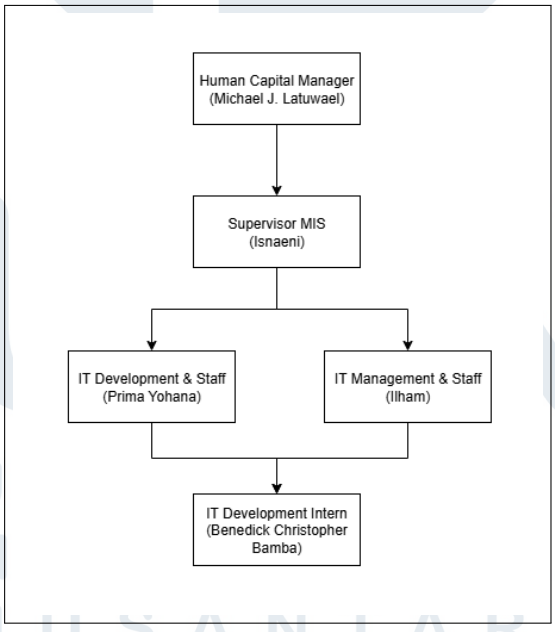
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama pelaksanaan kerja magang, posisi yang ditempati adalah *IT Development Intern* pada divisi *Management Information Systems* (*MIS*). Divisi *MIS* berada di bawah koordinasi *Manager Human Capital*, yaitu Bapak Michael J. Latuwael. Bapak Isnaeni selaku *supervisor MIS* bertindak sebagai pembimbing utama yang memberikan arahan, menugaskan kegiatan terkait proyek pengembangan sistem, serta melakukan pemantauan progres secara berkala. Dalam pelaksanaannya, *intern* juga berkoordinasi dengan Saudari Prima Yohana (*IT Development & Staff*) yang memberikan masukan terkait antarmuka pengguna, serta Saudara Ilham (*IT Management & Staff*) yang turut memberikan dukungan dalam berbagai kegiatan divisi.



Gambar 3.1. Kedudukan *IT Development Intern* dalam Struktur Divisi *MIS*

Mekanisme koordinasi dilakukan melalui pertemuan harian untuk meninjau perkembangan proyek dan menyelaraskan hasil kerja dengan kebutuhan perusahaan. Selain fokus pada pengembangan aplikasi, *intern* turut mendukung kegiatan divisi *MIS* lainnya, seperti penanganan permintaan *helpdesk IT* dan penginputan data ke dalam sistem *ERP* perusahaan, sesuai arahan dari pihak terkait.

Struktur koordinasi ini memastikan bahwa seluruh kegiatan magang berjalan terarah dan selaras dengan tujuan divisi maupun perusahaan secara keseluruhan.

3.2 Tugas yang Dilakukan

Selama pelaksanaan kegiatan magang di PT. Jaya Bersama Saputra Perkasa pada divisi *Management Information Systems* (*MIS*) dalam pengembangan aplikasi *mobile cross-platform Flutter* dan integrasi sistem Odoo ERP, tugas-tugas yang dilakukan meliputi:

1. Analisis kebutuhan pengguna dan sistem untuk pengembangan aplikasi *mobile* berbasis *Flutter* yang terintegrasi dengan Odoo ERP.
2. Perancangan *flowchart* dan *mockup* aplikasi *mobile* yang intuitif.
3. Implementasi fitur-fitur aplikasi *mobile cross-platform* menggunakan *framework Flutter* untuk *platform Android* dan *iOS*.
4. Pengembangan dan kustomisasi modul-modul Odoo ERP Perusahaan.
5. Integrasi aplikasi *mobile Flutter* dengan sistem Odoo ERP melalui *JSON-RPC API* untuk memastikan sinkronisasi data secara *real-time*.
6. Pengujian aplikasi untuk memastikan fungsionalitas, performa, dan kompatibilitas lintas *platform* berjalan dengan baik.
7. Perbaikan *bug* dan revisi fitur berdasarkan *feedback* dari *supervisor* dan pengguna aplikasi.
8. Penanganan permintaan *helpdesk IT* dan dukungan teknis terkait sistem informasi perusahaan.
9. Penginputan dan pengelolaan data ke dalam sistem Odoo ERP sesuai kebutuhan operasional divisi.
10. Pelaporan progres harian dan mingguan kepada *supervisor* mengenai perkembangan proyek pengembangan aplikasi.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan magang dilaksanakan dari hari Senin hingga Sabtu dengan total 40 jam kerja per minggu. Rincian aktivitas dan jadwal pelaksanaan magang dijabarkan pada Tabel 3.3.

Tabel 3.1. Uraian Kegiatan Magang per Minggu (Minggu 1-20)

Minggu	**Tanggal**	**Pekerjaan yang dilakukan**
1	4-9 Ags 2025	Orientasi divisi MIS, mempelajari sistem Odoo ERP, observasi *workflow existing*, dan instalasi *software* operasional.
2	11-15 Ags 2025	Pembuatan dokumentasi *workflow*, instalasi *Linux Debian Server* di *Virtual Machine*, dan konfigurasi jaringan.
3	19-23 Ags 2025	Konfigurasi *NAT Network*, instalasi *WordPress*, *Webmin*, *Odoo Server*, *PostgreSQL*, dan *Laravel*.
4	25-30 Ags 2025	*Setup Flutter development environment*, pembuatan fitur *Login* dengan integrasi *API* Odoo, pembuatan *homepage* dan modul *Employee*.
5	1-6 Sep 2025	Pembuatan fitur *Remember Me*, modul *Sales* (*Quotations*, *Products*, *Customers*), pembuatan *UML*, dan modul *Sales Promotor*.
6	8-13 Sep 2025	Pembuatan modul Promotor dan Toko, konfigurasi *Postman* untuk *testing API*, dan implementasi visualisasi *chart*.
7	15-20 Sep 2025	Pengembangan modul *CRM* (*My Pipeline*), implementasi *chart*, dan pembuatan *mockup UI* berbagai modul.
8	22-27 Sep 2025	Implementasi *UI* untuk *Login*, *Sales*, dan *Customers*, pembuatan fitur *Geolocation Attendances*, dan perbaikan format angka.
9	29 Sep-4 Okt 2025	Pengembangan modul *Attendances* berbasis *GPS*, pembuatan model *hr_attendance_geofence*, dan implementasi *UI History* dan *Profile*.

Bersambung ke halaman berikutnya

Tabel 3.1. Uraian Kegiatan Magang per Minggu (Minggu 1-20) (lanjutan)

Minggu	Tanggal	Pekerjaan yang dilakukan
10	6-11 Okt 2025	Migrasi *HTTP* ke *HTTPS*, pengembangan model *geofence* terintegrasi *operating unit*, pembuatan sistem *auto reminder* kontrak, dan modul *Approval Contract*.
11	13-18 Okt 2025	Pembuatan modul *Approval Contract* dan *My Contract Approval*, pembuatan modul *multi approval system generic*.
12	20-25 Okt 2025	Perbaikan *UI Flutter App*, implementasi status *Geofence Area*, *bug fixing*, dan optimasi aplikasi.
13	27 Okt-1 Nov 2025	Modifikasi *UI Flutter App*, pembuatan animasi status lokasi *attendances*, perbaikan fitur *Sales* dan *Attendances*, konfigurasi *Outgoing Mail Servers* Odoo, dan mempelajari serta konfigurasi *SIP Server* menggunakan *Twilio*.
14	3-8 Nov 2025	Integrasi *SIP Provider*, pembuatan simulasi *AR* produk pintu menggunakan *Flutter* dengan *Google ML Kit Object Detection*, modifikasi *UI* dan *color palette*, pembuatan modul Odoo penyimpanan foto produk, dan implementasi fitur Resize.
15	10-15 Nov 2025	Pengembangan modul Odoo *Website TryOn Doors*, modifikasi *UI* dan fitur deteksi objek, integrasi ke *Flutter App*, pencarian *theme* Odoo, dan pengerjaan *Landing Page Corporate*.
16	17-22 Nov 2025	Integrasi data modul *TryOn Doors* ke *Flutter App*, perbaikan fitur *Reshape* dan *auto-fit* objek pintu, konfigurasi modul Odoo *TryOn* menjadi *Web Portal*, perbaikan *UI* modul Odoo *TryOn*, dan pembuatan surat serah terima barang.

Bersambung ke halaman berikutnya

Tabel 3.1. Uraian Kegiatan Magang per Minggu (Minggu 1-20) (lanjutan)

Minggu	Tanggal	Pekerjaan yang dilakukan
17	24-29 Nov 2025	Perbaikan fitur *Sales Promotor Flutter App* (*Search Product*, *Filtering* Tahun), integrasi data Odoo *TryOn Doors* ke *Flutter App*, modifikasi fitur *Reshape* (bentuk sudut, *hide* objek pintu), dan pengembangan *Landing Page Corporate*.
18	1-6 Des 2025	Pengembangan *Landing Page Corporate*, perbaikan fitur *saving* gambar dan *Reshape* modul *Web TryOn Doors* (paginasi, *fix error load*), pembuatan *UI Theme*, penginputan data gambar pintu, dan perbaikan *view* garis artifak.
19	8-13 Des 2025	Modifikasi fitur *CRM Pipeline* (*stage table view*, tampilan *stage won/lost*), penyelesaian *Landing Page Corporate*, perbaikan *bug load data Pipeline*, *fix UI Pie Chart*, dan konfigurasi *API* Odoo *Sales Promotor* ke *Excel* menggunakan *Power Query*.
20	15-17 Des 2025	Modifikasi fitur input *Pipeline* (*stage table*, indikasi *staging*), pembuatan *filtering* produk untuk modul *Pipeline* di Odoo, modifikasi *UI* modul *Web TryOn Doors*, dan konfigurasi *Power Query Excel* untuk data *CRM Pipeline* Odoo.

