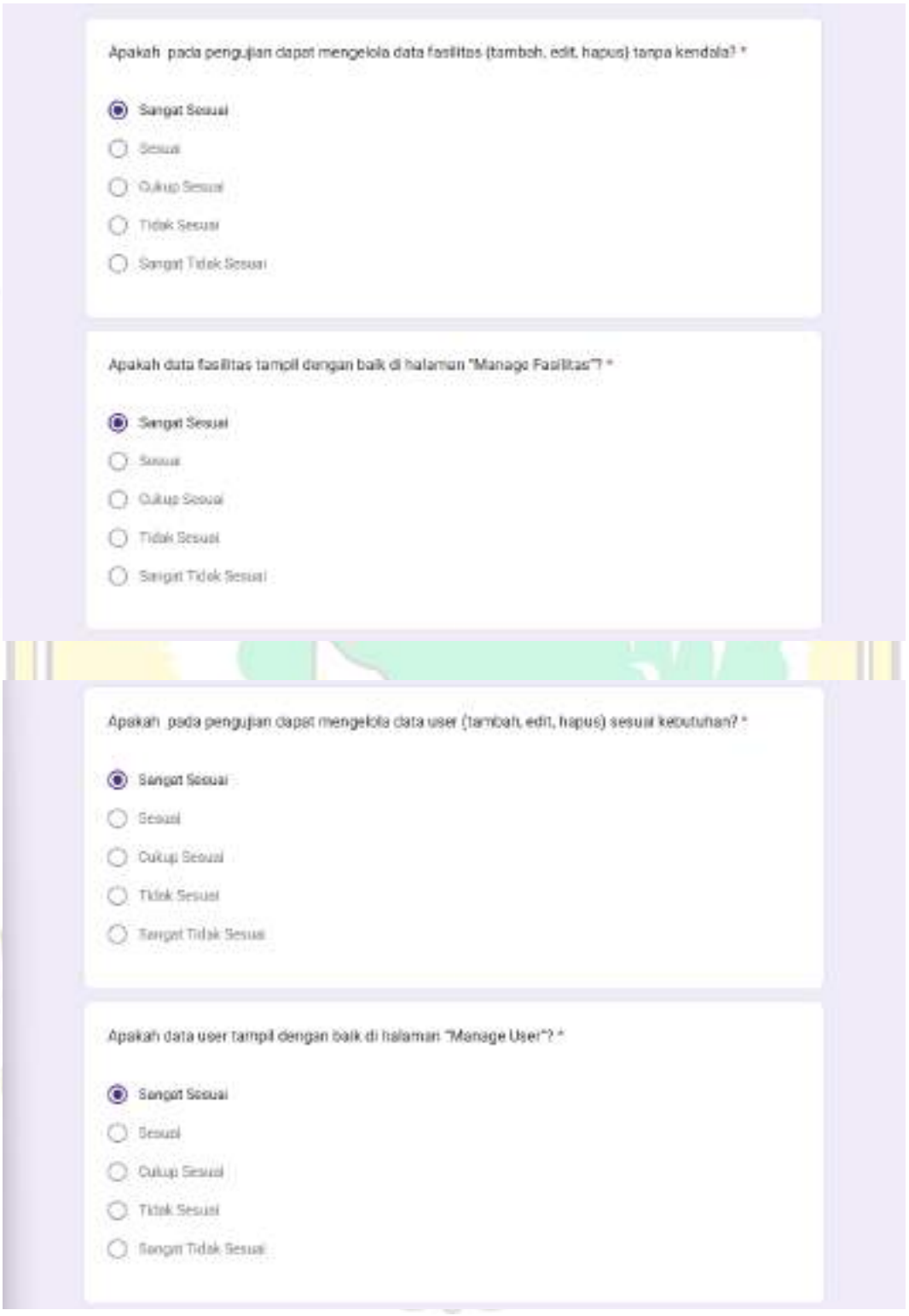


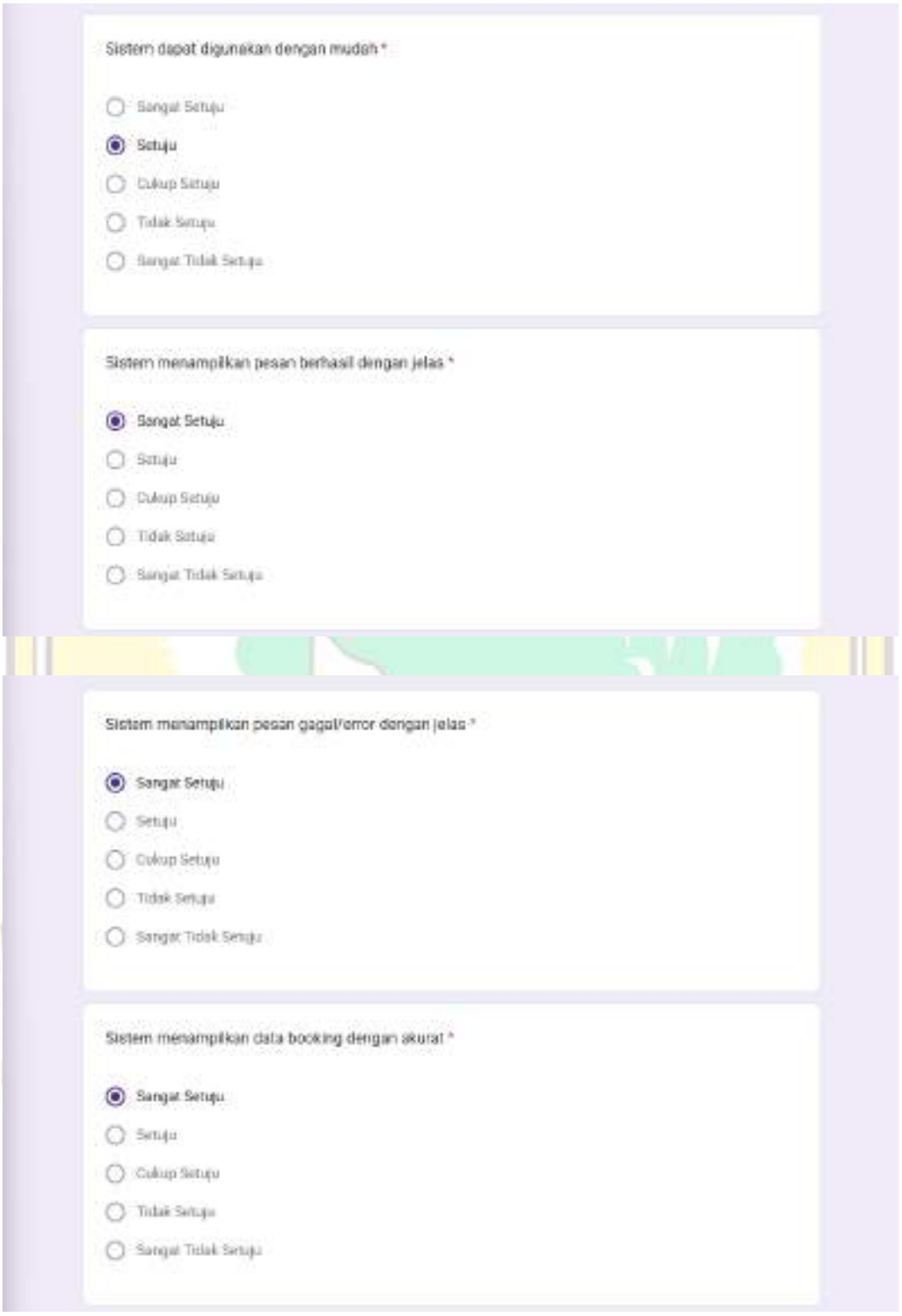
Tipe Alur Pengujian	Alternatif
Hasil Yang Diharapkan	Admin menolak booking pengguna dan merubah status booking menjadi "*Reject*"
Pengamatan	Sistem mengembalikan pesan bahwa booking telah ditolak dan konfirmasi terkirim ke email pengguna yang membooking
Hasil Akhir	Sesuai