3.4 Perangkat Lunak dan Perangkat Keras yang Digunakan

Selama pelaksanaan magang, digunakan sejumlah perangkat lunak dan perangkat keras untuk mendukung proses perancangan, pengembangan, integrasi, serta pengujian aplikasi *mobile cross-platform* dan sistem Odoo ERP. Perangkat-perangkat tersebut meliputi *tools* pengembangan aplikasi, lingkungan pengujian, serta perangkat pendukung lainnya yang digunakan secara langsung dalam kegiatan magang. Penjelasan mengenai perangkat lunak dan perangkat keras tersebut disajikan sebagai berikut:

3.4.1 Perangkat Lunak

Tabel 3.2. Daftar Perangkat Lunak yang Digunakan

No.	Perangkat Lunak	Fungsi
1	Visual Studio Code	Digunakan sebagai *Integrated Development Environment* (*IDE*) utama dalam proses pengembangan aplikasi *Flutter* dengan dukungan ekstensi *Flutter*, *Dart*, dan *Android iOS Emulator*.
2	Flutter SDK	Digunakan sebagai *framework* utama untuk pengembangan aplikasi *mobile cross-platform* yang dapat berjalan di *platform Android* dan *iOS*.
3	Android Studio	Digunakan sebagai *Android SDK manager* dan penyedia *tools* untuk emulator serta *debugging* aplikasi *mobile*.
4	Odoo ERP versi 16.0 (*Production*)	Digunakan sebagai sistem produksi milik perusahaan untuk referensi*business logic*, integrasi *API* dengan aplikasi *Flutter*, dan akses data operasional.
5	Odoo ERP versi 16.0 (*Development*)	Digunakan sebagai *environment development* yang di-*hosting* di *Linux Debian Server* untuk pengembangan, kustomisasi modul, dan pengujian fitur sebelum di-*deploy* ke *production*.
6	VirtualBox	Digunakan untuk virtualisasi dan instalasi *Linux Debian Server* sebagai *environment* Odoo ERP.
7	GitHub	Digunakan sebagai platform penyimpanan repositori proyek.
8	Postman	Digunakan sebagai *tools* untuk pengujian *API JSON-RPC* Odoo dan *debugging*.

3.4.2 Perangkat Keras

Tabel 3.3. Daftar Perangkat Keras yang Digunakan

No.	Perangkat Keras	Fungsi
1	Laptop (Lenovo IdeaPad Gaming 3)	Digunakan sebagai perangkat utama untuk pengembangan aplikasi *mobile Flutter* dan konfigurasi sistem Odoo ERP dengan spesifikasi Processor Intel Core i5-10300H @ 2.50 GHz, RAM 16 GB DDR4, dan Operating System Windows 11 Home Single Language 64-bit.
2	Smartphone Vivo Y27s	Digunakan untuk pengujian aplikasi *mobile* dalam kondisi *real device* dan memastikan kompatibilitas aplikasi pada platform *Android*.

3.5 Proses Pelaksanaan Kerja Magang

Enterprise Resource Planning (*ERP*) perluasan dari konsep perencanaan sumber daya sebelumnya yaitu *Material Resources Planning* (*MRP II*) yang berfungsi sebagai kerangka kerja untuk mengatur, mendefinisikan, dan menstandarisasi proses bisnis yang diperlukan untuk merencanakan dan mengontrol semua sumber daya di seluruh perusahaan, termasuk keuangan, sumber daya manusia, dan manajemen, demi mencari keunggulan eksternal [5]. Odoo yang sebelumnya dikenal sebagai *TinyERP* atau *OpenERP* merupakan alternatif *cloud computing* (*SaaS*) dan sistem perencanaan sumber daya perusahaan berbasis *open-source* dengan struktur modular yang berfokus pada pengoptimalan operasional dan hubungan pelanggan [1].

Berdasarkan pemahaman tersebut, proses pelaksanaan kerja magang di PT JBS Perkasa dalam mengembangkan dan mengintegrasikan modul Odoo ERP menggunakan pendekatan *Software Development Life Cycle* (*SDLC*) sebagai kerangka kerja agar kegiatan dapat berjalan secara terstruktur, terarah, dan efektif. Dalam pendekatan *SDLC* tersebut, metodologi yang diterapkan adalah *Agile*. Keunggulan utama *Agile* terletak pada kemampuannya untuk diterapkan secara berkala dan iteratif, memungkinkan proses pengembangan yang fleksibel, responsif terhadap perubahan, serta didukung umpan balik dari berbagai pemangku kepentingan [6]. Pendekatan *Agile* bersifat adaptif sehingga dapat diterapkan baik dalam perancangan dan pengembangan aplikasi *mobile* maupun dalam pengembangan dan integrasi modul pada Odoo ERP, serta memungkinkan

penyesuaian selama proses berlangsung tanpa mengganggu keselarasan dan efisiensi pengembangan.

3.5.1 Perencanaan Proyek: Identifikasi Kebutuhan Bisnis dan Penentuan Fitur Aplikasi Mobile

Pengerjaan aplikasi *mobile cross-platform* diawali dengan diskusi dan penjelasan dari *supervisor* mengenai latar belakang dan tujuan pengembangan aplikasi tersebut. Dalam penjelasan tersebut, disampaikan bahwa aplikasi ini dirancang untuk mengakses sejumlah modul dan fitur dari sistem Odoo ERP perusahaan Fortress yang akan diintegrasikan ke dalam bentuk aplikasi *mobile*. Modul-modul yang menjadi fokus antara lain *Sales*, *Sales Promotor* (sebagai modul *add-on* khusus perusahaan), *CRM*, serta modul pendukung lainnya yang relevan dengan kebutuhan operasional tim lapangan.

Selain itu, ditekankan pula bahwa tim lapangan khususnya tim *sales* memerlukan antarmuka pengguna yang lebih sederhana dan intuitif dibandingkan akses melalui tampilan web. Hal ini bertujuan memudahkan penggunaan aplikasi di perangkat *mobile* tanpa mengalami hambatan navigasi atau keterbatasan responsivitas. Berdasarkan kebutuhan tersebut, *supervisor* memberikan arahan untuk menggunakan *Flutter* sebagai *framework* pengembangan aplikasi *cross-platform*, dengan integrasi langsung ke sistem Odoo perusahaan. Pendekatan ini tidak hanya mempertimbangkan aspek antarmuka, tetapi juga konsistensi dan sinkronisasi data antara aplikasi *mobile* dan sistem Odoo ERP perusahaan.

3.5.2 Perancangan Sistem: Diagram UML dan Mockup UI

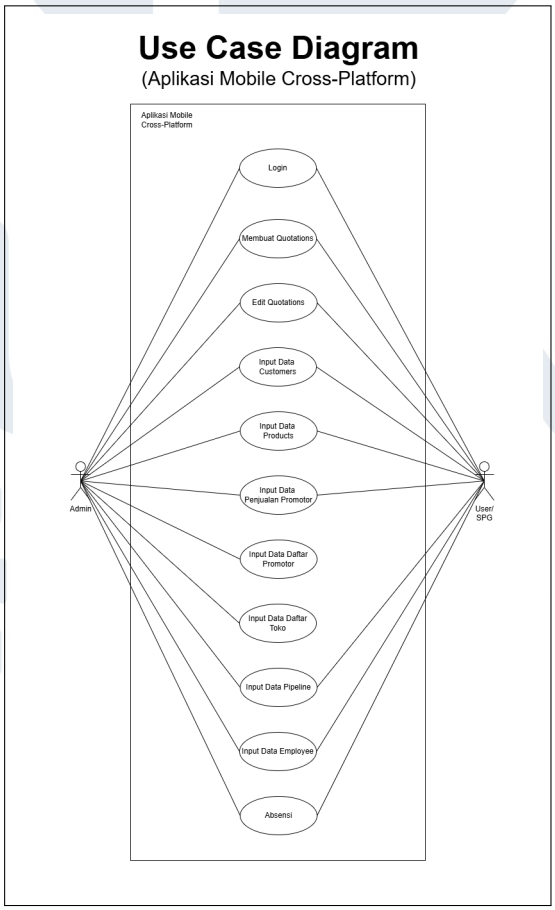
Tahap perancangan dalam pengembangan aplikasi *mobile cross-platform* dilakukan untuk mengvisualisasikan alur kerja sistem pada setiap modul dan fitur, sehingga aplikasi dapat berjalan secara efektif dan efisien. Salah satu keunggulan penggunaan diagram *UML* adalah kemampuannya dalam mendokumentasikan arsitektur dan proses sistem secara terstruktur dan terstandarisasi, sehingga mempermudah komunikasi dan pelaksanaan selama proses pengembangan [7]. Pada tahap ini, dikembangkan *use case diagram* untuk menggambarkan interaksi pengguna dengan sistem, serta *activity diagram* dan *sequence diagram* yang memperlihatkan alur proses dari masing-masing *use case*. Selain itu, juga dibuat mockup antarmuka pengguna (*UI*) untuk memvisualisasikan tampilan dan navigasi

aplikasi pada setiap modul dan fitur yang dikembangkan.

A Use Case Diagram Aplikasi Mobile Cross-Platform

Gambar 3.2 merupakan *Use Case Diagram* yang menggambarkan interaksi pengguna dengan aplikasi *mobile cross-platform*. Aplikasi dirancang untuk menyesuaikan alur kerja pengguna dalam berinteraksi dengan sistem Odoo, sekaligus mempertimbangkan penerapan *Record Rules* yang mengatur siapa saja yang berhak mengakses atau memodifikasi data tertentu. Hal ini memastikan bahwa fitur-fitur tertentu hanya tersedia bagi pengguna yang berwenang.

Dalam diagram tersebut terlihat bahwa *User Sales* tidak dapat mengakses beberapa *use case* tertentu, seperti Input Data Daftar Promotor, Input Data Daftar Toko yang secara eksplisit diperuntukkan bagi *User Admin*. Pembatasan ini mencerminkan penerapan prinsip *role-based access control* dalam sistem. Setiap pengguna hanya dapat menjalankan fungsi yang sesuai dengan tanggung jawabnya.