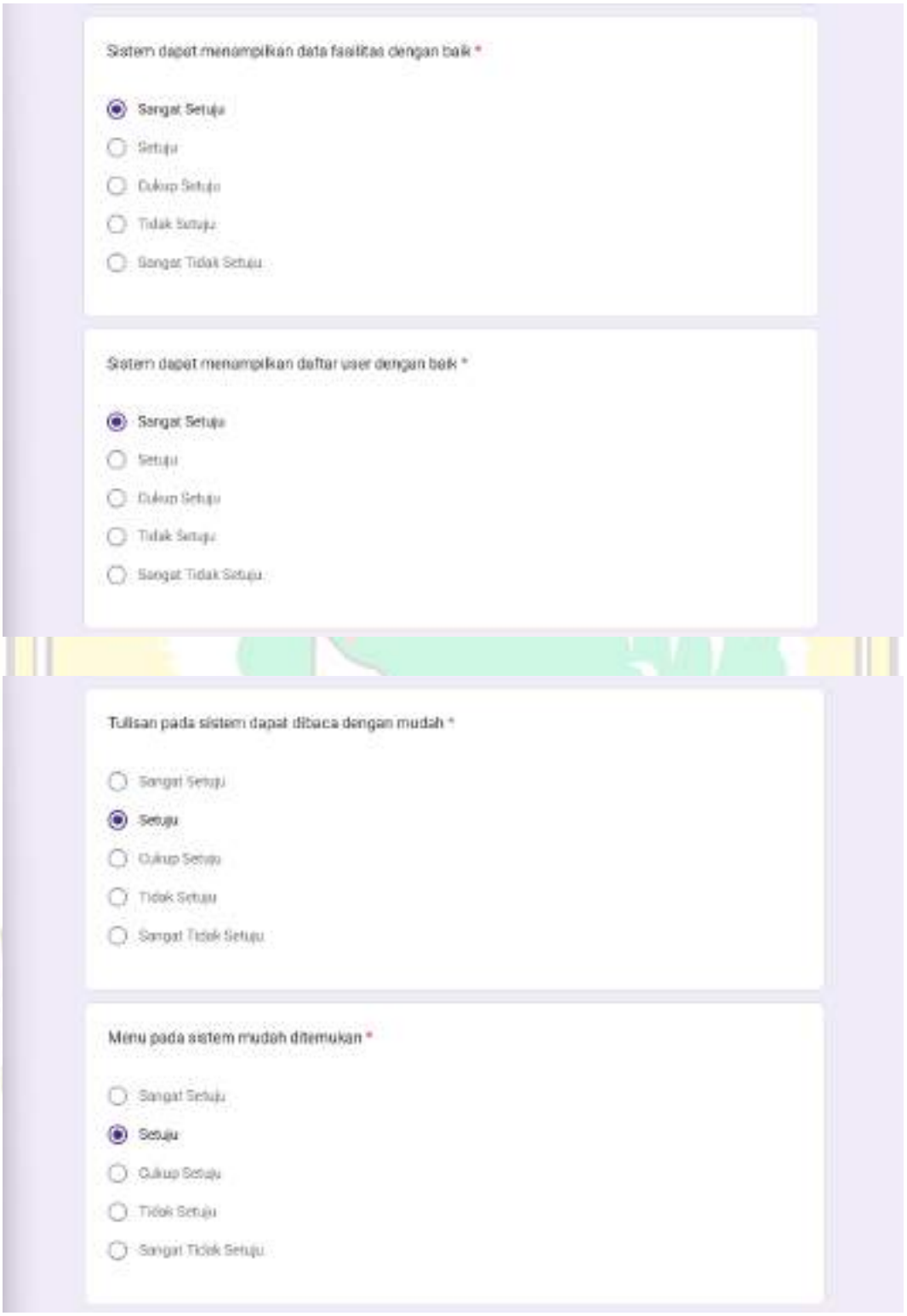
3. Form Pengujian Admin

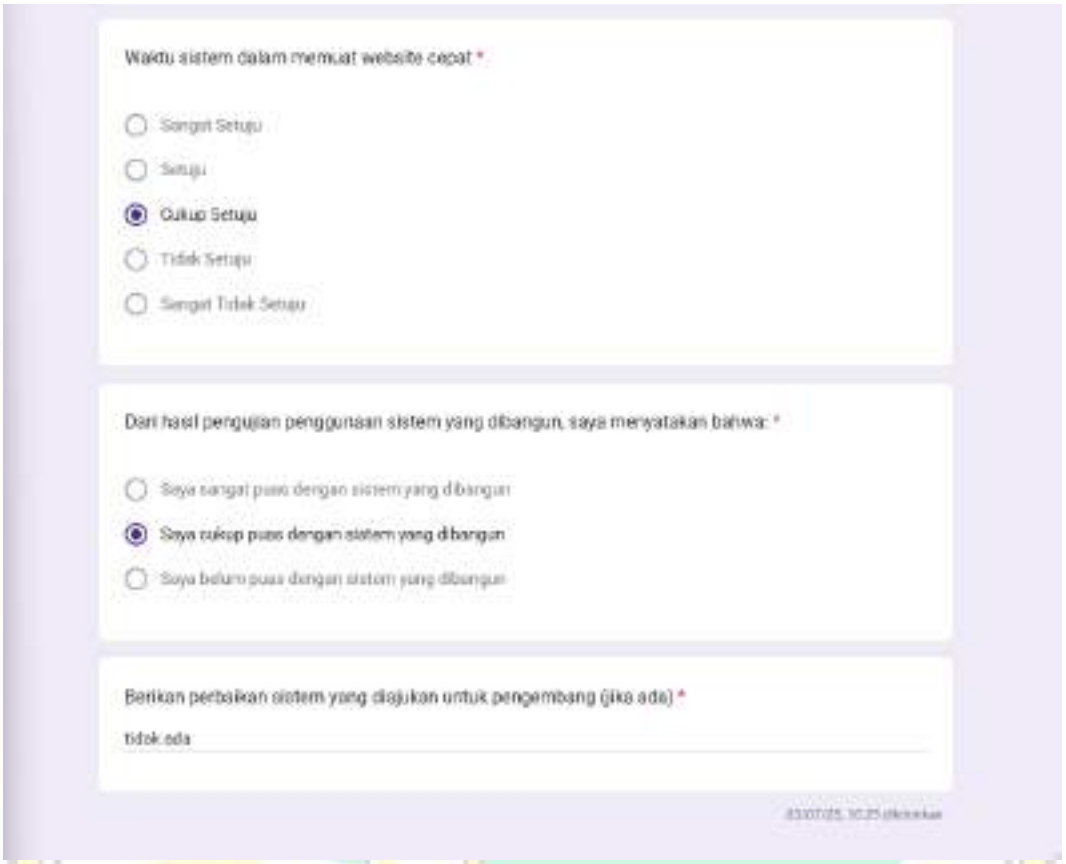



PENGUJIAN APLIKASI SISI BOOKING ROLE ADMIN
Apakah pada pengujian berhasil login dan diarahkan ke halaman dashboard? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai
Apakah pada pengujian dapat melakukan approval booking dengan baik dan data berhasil dikonfirmasi? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai
Apakah daftar data booking pengguna tampil di halaman list booking? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai
Apakah detail booking seperti nama, tanggal, waktu, dan agenda tampil dengan benar di halaman detail booking? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai


Apakah pada pengujian dapat mengelola data fasilitas (tambah, edit, hapus) tanpa kendala? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai
Apakah data fasilitas tampil dengan baik di halaman "Manage Fasilitas"? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai
Apakah pada pengujian dapat mengelola data user (tambah, edit, hapus) sesuai kebutuhan? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai
Apakah data user tampil dengan baik di halaman "Manage User"? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai


Sistem dapat digunakan dengan mudah *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju
Sistem menampilkan pesan berhasil dengan jelas *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju
Sistem menampilkan pesan gagal/error dengan jelas *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju
Sistem menampilkan data booking dengan akurat *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju


Sistem dapat menampilkan data fasilitas dengan baik *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju
Sistem dapat menampilkan daftar user dengan baik *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju
Tulisan pada sistem dapat dibaca dengan mudah *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju
Menu pada sistem mudah ditemukan *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju


Waktu sistem dalam memuat website cepat *

- Sangat Setuju
- Setuju
- (●) Cukup Setuju
- Tidak Setuju
- Sangat Tidak Setuju

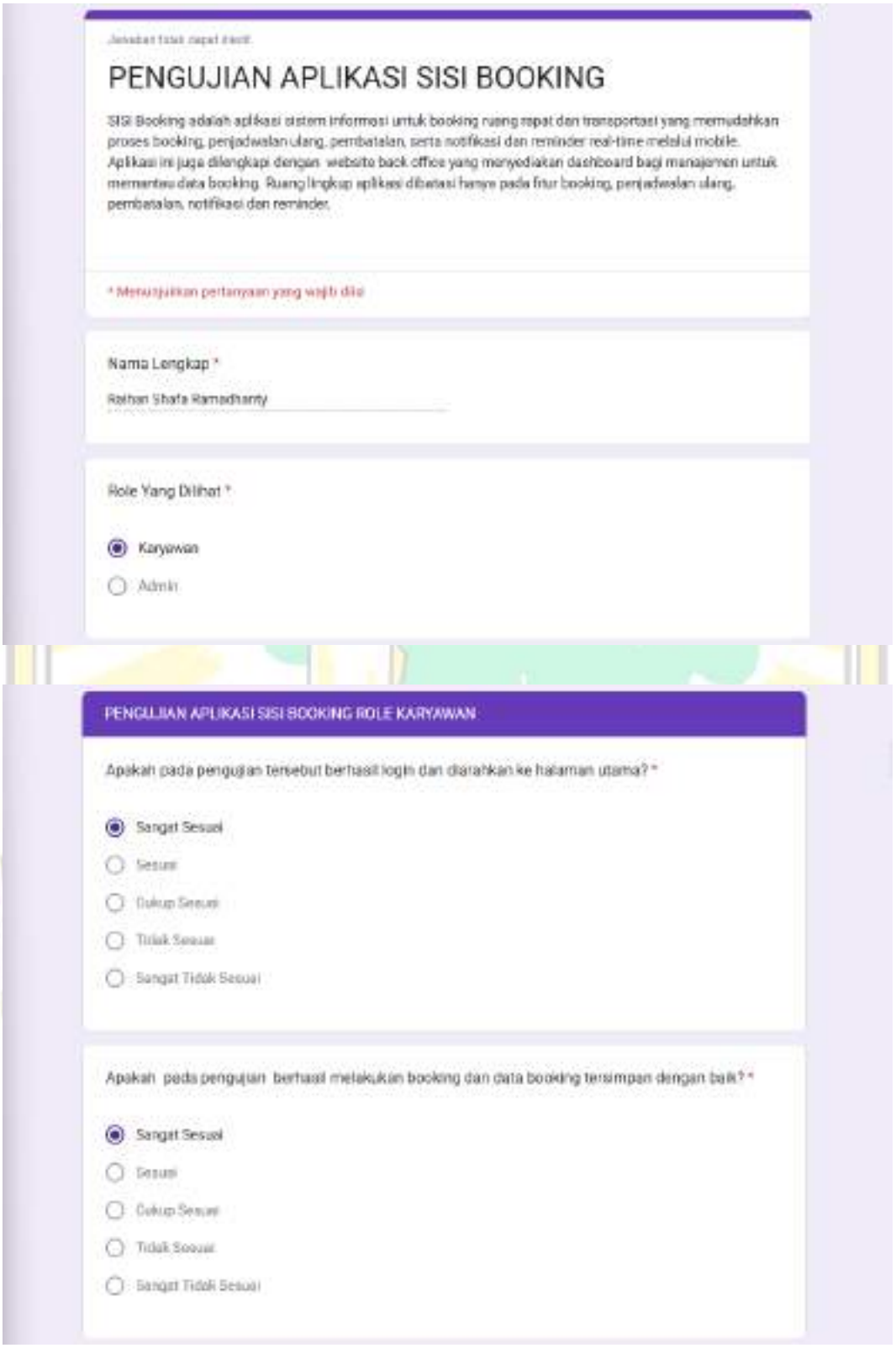
Dari hasil pengujian penggunaan sistem yang dibangun, saya menyatakan bahwa: *

- Saya sangat puas dengan sistem yang dibangun
- (●) Saya cukup puas dengan sistem yang dibangun
- Saya belum puas dengan sistem yang dibangun

Berikan perbaikan sistem yang diajukan untuk pengembang (jika ada) *

tidak ada

4. Form Pengujian Karyawan 1



Jawaban tidak dapat direkam

PENGUJIAN APLIKASI SISI BOOKING

SISI Booking adalah aplikasi sistem informasi untuk booking ruang rapat dan transportasi yang memudahkan proses booking, penjadwalan ulang, pembatalan, serta notifikasi dan reminder real-time melalui mobile. Aplikasi ini juga dilengkapi dengan website back office yang menyediakan dashboard bagi manajemen untuk memantau data booking. Ruang lingkup aplikasi dibatasi hanya pada fitur booking, penjadwalan ulang, pembatalan, notifikasi dan reminder.