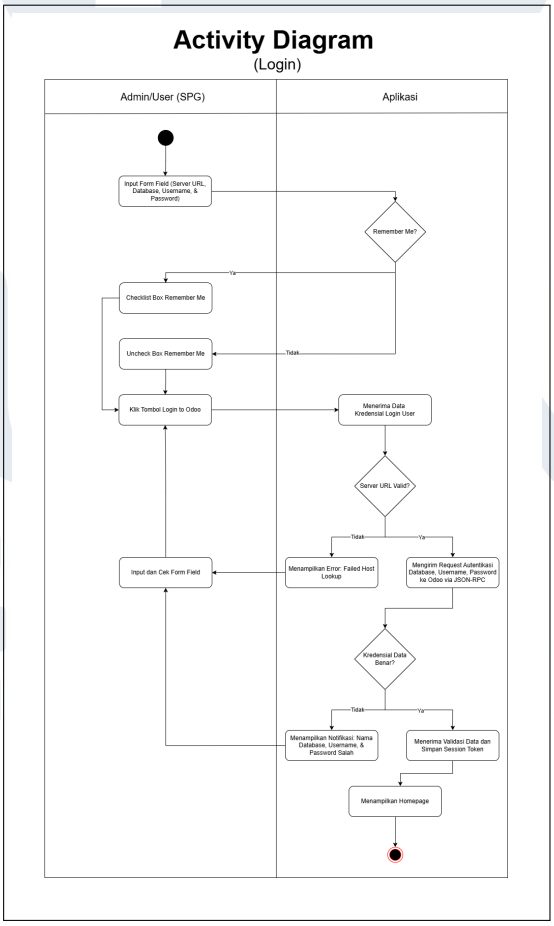


Gambar 3.2. Use Case Diagram Aplikasi Mobile Cross-Platform

B Activity Diagram Login

Gambar 3.3 menampilkan *Activity Diagram* yang menggambarkan alur interaksi pengguna dengan aplikasi. Pada diagram tersebut dijelaskan proses yang dimulai ketika pengguna memasukkan data yang diperlukan, seperti nama *database*, *username*, dan *password*. Sebelum melanjutkan, sistem memvalidasi apakah *server URL* yang dimasukkan diawali dengan protokol *http* atau *https*.

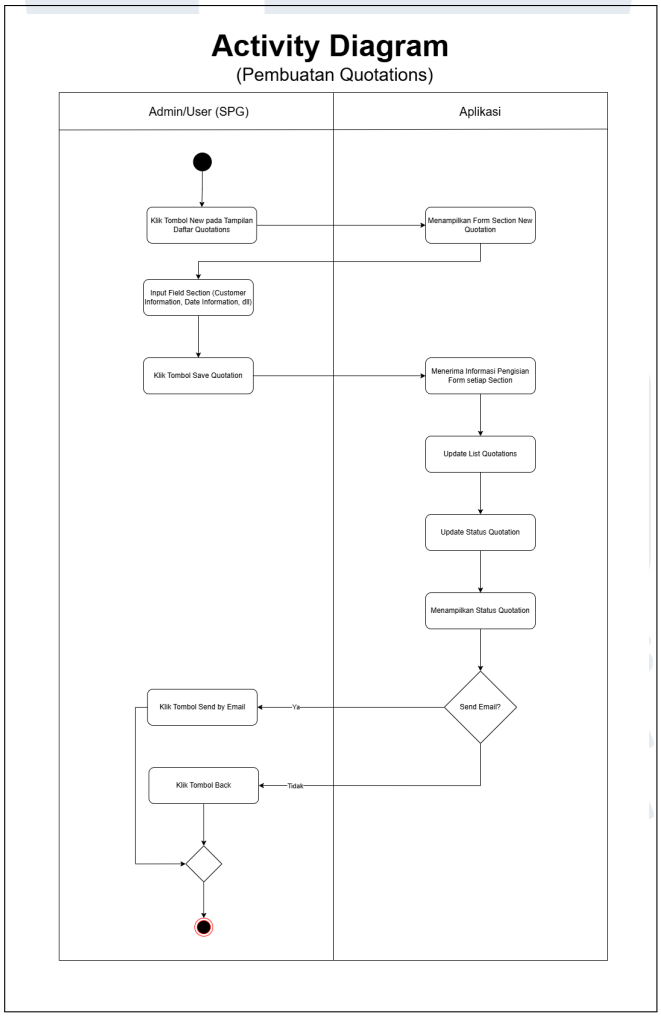
Proses validasi alamat *server* penting karena aplikasi hanya dapat melakukan koneksi dan mengarahkan pengguna ke halaman beranda apabila format alamat *server* memenuhi persyaratan tersebut. Pengguna yang tidak mencantumkan protokol (*http*/*https*) atau memasukkan alamat *server* yang tidak valid, maka sistem akan menampilkan notifikasi kesalahan berupa "*failed host lookup*". Proses ini memastikan bahwa kredensial dan konfigurasi koneksi yang dimasukkan oleh pengguna sesuai sebelum proses autentikasi dan akses ke sistem dilanjutkan.



Gambar 3.3. Activity Diagram Login Aplikasi

C Activity Diagram Pembuatan Quotations (Modul Sales)

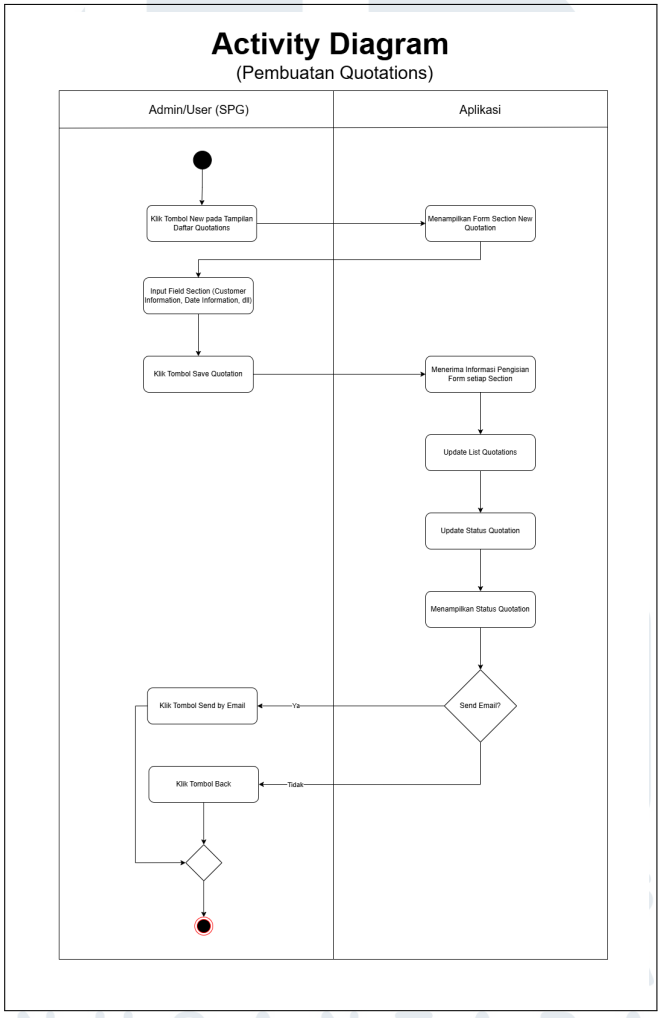
Gambar 3.4 menampilkan *Activity Diagram* yang menggambarkan alur interaksi *user* dengan aplikasi saat memilih menu modul *Sales* pada halaman beranda. Setelah memilih modul tersebut, pengguna diarahkan ke submenu *Quotations*, di mana ia dapat membuat penawaran baru. *Quotation* yang telah dibuat akan ditambahkan ke dalam daftar *Quotations* yang tersedia dalam sistem. Proses ini dirancang agar dapat dilanjutkan ke tahap berikutnya, yaitu konversi menjadi *Sales Order*, sehingga mendukung kelangsungan alur penjualan. Selain itu, pengguna juga memiliki opsi untuk mengirimkan *quotation* tersebut melalui email kepada pihak yang dituju, memungkinkan penerima untuk melihat detail penawaran secara langsung.



Gambar 3.4. Activity Diagram Pembuatan Quotations (Modul Sales)

D Activity Diagram Edit Quotations (Modul Sales)

Gambar 3.5 menampilkan *Activity Diagram* yang menggambarkan proses ketika *user* mengedit dan mengubah status *Quotation* menjadi *Sales Order*. *User* dapat memilih *Quotation* dari daftar, lalu melakukan perubahan pada berbagai *field* di masing-masing *form section*. Setelah pengeditan selesai, *user* dapat mengonversi *Quotation* tersebut menjadi *Sales Order* untuk melanjutkan alur penjualan.

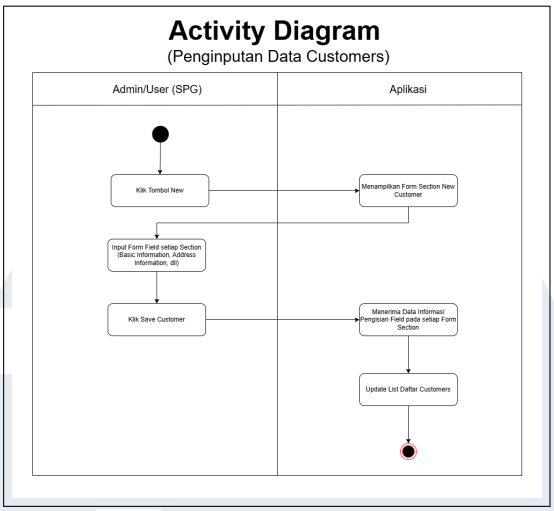


Gambar 3.5. Activity Diagram Edit Quotations (Modul Sales)

E Activity Diagram Penginputan Data Customers (Modul Sales)

Gambar 3.6 menampilkan *Activity Diagram* yang menggambarkan proses ketika *user* ingin menginput data *Customers*. *User* memasukkan informasi pelanggan yang mencakup *Basic Information*, *Address*, dan data pendukung

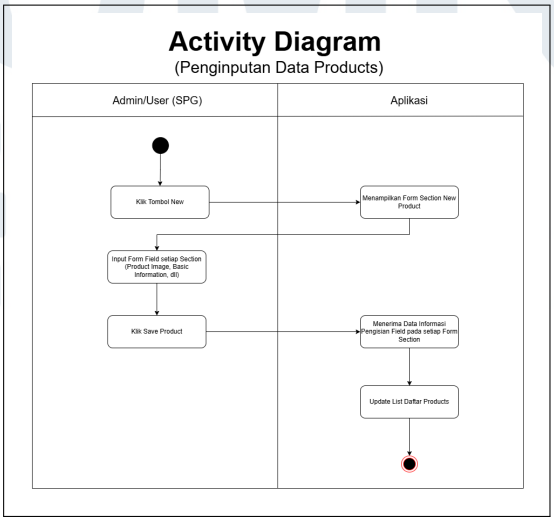
lainnya. Setelah proses input selesai, data tersebut akan tersimpan dan muncul dalam list daftar *Customers*.



Gambar 3.6. Activity Diagram Penginputan Data Customers (Modul Sales)

F Activity Diagram Penginputan Data Products (Modul Sales)

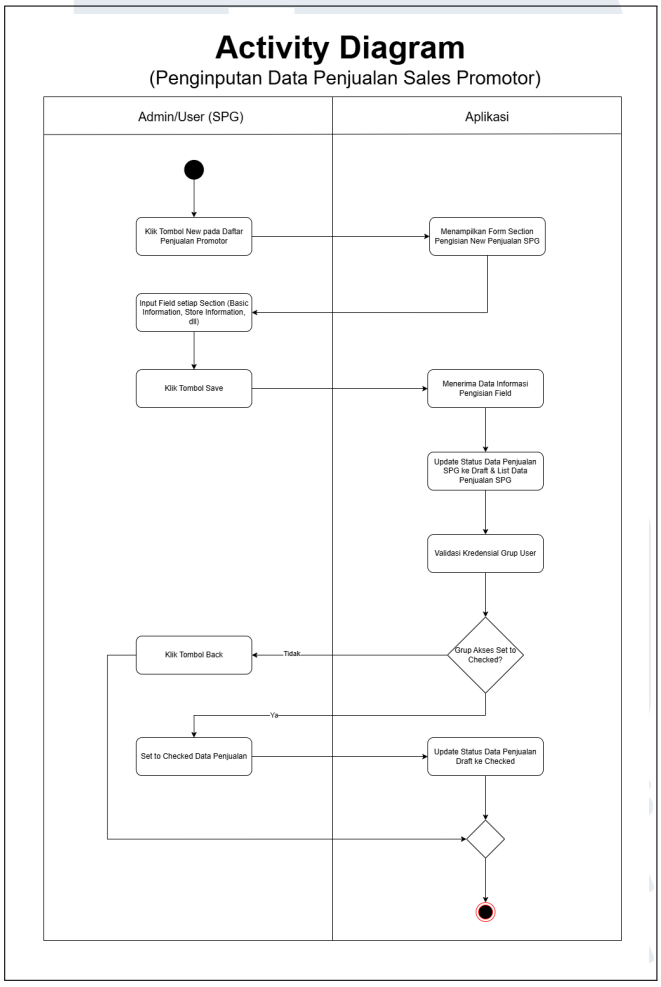
Gambar 3.7 menampilkan *Activity Diagram* yang menggambarkan proses ketika *user* menginput data *Products*. *User* memilih menu *Products* pada modul *Sales*, kemudian mengklik tombol *New Products* dan mengisi *form section New Products* yang mencakup *Basic Information* serta data pendukung lainnya. Setelah disimpan, data *Products* tersebut akan muncul dalam list daftar *Products*.



Gambar 3.7. Activity Diagram Penginputan Data Products (Modul Sales)

G Activity Diagram Penginputan Data Penjualan (Modul Sales Promotor)

Gambar 3.8 menampilkan *Activity Diagram* yang menggambarkan proses ketika *user* masuk ke modul *Sales Promotor* dan memilih menu Penjualan. *User* kemudian mengklik tombol *New* untuk mengisi *form section New* Penjualan SPG, yang mencakup *Store Information* dan data pendukung lainnya. Sebelum menampilkan tombol *Set To Checked* pada tampilan *New* Penjualan SPG, sistem akan memeriksa *Record Rules* di Odoo untuk memastikan apakah *user* termasuk dalam grup akses yang berhak menggunakan tombol tersebut.

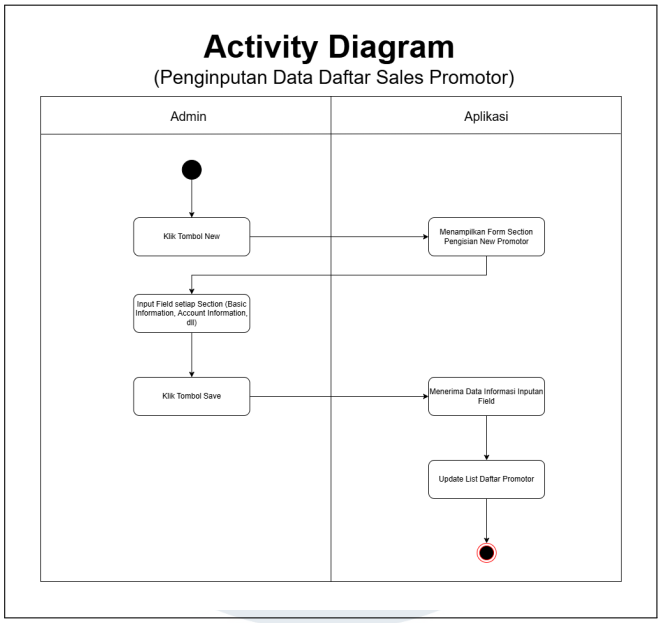


Gambar 3.8. Activity Diagram Penginputan Data Penjualan (Modul Sales Promotor)

H Activity Diagram Penginputan Data Promotor (Modul Sales Promotor)

Gambar 3.9 menampilkan *Activity Diagram* yang menggambarkan proses ketika *user* melakukan input data pada daftar *Sales Promotor*. *User* terlebih dahulu

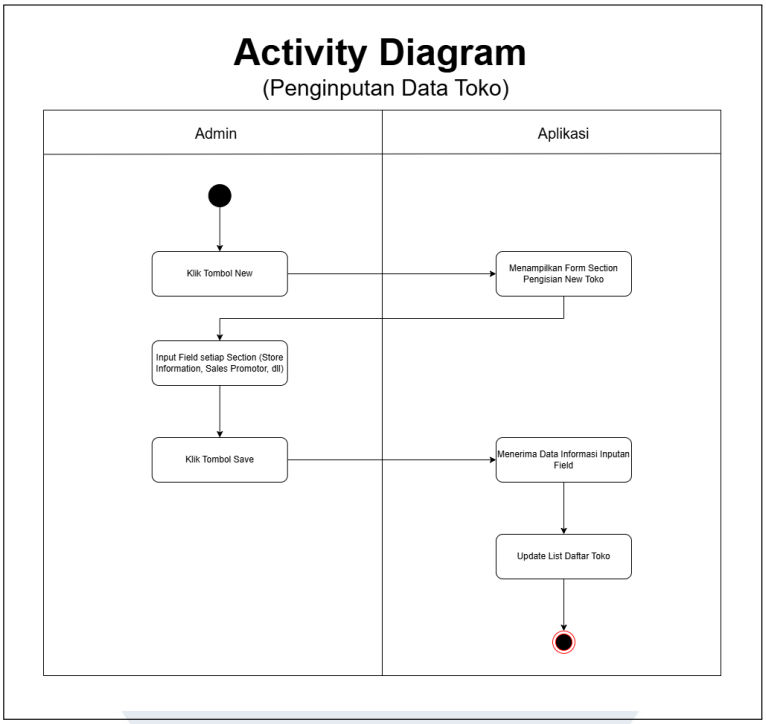
memasuki menu Promotor pada modul *Sales Promotor* untuk mengakses tampilan daftar promotor. Selanjutnya, *user* membuat entri baru melalui *View New Promotor* dan mengisi sejumlah *field* pada *form section*, seperti *Account Information* dan data pendukung lainnya.



Gambar 3.9. Activity Diagram Penginputan Data Promotor (Modul Sales Promotor)

I Activity Diagram Penginputan Data Toko (Modul Sales Promotor)

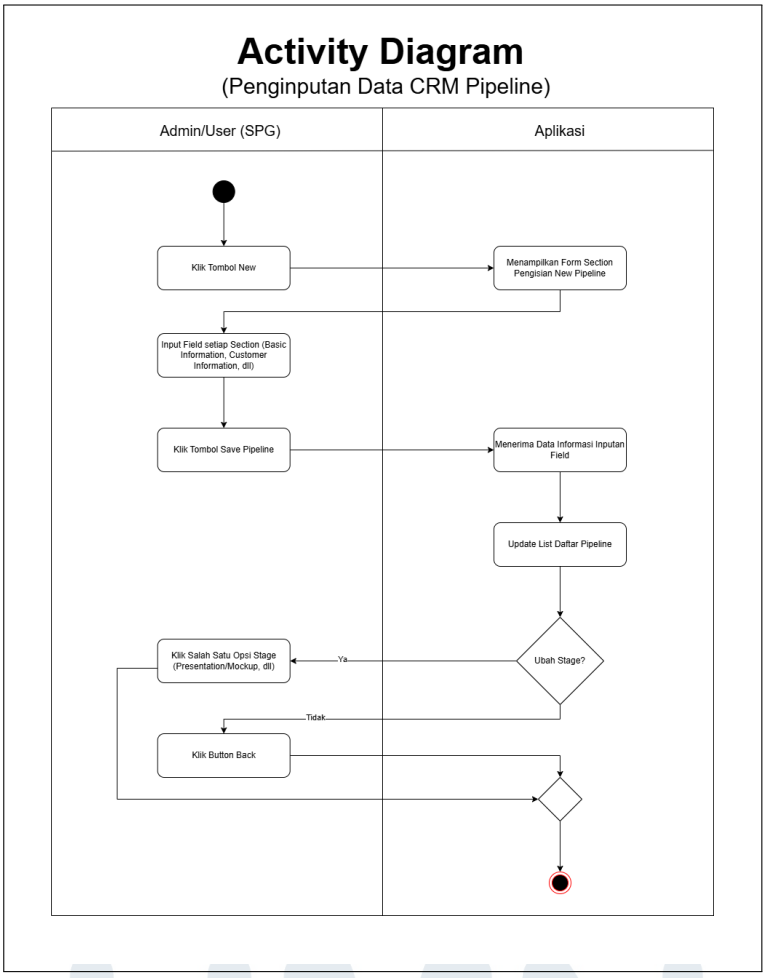
Gambar 3.10 menampilkan *Activity Diagram* yang menggambarkan proses ketika *user* melakukan penginputan data toko pada daftar toko. *User* harus memasuki menu Toko pada modul *Sales Promotor*, yang keberadaannya dikendalikan oleh *Record Rules* di *Odoo* sehingga hanya muncul bagi grup akses tertentu khususnya admin. Proses penginputan data dilakukan melalui pengisian *form section* standar, serupa dengan cara pengisian *form* pada menu-menu lainnya.