* Menunjukkan pertanyaan yang wajib diisi

Nama Lengkap *

Raihan Shafa Ramadhanty

Role Yang Dilihat *

- (●) Karyawan
- ( ) Admin

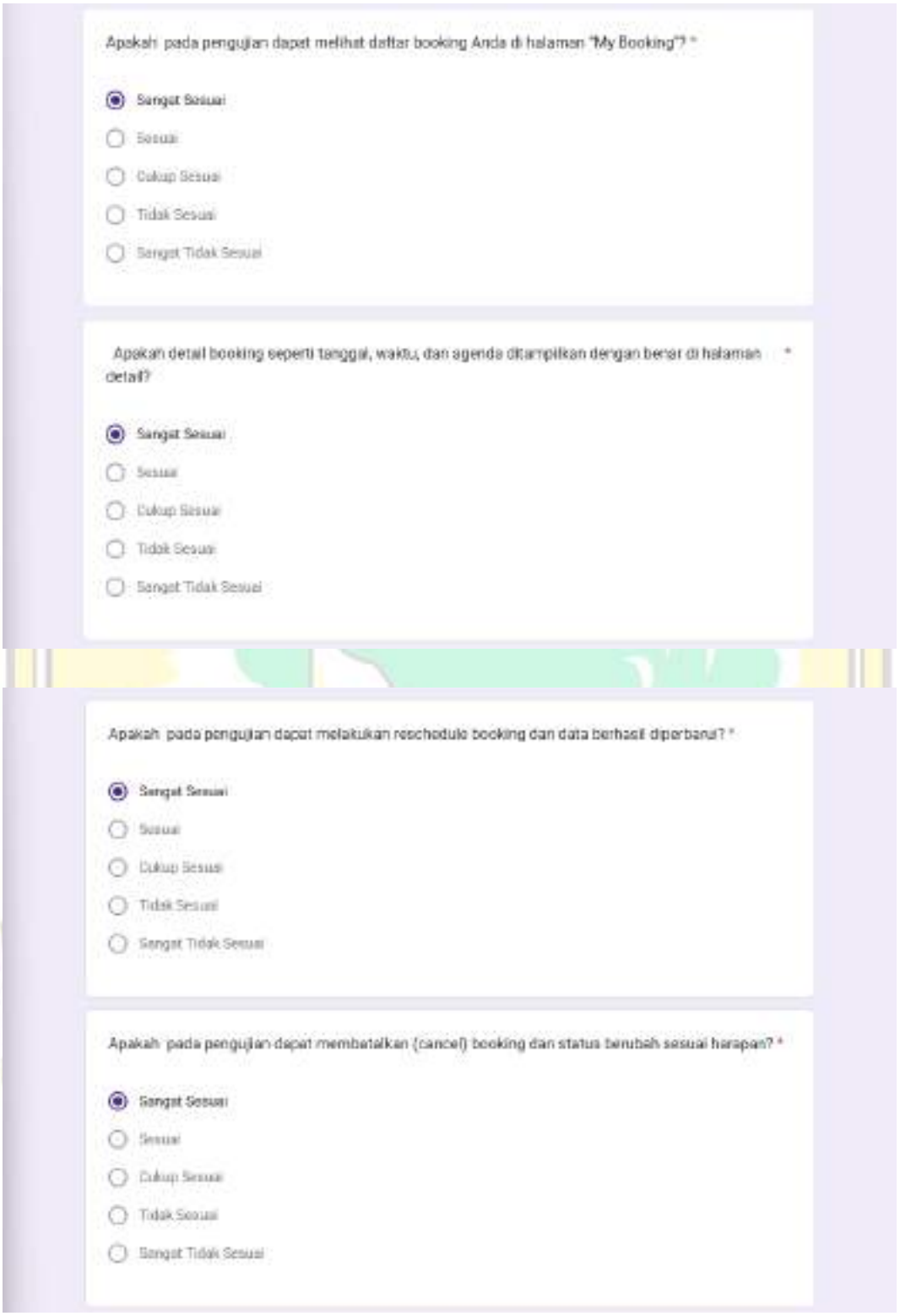
PENGUJIAN APLIKASI SISI BOOKING ROLE KARYAWAN

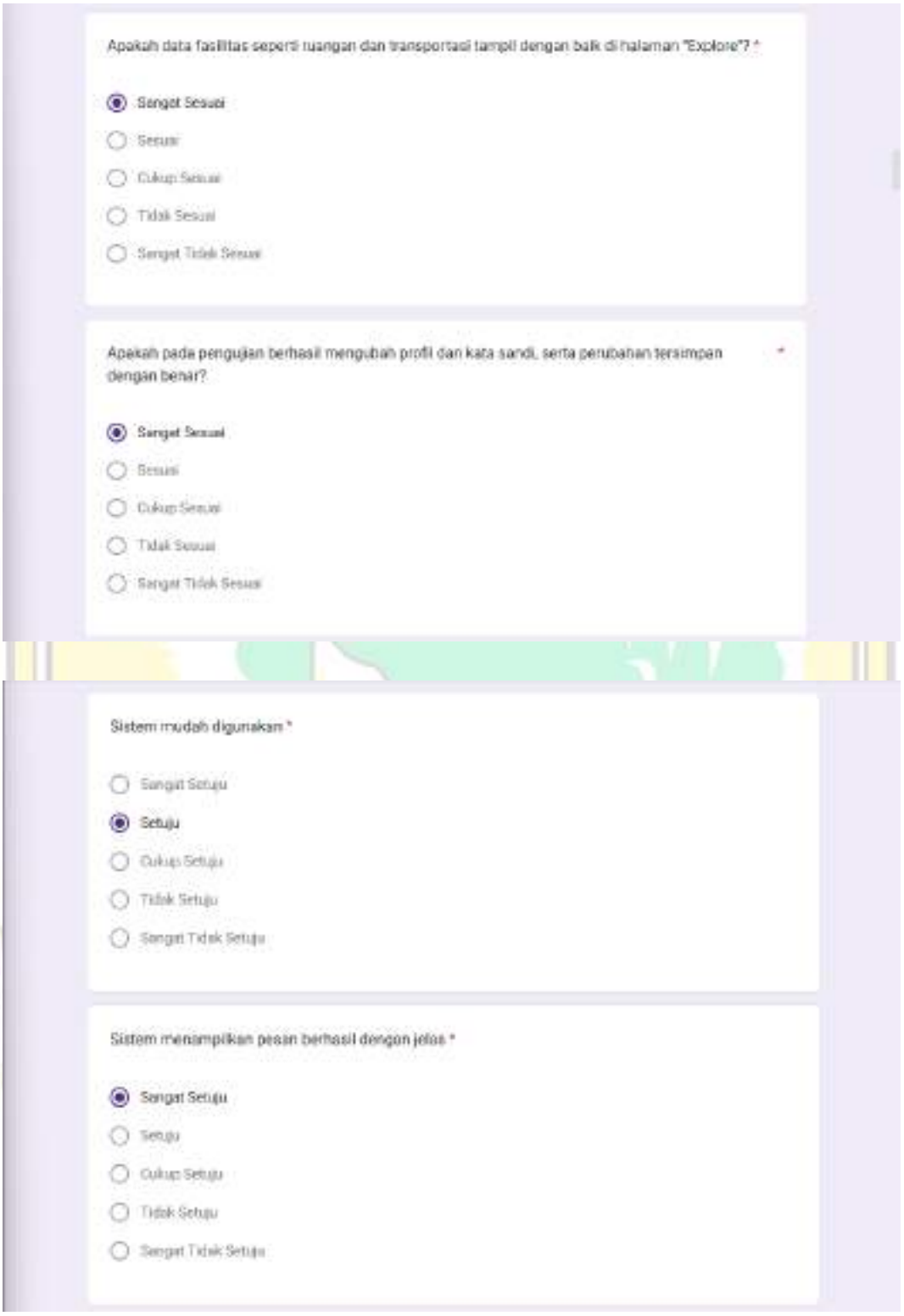
Apakah pada pengujian tersebut berhasil login dan diarahkan ke halaman utama? *

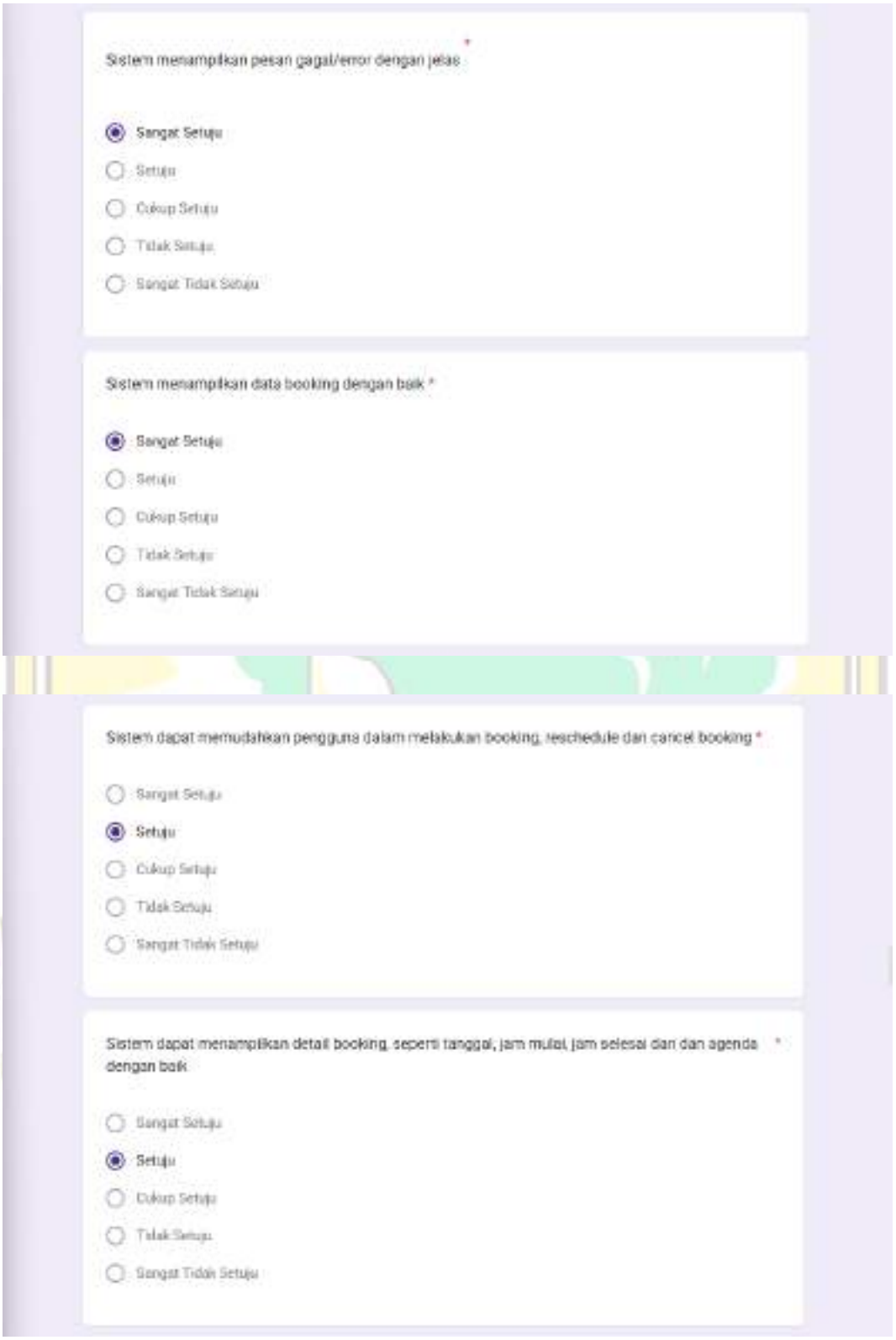
- (●) Sangat Sesuai
- ( ) Sesuai
- ( ) Cukup Sesuai
- ( ) Tidak Sesuai
- ( ) Sangat Tidak Sesuai

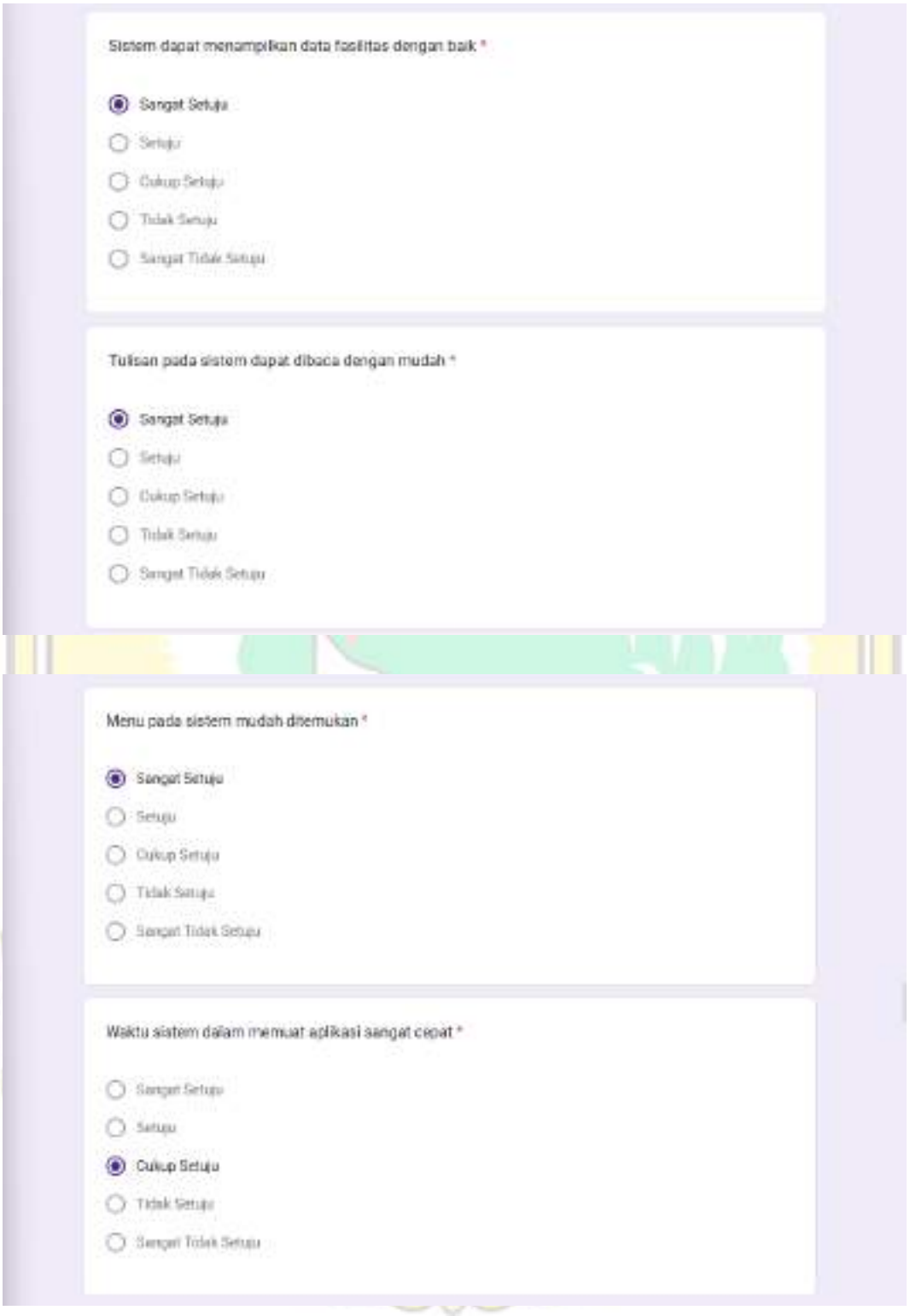
Apakah pada pengujian berhasil melakukan booking dan data booking tersimpan dengan baik? *