Gambar 3.10. Activity Diagram Penginputan Data Toko (Modul Sales Promotor)

J Activity Diagram Penginputan Data CRM Pipeline (Modul CRM)

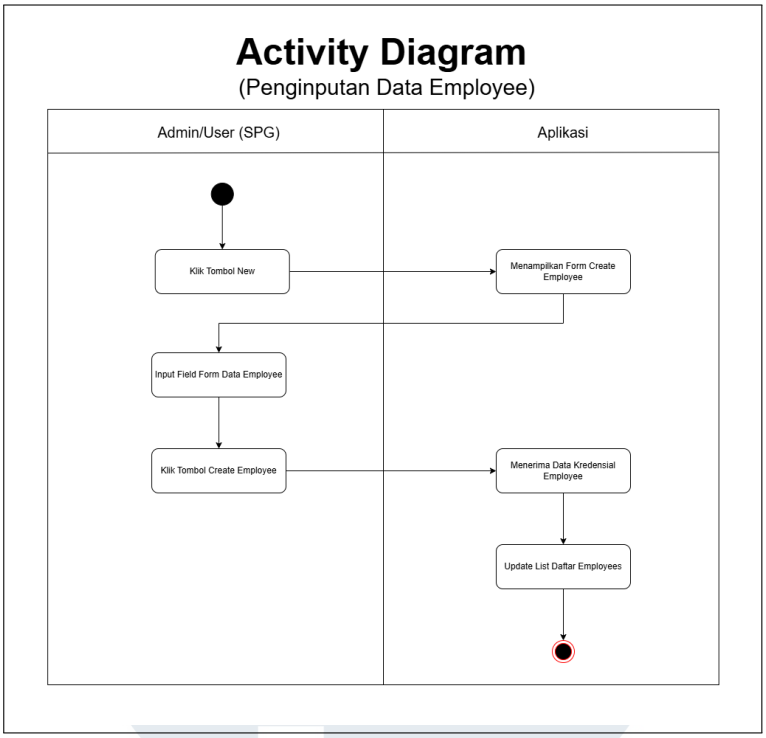
Gambar 3.11 menampilkan *Activity Diagram* yang menggambarkan proses ketika *user* melakukan penginputan data *CRM MyPipeline*. *User* memasuki modul *CRM* dan memilih menu *MyPipeline*, lalu membuat entri baru pada daftar *pipeline* dengan mengisi beberapa *form section*, seperti *Customer Information* dan data pendukung lainnya. Selama proses tersebut, *user* juga dapat mengubah *stage* sesuai dengan perkembangan prospek dalam alur penjualan.



Gambar 3.11. Activity Diagram Penginputan Data CRM Pipeline (Modul CRM)

K Activity Diagram Penginputan Data Employee (Modul Employees)

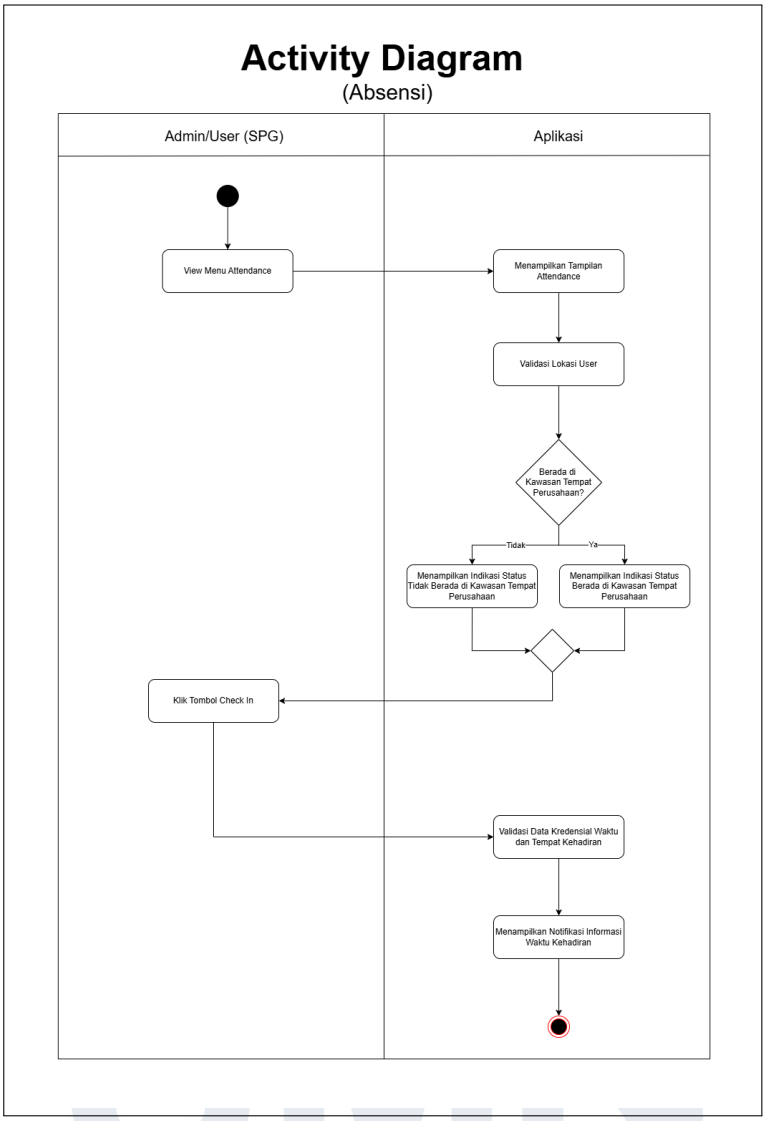
Gambar 3.12 menampilkan *Activity Diagram* yang menggambarkan proses ketika *user* ingin menginput data *Employee*. *User* terlebih dahulu memasuki modul *Employees* dari halaman beranda, kemudian mengisi sejumlah *field* yang diperlukan dalam formulir data *Employee*. Setelah data dikirim, kredensial divalidasi dan informasi *Employee* tersebut disinkronkan ke dalam sistem Odoo



Gambar 3.12. Activity Diagram Penginputan Data Employee (Modul Employees)

L Activity Diagram Absensi (Menu Attendances)

Gambar 3.13 menampilkan *Activity Diagram* yang menggambarkan proses absensi yang dilakukan oleh *user* atau *employee*. Ketika *user* membuka menu *Attendances*, sistem menampilkan tampilan berisi tombol *Check In* dan status box yang menunjukkan kondisi kehadiran. Sebelum proses *Check In* berhasil, sistem memvalidasi lokasi *user* untuk memastikan bahwa *user* berada di kawasan yang telah ditetapkan oleh perusahaan, sesuai dengan konfigurasi grup akses dalam integrasi Odoo. Jika validasi lokasi berhasil, *user* akan menerima notifikasi yang menampilkan waktu kehadirannya.

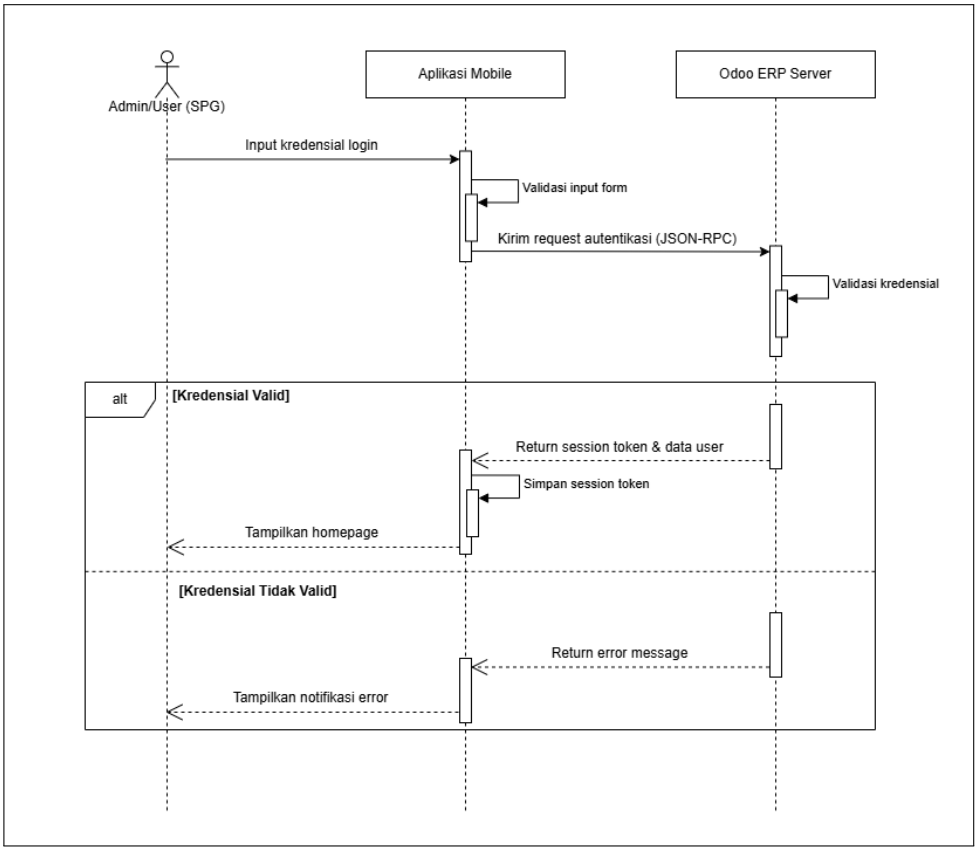


Gambar 3.13. Activity Diagram Absensi (Menu Attendances)

M Sequence Diagram Login

Gambar 3.14 menampilkan *sequence diagram* yang menggambarkan alur komunikasi antara pengguna (*Admin*/*User* yang telah *login*) dengan sistem Odoo ERP selama proses *login* melalui aplikasi *mobile*. Pengguna memasukkan kredensial seperti nama *database*, *username*, dan *password*, melalui antarmuka aplikasi. Aplikasi kemudian memvalidasi input tersebut dan mengirim permintaan autentikasi ke *endpoint*/*jsonrpc* menggunakan protokol *JSON-RPC*. Diagram ini juga mencakup alur alternatif untuk membedakan skenario *login* berhasil dan *login* gagal, sehingga memastikan struktur alur kerja sistem tetap jelas dan terkelola

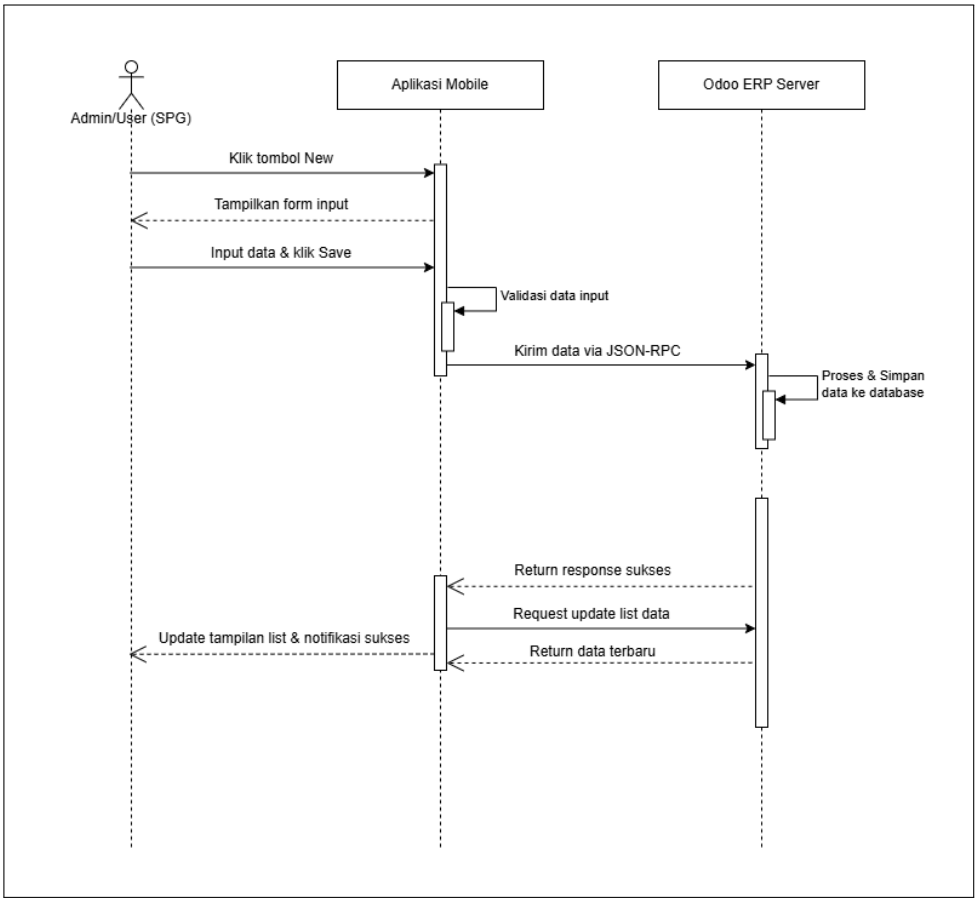
dengan baik.



Gambar 3.14. Sequence Diagram Login

N Sequence Diagram Input Master Data

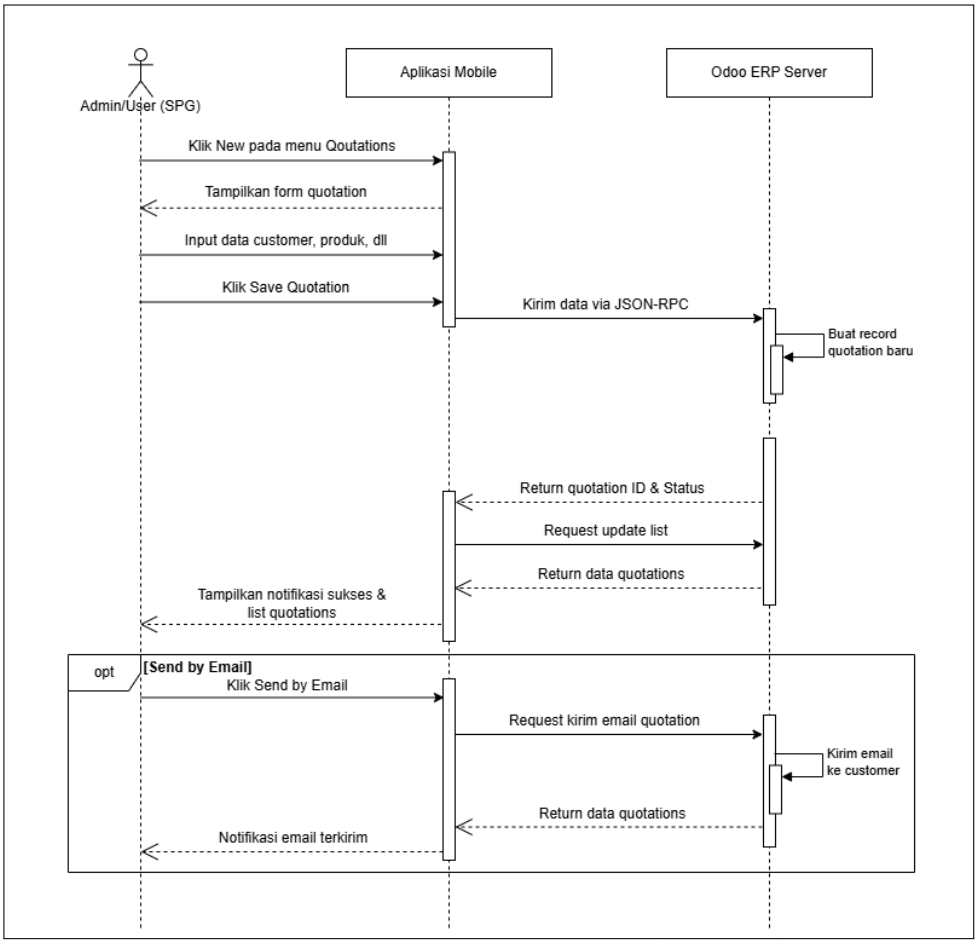
Gambar 3.15 menampilkan *sequence diagram* yang menggambarkan alur penginputan data *master* (misalnya: karyawan, pelanggan, atau produk) oleh pengguna (*Admin*/SPG) melalui aplikasi *mobile*.Proses dimulai ketika pengguna menekan tombol "*New*", yang memicu tampilan *form input* di aplikasi. Setelah pengguna memasukkan data dan menekan "*Save*", sistem melakukan validasi lokal terlebih dahulu sebelum mengirimkan data ke *server* Odoo ERP melalui protokol *JSON-RPC*. *Server* kemudian memproses dan menyimpan data ke dalam *database*, lalu mengembalikan respons sukses.



Gambar 3.15. Sequence Diagram Input Master Data

O Sequence Diagram Pembuatan Quotation

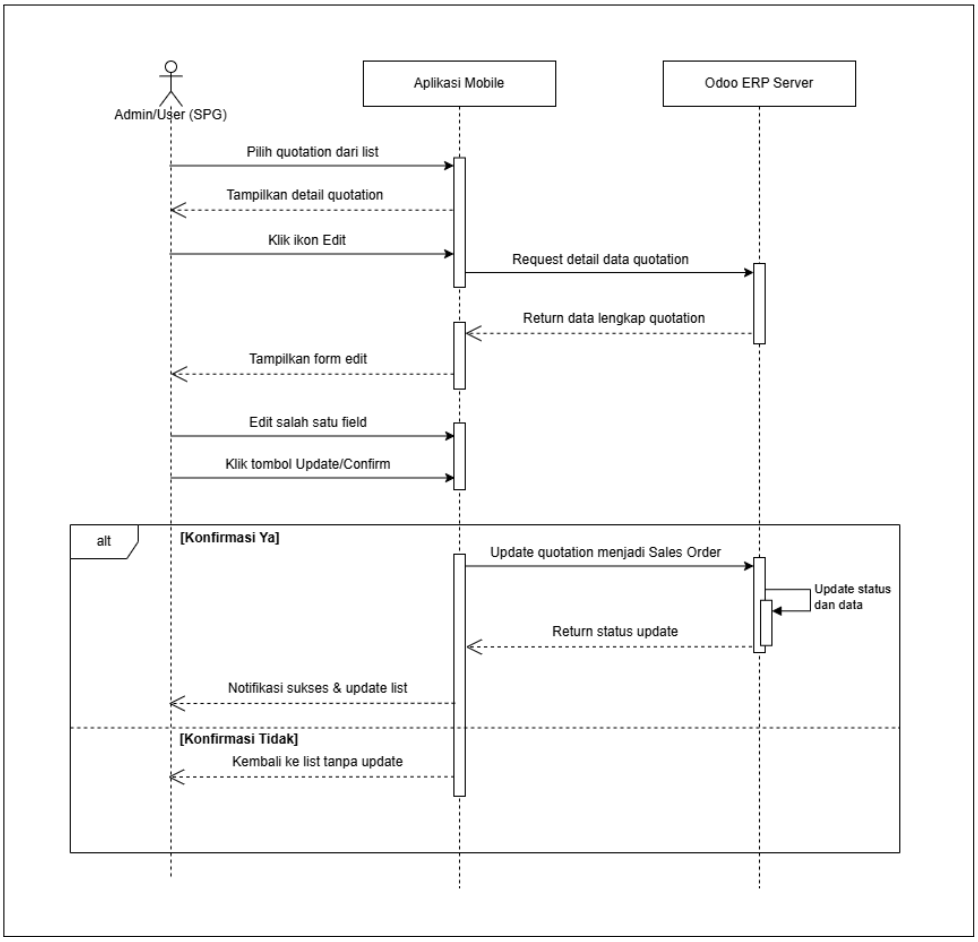
Gambar 3.16 menggambarkan *sequence diagram* yang menggambarkan alur kerja pengguna memulai pembuatan *quotation* melalui aplikasi *mobile* dengan mengisi form data pelanggan dan produk, lalu menyimpannya ke sistem Odoo ERP melalui protokol *JSON-RPC*. Sistem Odoo memproses dan menyimpan data tersebut sebagai *record quotation* baru, kemudian mengembalikan respons sukses ke aplikasi. Sebagai fitur opsional, pengguna dapat mengirim *quotation* tersebut *via email* langsung dari aplikasi, yang akan diproses oleh *server* Odoo dan dikonfirmasi dengan notifikasi pengiriman berhasil.