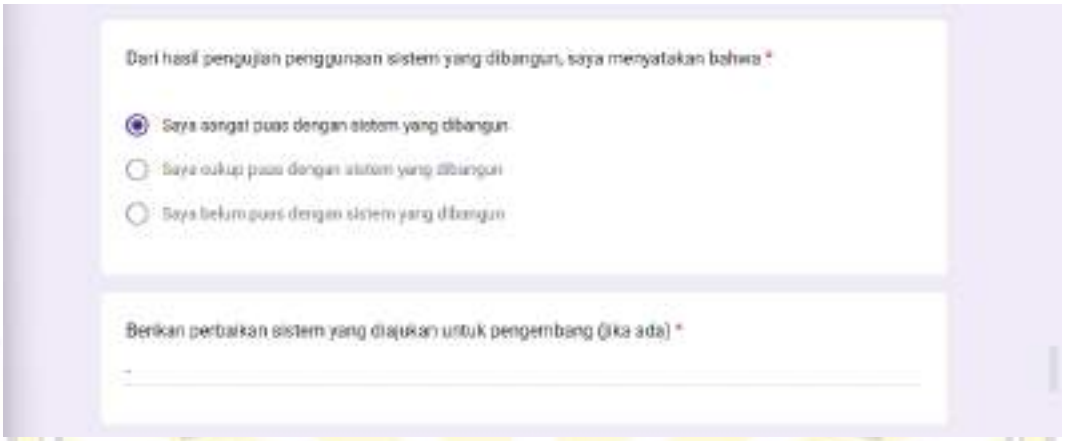
- (●) Sangat Sesuai
- ( ) Sesuai
- ( ) Cukup Sesuai
- ( ) Tidak Sesuai
- ( ) Sangat Tidak Sesuai


Apakah pada pengujian dapat melihat daftar booking Anda di halaman "My Booking"? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai
Apakah detail booking seperti tanggal, waktu, dan agenda ditampilkan dengan benar di halaman detail? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai
Apakah pada pengujian dapat melakukan reschedule booking dan data berhasil diperbarui? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai
Apakah pada pengujian dapat membatalkan (cancel) booking dan status berubah sesuai harapan? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai

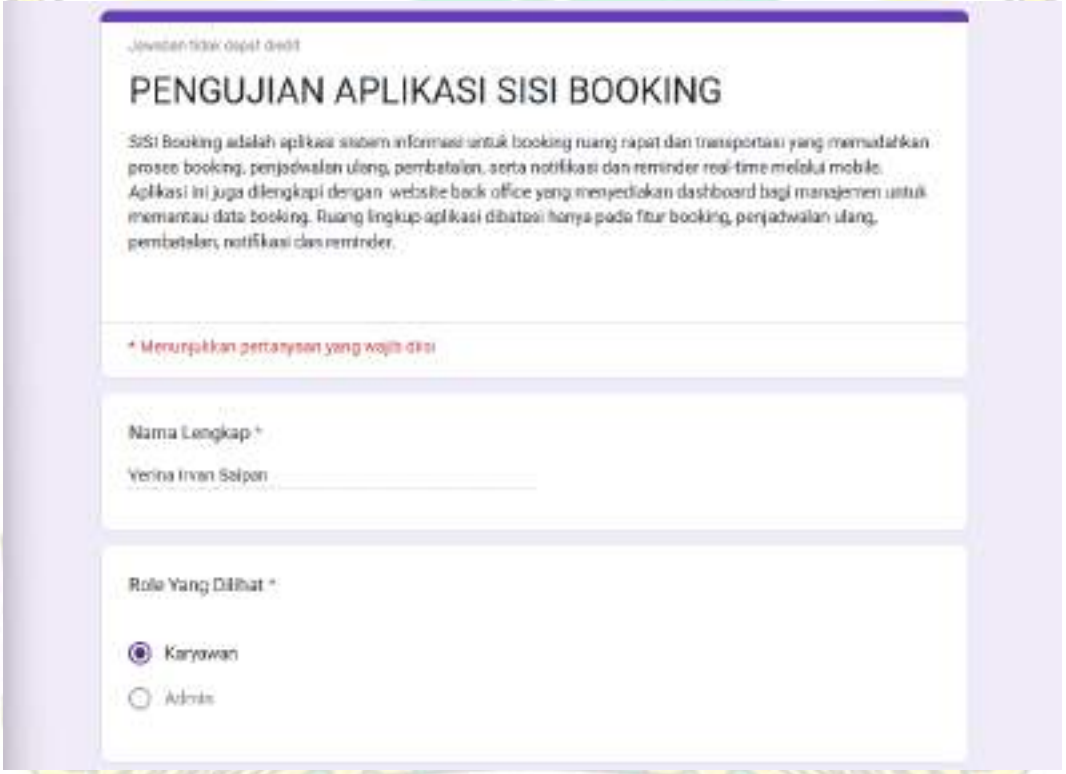

Apakah data fasilitas seperti ruangan dan transportasi tampil dengan baik di halaman "Explore"? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai
Apakah pada pengujian berhasil mengubah profil dan kata sandi, serta perubahan tersimpan dengan benar? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai
Sistem mudah digunakan *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju
Sistem menampilkan pesan berhasil dengan jelas *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju

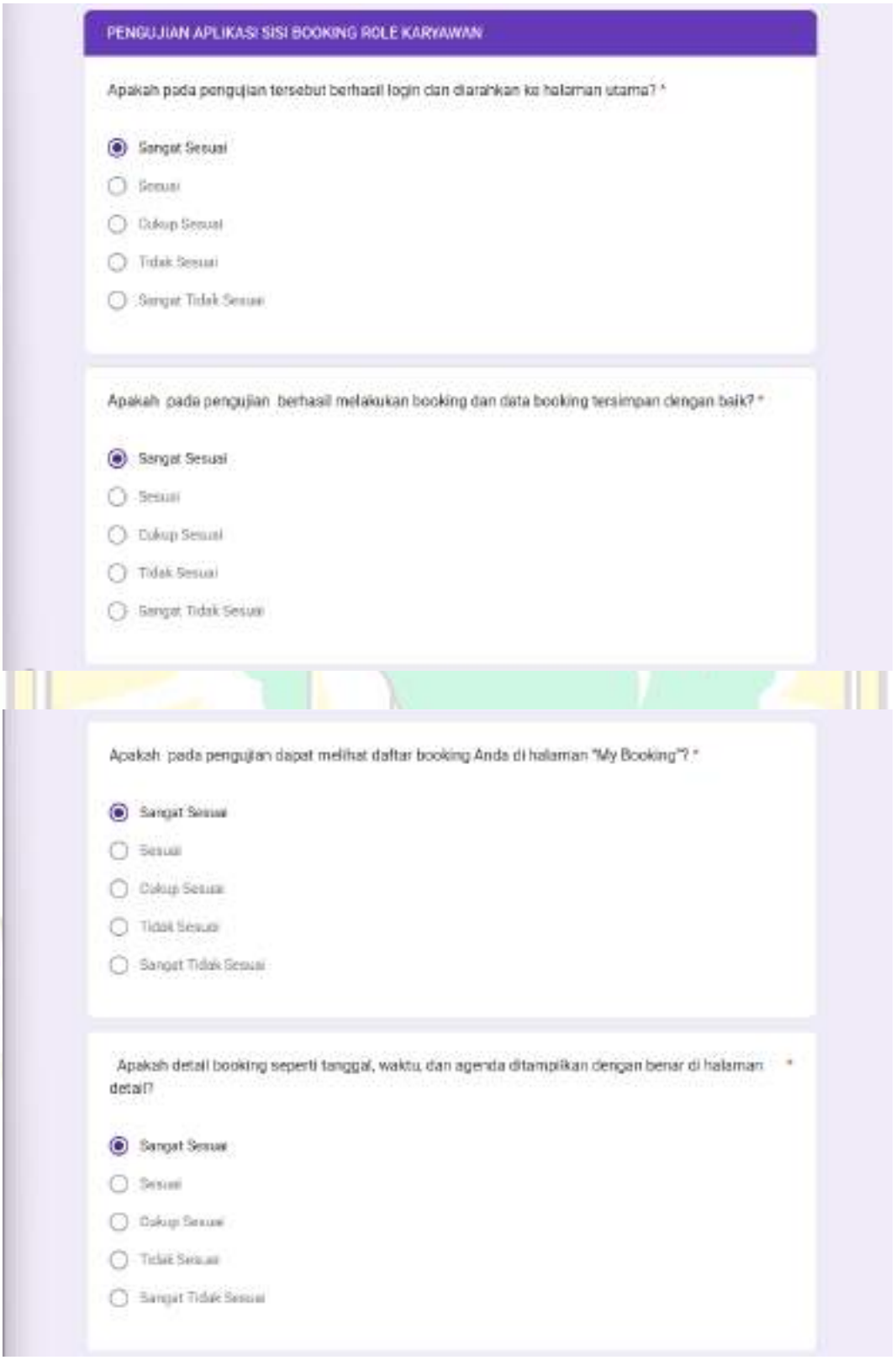

Sistem menampilkan pesan gagal/error dengan jelas *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju
Sistem menampilkan data booking dengan baik *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju
Sistem dapat memudahkan pengguna dalam melakukan booking, reschedule dan cancel booking *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju
Sistem dapat menampilkan detail booking, seperti tanggal, jam mulai, jam selesai dan dan agenda dengan baik *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju

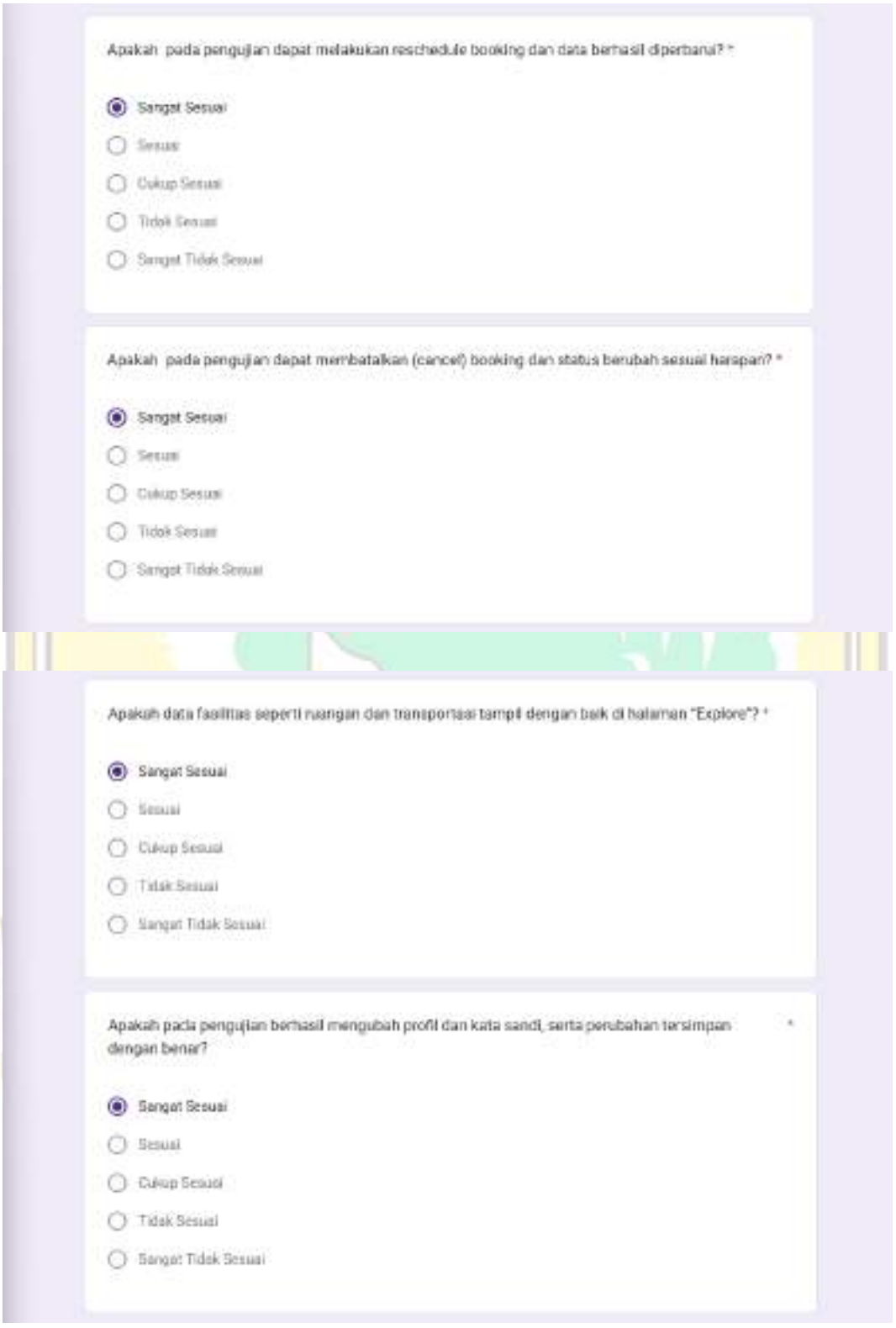

Sistem dapat menampilkan data fasilitas dengan baik *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju
Tulisan pada sistem dapat dibaca dengan mudah *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju
Menu pada sistem mudah ditemukan *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju
Waktu sistem dalam memuat aplikasi sangat cepat *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju

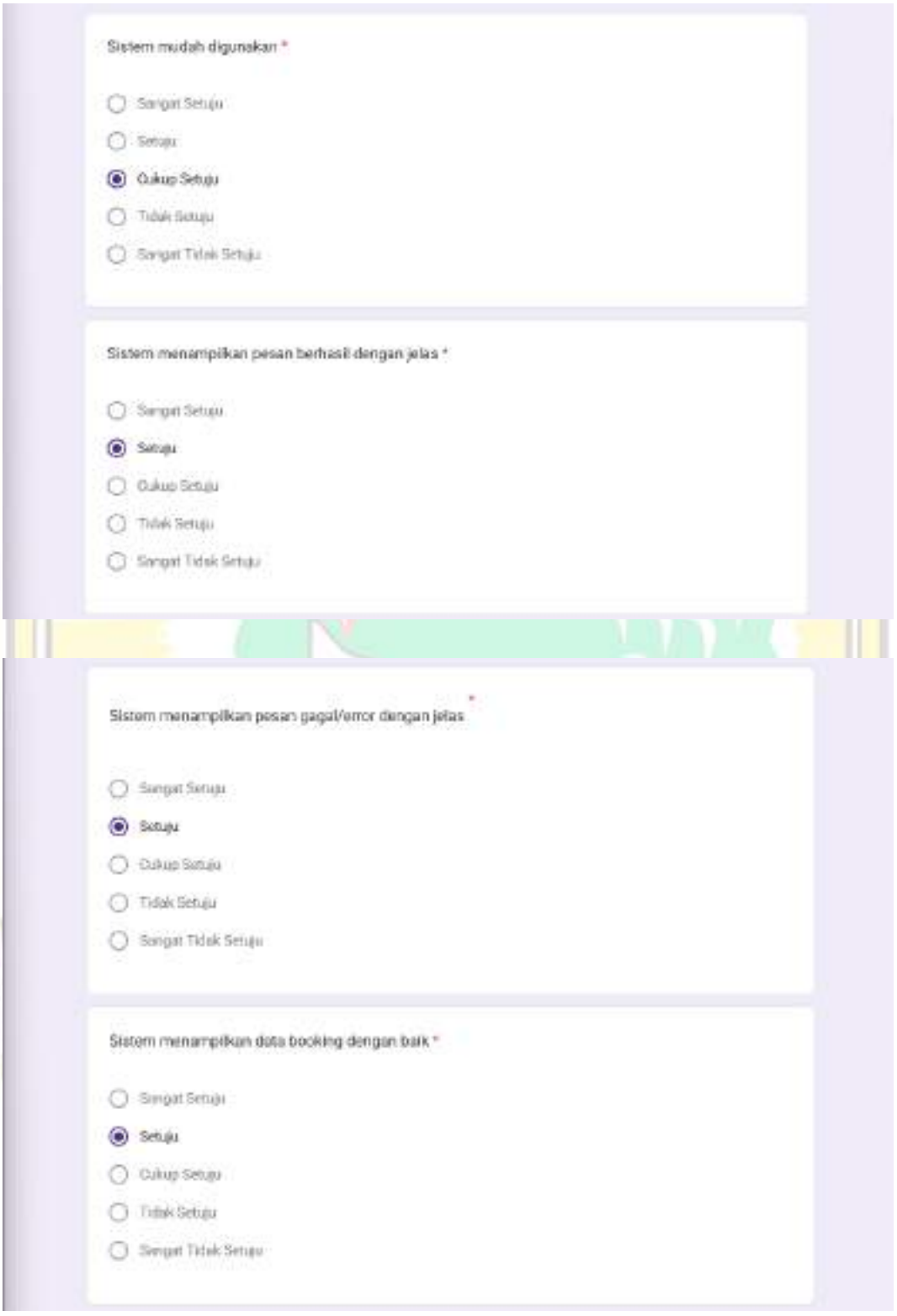


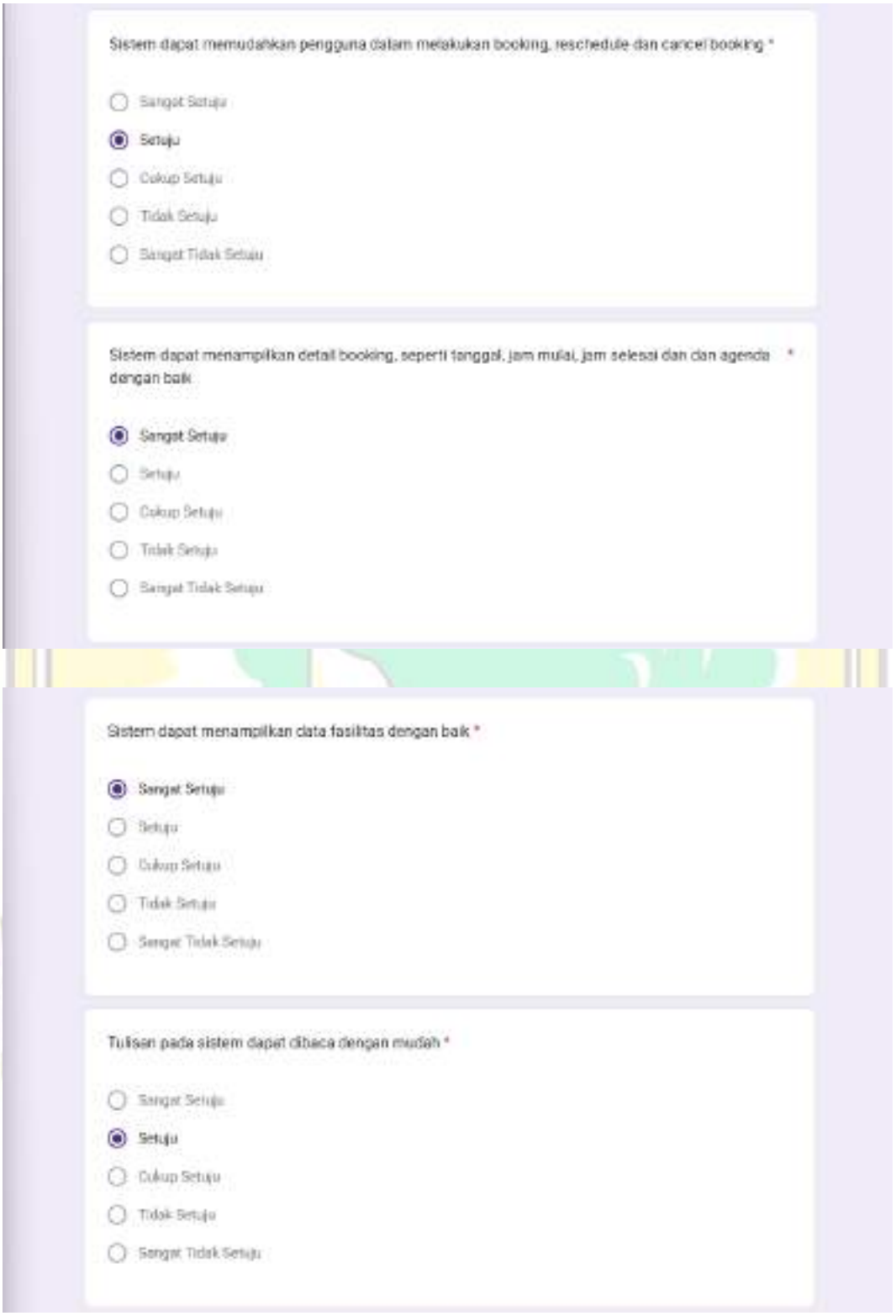
5. Form Pengujian Karyawan 2




PENGUJIAN APLIKASI SISI BOOKING ROLE KARYAWAN
Apakah pada pengujian tersebut berhasil login dan diarahkan ke halaman utama? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai
Apakah pada pengujian berhasil melakukan booking dan data booking tersimpan dengan baik? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai
Apakah pada pengujian dapat melihat daftar booking Anda di halaman "My Booking"? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai
Apakah detail booking seperti tanggal, waktu, dan agenda ditampilkan dengan benar di halaman detail? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai


Apakah pada pengujian dapat melakukan reschedule booking dan data berhasil diperbarui? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai
Apakah pada pengujian dapat membatalkan (cancel) booking dan status berubah sesuai harapan? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai
Apakah data fasilitas seperti ruangan dan transportasi tampil dengan baik di halaman "Explore"? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai
Apakah pada pengujian berhasil mengubah profil dan kata sandi, serta perubahan tersimpan dengan benar? *
Sangat Sesuai
Sesuai
Cukup Sesuai
Tidak Sesuai
Sangat Tidak Sesuai


Sistem mudah digunakan *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju
Sistem menampilkan pesan berhasil dengan jelas *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju
Sistem menampilkan pesan gagal/error dengan jelas *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju
Sistem menampilkan data booking dengan baik *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju


Sistem dapat memudahkan pengguna dalam melakukan booking, reschedule dan cancel booking *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju
Sistem dapat menampilkan detail booking, seperti tanggal, jam mulai, jam selesai dan clan agenda dengan baik *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju
Sistem dapat menampilkan data fasilitas dengan baik *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju
Tulisan pada sistem dapat dibaca dengan mudah *
Sangat Setuju
Setuju
Cukup Setuju
Tidak Setuju
Sangat Tidak Setuju

Menu pada sistem mudah ditemukan *

- ○ Sangat Setuju
- ◉ Setuju
- ○ Cukup Setuju
- ○ Tidak Setuju
- ○ Sangat Tidak Setuju

Waktu sistem dalam memuat aplikasi sangat cepat *

- ○ Sangat Setuju
- ○ Setuju
- ◉ Cukup Setuju
- ○ Tidak Setuju
- ○ Sangat Tidak Setuju

Dari hasil pengujian penggunaan sistem yang dibangun, saya menyatakan bahwa *

- ◉ Saya sangat puas dengan sistem yang dibangun
- ○ Saya cukup puas dengan sistem yang dibangun
- ○ Saya belum puas dengan sistem yang dibangun

Berikan perbaikan sistem yang diajukan untuk pengembang (jika ada) *

tidak ada

LAMPIRAN G

DATA DOKUMEN (LANJUTAN)

1. Permohonan Izin Survey

KEMENTERIAN PENDIDIKAN TINGGI, SAINS, DAN TEKNOLOGI
UNIVERSITAS ANDALAS
FAKULTAS TEKNOLOGI INFORMASI
DEPARTEMEN SISTEM INFORMASI
Kampus Universitas Andalas, Limau Manis, Padang, Kode Pos 25163
Telp: 0751-9824667 website: http://si.ft.unand.ac.id

Nomor : 155/UN.16.15.3.2/KM.07/2025 Padang, 27 Mei 2025
Hal : **Permohonan Izin Survey dan Pengambilan Data Tugas Akhir**

Kepada Yth.
Pimpinan PT Sinergi Informatika Semen Indonesia
Di
Tempat

Assalamu'alaikum Wr.Wb.