Gambar 3.16. Sequence Diagram Pembuatan Quotation

P Sequence Diagram Edit Quotation

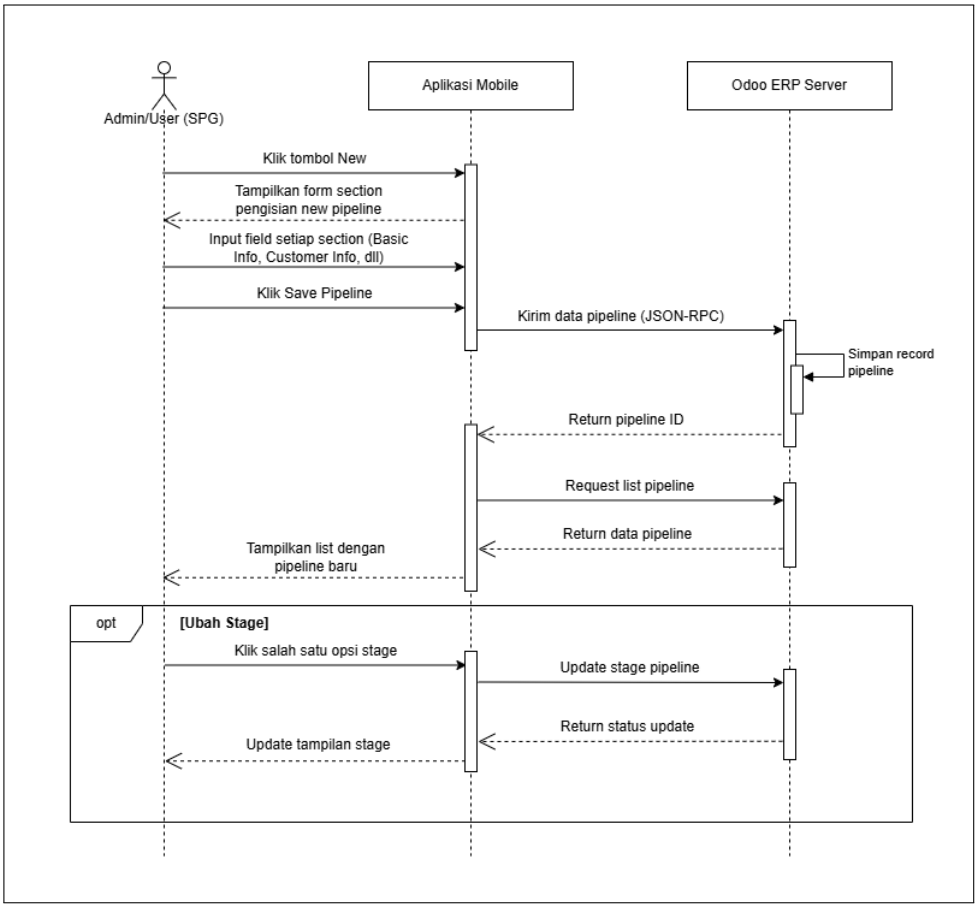
Gambar 3.17 menampilkan *sequence diagram* yang menggambarkan alur pengguna (*Admin*/SPG) dalam melakukan proses edit dan konfirmasi perubahan pada data *quotation* melalui aplikasi *mobile*. Pengguna memilih *quotation* dari daftar, membuka detailnya, lalu mengklik tombol edit untuk memperbarui satu atau beberapa *field* sebelum menyimpan perubahan ke *server* Odoo ERP melalui *JSON-RPC*. Setelah perubahan disimpan, sistem menampilkan opsi konfirmasi: jika dikonfirmasi "Ya", *quotation* akan diperbarui menjadi *Sales Order*, dan apabila "Tidak", data kembali ke tampilan *list* tanpa perubahan.



Gambar 3.17. Sequence Diagram Edit Quotation

Q Sequence Diagram Input Data CRM Pipeline

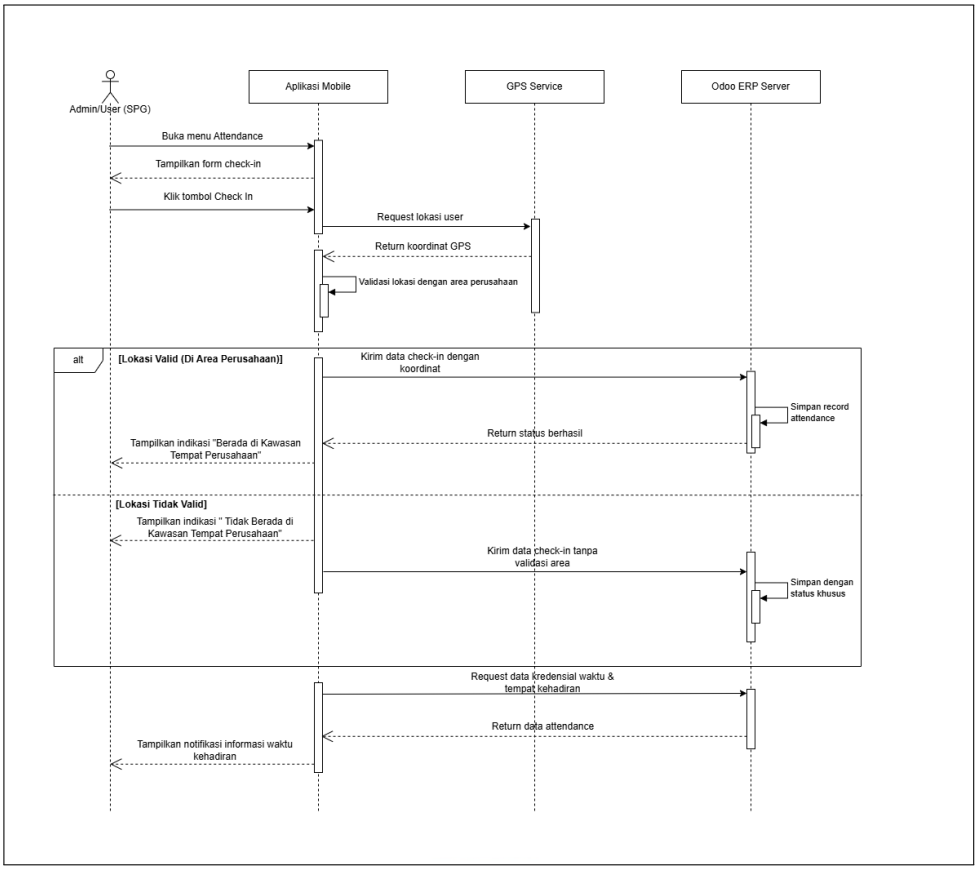
Gambar 3.18 menampilkan *sequence diagram* yang menggambarkan alur pengguna (*Admin*/SPG) dalam membuat dan memperbarui *pipeline* baru melalui aplikasi *mobile*. Pengguna memulai proses dengan menekan tombol "*New*", mengisi form data dasar seperti *Basic Info* dan *Customer Info*, lalu menyimpannya ke *server* Odoo ERP melalui *JSON-RPC* untuk menciptakan *record pipeline* baru. Sebagai fitur opsional, pengguna dapat memilih untuk mengubah *stage pipeline*, yang akan memicu *update* status di *server* dan memperbarui tampilan *list* secara *real-time*.



Gambar 3.18. Sequence Diagram Input Data CRM Pipeline

R Sequence Diagram Absensi

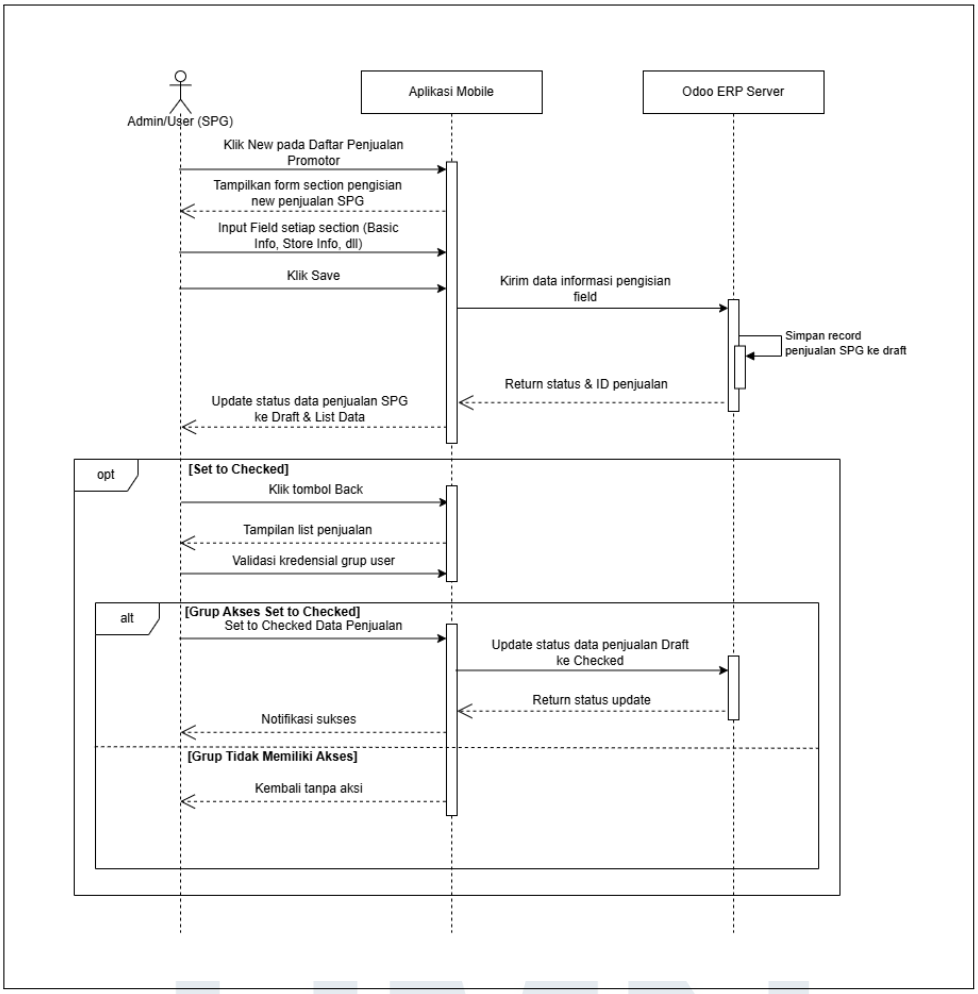
Gambar 3.19 menampilkan *sequence diagram* yang menggambarkan alur proses absensi (*check-in*) oleh pengguna (*Admin*/SPG) melalui aplikasi *mobile*, yang terintegrasi dengan layanan *GPS* dan sistem Odoo ERP. Pengguna membuka menu *Attendance*, lalu menekan tombol "*Check In*", aplikasi akan meminta koordinat lokasi dari *GPS Service*, lalu memvalidasinya dengan area perusahaan sebelum mengirim data ke *server* Odoo. Jika lokasi valid, data disimpan sebagai *record attendance* dengan status "berhasil", jika tidak, data tetap dikirimkan namun dengan status khusus, dan aplikasi menampilkan notifikasi informasi waktu serta lokasi kehadiran.