Bersama ini disampaikan bahwa mahasiswa Departemen Sistem Informasi Fakultas Teknologi Informasi Universitas Andalas berikut :

Nama	: Fikran Shadiq Elyafit
NIM	: 2111521024
Judul TA	: Pembangunan Sistem Informasi *Booking* Ruang Rapat dan Transportasi Berbasis *Mobile* Pada PT SISI (Sinergi Informatika Semen Indonesia)

Akan melaksanakan survey, wawancara dan pengambilan data untuk keperluan Tugas Akhir. Untuk itu, mohon izin, bantuan, dan kerjasama dalam pengambilan data yang diperlukan.

Demikianlah surat ini disampaikan, atas perhatian dan kerjasamanya diucapkan terima kasih.

Ketua,

Ricky Akbar, M.Kom
NIP. 19841006201212100 1

Tembusan :
1. Arsip

2. Rekap Hasil Wawancara

Jadwal wawancara 31 oktober 2024

Pewawancara:

Nama : Fikran Shadiq Elyafit

NIM : 2111521024

Judul TA : Pembangunan Sistem Informasi *Booking* Ruang Rapat dan Transportasi Berbasis *Mobile* Pada PT SISI (SINERGI INFORMATIKA SEMEN INDONESIA)

a. Pertanyaan : **Apa saja kendala atau kesulitan yang mungkin dihadapi saat menggunakan sistem yang sudah kami buat sebelumnya?**

   Jawaban : Sejujurnya, saya sendiri belum terlalu mendalami sistemnya. Jadi saya belum bisa memberikan banyak masukan. Saya ingat pernah memberikan beberapa catatan di awal, tetapi saya lupa detailnya

b. Pertanyaan : **Mengenai sistem pemesanan (booking) yang ada saat sebelumnya, di mana pengguna memasukkan data lalu admin yang akan memprosesnya, apakah menurut Anda proses tersebut sudah efektif?**

   Jawaban : Menurut saya, itu **kurang efektif**. Ada kemungkinan admin bisa lupa atau terlewat dalam mencatat pesanan. Akan jauh lebih baik jika sistemnya memungkinkan karyawan untuk melakukan pemesanan secara mandiri (*self-service*). Dengan begitu, mereka bisa langsung melihat jadwal yang tersedia dan melakukan pemesanan sendiri tanpa harus bergantung pada admin.

c. Pertanyaan : **Dari beberapa fitur yang ada, seperti pemantauan (*monitoring*), persetujuan (*approval*), dan penambahan pengguna, fitur mana yang menurut Anda paling bermanfaat?**

Jawaban : Sistem pemesanan seperti ini secara keseluruhan sangat membantu. Ini dapat meringankan beban kerja admin dan memungkinkan karyawan untuk langsung melihat jadwal ruangan yang tersedia tanpa perlu bertanya berulang kali.

d. Pertanyaan : **Apakah menurut Anda perlu ada fitur untuk membuat laporan dari data pemesanan yang terkumpul?**

Jawaban : Ya, itu **sangat diperlukan.** Fitur laporan ini penting untuk rekapitulasi. Idealnya, laporan tersebut bisa diekspor ke format Excel dan mencakup detail seperti: Nama Ruangan, Tanggal, Nama Pemesan (PIC), Divisi, Agenda Rapat, Waktu Mulai, Waktu Selesai, dan jika ada, permintaan khusus seperti pesanan makanan atau minuman.

e. Pertanyaan : **Apakah ada fitur lain yang Anda rasa perlu ditambahkan ke dalam sistem ini?**

Jawaban : Ya, sangat perlu ditambahkan fitur untuk **membatalkan (*cancel*)** atau **menjadwalkan ulang (*reschedule*)** pemesanan. Ini sangat penting karena rencana rapat sering kali dapat berubah.

f. Pertanyaan : **Bagaimana dengan sistem pemesanan untuk transportasi? Apakah perlu dibuatkan sistem serupa?**

Jawaban : Untuk transportasi, situasinya lebih rumit karena **kondisi lalu lintas yang tidak dapat diprediksi**. Sulit untuk memastikan seorang pengemudi akan selesai tepat waktu sesuai jadwal, yang berisiko menyebabkan bentrok dengan pesanan berikutnya. Jika ingin dibuatkan, perlu juga dicatat detail seperti destinasi, kendaraan yang digunakan, misalnya Avanza untuk 5 orang atau Innova untuk 7 orang.

g. Pertanyaan : **Terakhir, apakah sistem ini akan lebih baik jika dikembangkan menjadi aplikasi mobile?**

Jawaban : Ya, itu ide yang sangat bagus. Aplikasi versi **mobile akan sangat memudahkan pengguna** untuk melakukan pemesanan dari mana saja. Sementara itu, admin bisa tetap menggunakan versi desktop untuk mengelola sistem, dan pengguna bisa memakai aplikasi mobile yang lebih praktis.

Berikut adalah 3 masukan utama dari narasumber untuk pengembangan sistem:

1. **Utamakan Sistem Mandiri (*Self-Service*):** Pengguna harus bisa memesan, membatalkan, dan menjadwal ulang booking sendiri tanpa perlu melalui admin.
2. **Tambahkan Fitur Laporan:** Sistem wajib memiliki fitur untuk ekspor data pemesanan ke Excel untuk keperluan rekapitulasi admin.
3. **Kembangkan Versi Mobile:** Aplikasi akan jauh lebih praktis dan sangat disarankan untuk dikembangkan dalam versi mobile untuk kemudahan pengguna.

3. Lembar Pernyataan Admin

LEMBAR PERNYATAAN

Dengan ini saya yang bertanda tangan di bawah ini:

Nama	: Zainal Abidin
Jabatan	: Team Lead of Consulting

Menyatakan bahwa mahasiswa Perguruan Tinggi Negeri Berbadan Hukum yang bernama:

Nama	: Fikran Shadiq Elyafit
NIM	: 2111521024
Departemen	: Sistem Informasi
Fakultas	: Teknologi Informasi
PTN BH	: Universitas Andalas

Terkait dengan tugas akhir "**Pembangunan Sistem Informasi Booking Ruang Rapat dan Transportasi Berbasis Mobile di PT Sinergi Informatika Semen Indonesia**" bahwa sistem informasi yang dibangun telah sesuai dengan proses bisnis, sesuai dengan kebutuhan dari PT Sinergi Informatika Semen Indonesia, dan menyatakan data yang diberikan merupakan data yang sesungguhnya dan tidak direkayasa.

Jakarta Selatan, 03 Juli 2025

Penguji

(Zainal Abidin)

4. Lembar Pernyataan Karyawan 1

LEMBAR PERNYATAAN

Dengan ini saya yang bertanda tangan di bawah ini:

Nama	: Raihan Shafa Ramadhanty
Jabatan	: Staff Human Capital

Menyatakan bahwa mahasiswa Perguruan Tinggi Negeri Berbadan Hukum yang bernama:

Nama	: Fikrin Shadiq Elyafit
NIM	: 2111521024
Departemen	: Sistem Informasi
Fakultas	: Teknologi Informasi
PTN BH	: Universitas Andalas

Terkait dengan tugas akhir "**Pembangunan Sistem Informasi Booking Ruang Rapat dan Transportasi Berbasis Mobile di PT Sinergi Informatika Semen Indonesia**" bahwa sistem informasi yang dibangun telah sesuai dengan proses bisnis, sesuai dengan kebutuhan dari PT Sinergi Informatika Semen Indonesia, dan menyatakan data yang diberikan merupakan data yang sesungguhnya dan tidak direkayasa.

Jakarta Selatan, 03 Juli 2025

Penguji

(Raihan Shafa Ramadhanty)

5. Lembar Pernyataan Karyawan 2

LEMBAR PERNYATAAN

Dengan ini saya yang bertanda tangan di bawah ini:

Nama	: Verina Irvan Saipan
Jabatan	: Staff Human Capital

Menyatakan bahwa mahasiswa Perguruan Tinggi Negeri Berbadan Hukum yang bernama:

Nama	: Fikran Shafiq Elyafit
NIM	: 2111521024
Departemen	: Sistem Informasi
Fakultas	: Teknologi Informasi
PTN BH	: Universitas Andalas

Terkait dengan tugas akhir "**Pembangunan Sistem Informasi Booking Ruang Rapat dan Transportasi Berbasis Mobile di PT Sinergi Informatika Semen Indonesia**" bahwa sistem informasi yang dibangun telah sesuai dengan proses bisnis, sesuai dengan kebutuhan dari PT Sinergi Informatika Semen Indonesia, dan menyatakan data yang diberikan merupakan data yang sesungguhnya dan tidak direkayasa.

Jakarta Selatan, 03 Juli 2025

Penguji

(Verina Irvan Saipan)

Fikran Shadiq Elyafit - 2111521024 - Laporan TA

ORIGINALITY REPORT

5%	6%	3%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universitas Andalas Student Paper	3%
2	scholar.unand.ac.id Internet Source	2%
3	sisi.id Internet Source	1%

Exclude quotes	Off	Exclude matches	< 1%
Exclude bibliography	Off		