Gambar 3.19. Sequence Diagram Absensi

S Sequence Diagram Input Data Penjualan Sales Promotor

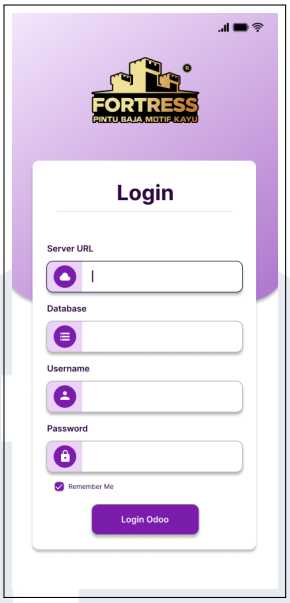
Gambar 3.20 menampilkan *sequence diagram* yang menggambarkan alur pengguna (*Admin*/SPG) dalam membuat dan memvalidasi data penjualan promotor melalui aplikasi *mobile*, yang terintegrasi dengan sistem Odoo ERP. Pengguna memulai proses dengan mengisi *form input* untuk data penjualan baru, lalu menyimpannya ke *server*, sistem akan menyimpan *record* sebagai status *draft* dan mengembalikan *ID* serta notifikasi sukses. Sebagai fitur opsional, pengguna dapat memilih "*Set to Checked*", yang akan memicu validasi kredensial grup *user*, dan jika pengguna memiliki akses, status data akan diperbarui menjadi *checked*, sebaliknya akan dikembalikan tanpa perubahan.



Gambar 3.20. Sequence Diagram Input Data Sales Promotor

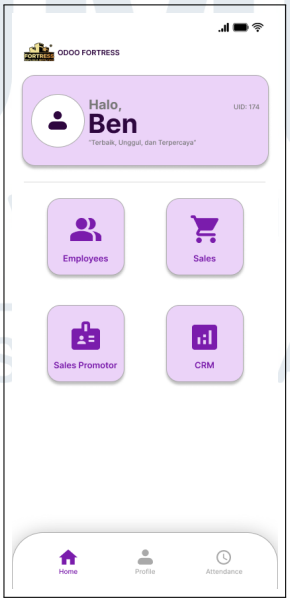
T Mockup UI Aplikasi

Pembuatan *Mockup UI* merupakan salah satu tahapan penting yang dilakukan setelah penyusunan *UML Diagram*, bertujuan untuk memberikan gambaran visual mengenai tata letak dan interaksi antarmuka pengguna sesuai dengan alur fitur yang telah dirancang. *Mockup* ini berfungsi sebagai panduan dalam implementasi *UI* saat pengembangan aplikasi, sehingga memudahkan pengembang dalam memahami struktur dan alur penggunaan aplikasi. Berikut ini adalah beberapa contoh *Mockup UI* yang menggambarkan tampilan aplikasi yang direncanakan:



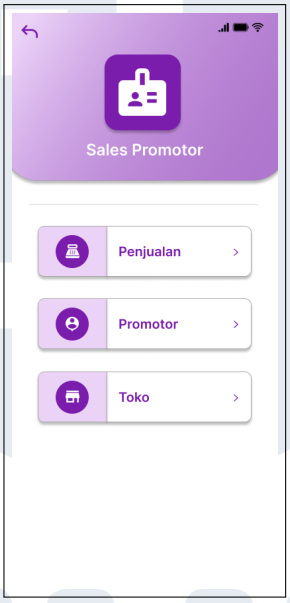
Gambar 3.21. Mockup UI Login

Gambar 3.21 menampilkan *Mockup UI* halaman awal aplikasi, yaitu tampilan *login*. Warna yang digunakan pada *mockup* ini bersifat sementara dan hanya bertujuan sebagai gambaran awal dan perubahan warna akan dilakukan setelah mendapat konfirmasi dari *supervisor*. Pada *mockup* tersebut terdapat kotak berisi sejumlah *field* input yang harus diisi user untuk melakukan autentikasi dan kemudian dialihkan ke halaman *homepage*.



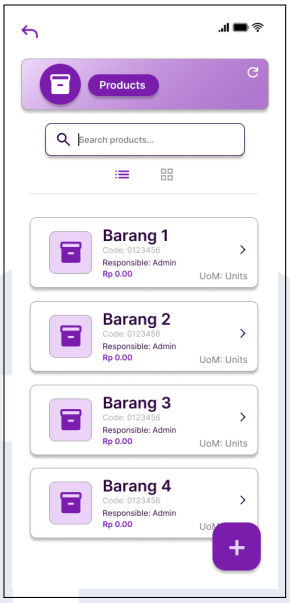
Gambar 3.22. Mockup UI Homepage

Gambar 3.22 menampilkan *Mockup UI* halaman *homepage*, yang muncul setelah *user* berhasil *login* melalui halaman pengisian data sebelumnya. Pada tampilan ini, *user* akan melihat daftar modul yang disajikan dalam bentuk kotak-kotak sederhana (*simplified*) sesuai permintaan tim *sales* agar tampilan lebih bersih dan mudah digunakan, serta menampilkan nama *username user*. Selain itu, terdapat pula *bottom navbar* yang menyediakan menu opsi tambahan untuk navigasi aplikasi.



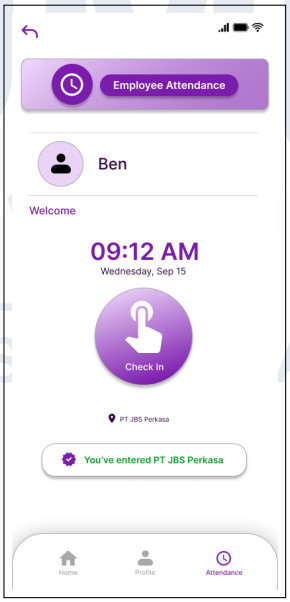
Gambar 3.23. Mockup UI Menu Modul

Gambar 3.23 menampilkan *Mockup UI* halaman yang muncul ketika *user* memilih salah satu modul dari tampilan *homepage*. Tampilan tersebut menggambarkan struktur umum menu-menu yang tersedia di dalam suatu modul. *Layout* ini konsisten digunakan untuk seluruh modul lainnya pada *homepage*, sehingga memberikan pengalaman navigasi yang seragam dan intuitif bagi *user*.



Gambar 3.24. Mockup UI Tampilan Umum dalam Menu

Gambar 3.24 menampilkan *Mockup UI* halaman ketika *user* masuk ke salah satu menu dalam modul tertentu. Tampilan ini bersifat representatif, di mana sebagian besar *layout* mengikuti pola yang sama, meskipun desain beberapa *item card* dapat berbeda menyesuaikan konteks datanya. Pada tampilan ini, *user* diberikan opsi untuk mengganti jenis tampilan, seperti *Grid View*, *Chart View*, maupun jenis *view* lainnya sesuai kebutuhan.



Gambar 3.25. Mockup UI Tampilan Attendances

Gambar 3.25 menampilkan *Mockup UI* halaman absensi pada menu *Attendances*. Tampilan ini menggambarkan antarmuka awal yang digunakan *user* untuk melakukan absensi melalui tombol *Check In*. Meskipun *user* tetap dapat melakukan *Check In* dari luar area perusahaan, kotak indikasi di bawah tombol tersebut akan memberikan informasi visual bahwa lokasi saat ini berada di luar kawasan yang ditetapkan. Setelah proses *Check In* selesai, sistem menampilkan notifikasi yang berisi waktu kehadiran *user*.

3.5.3 Implementasi Aplikasi Mobile Cross-Platform

A Implementasi Integrasi Sistem Odoo ERP

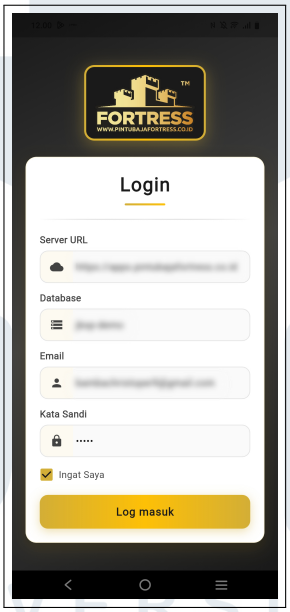
```
final url = Uri.parse("$serverUrl/jsonrpc");
final body = {
  "jsonrpc": "2.0",
  "method": "call",
  "params": {
    "service": "object",
    "method": "execute_kw",
    "args": [
      db, uid, password,
      "sale.order",         // nama model Odoo
      "search_read",        // method model
      [[]],                 // domain (semua data)
      {
        "fields": ["name", "partner_id", "amount_total"]
      }
    ]
  },
  "id": 1
};

final response = await http.post(
  url,
  headers: {"Content-Type": "application/json"},
  body: jsonEncode(body),
);
```

Kode 3.1: Potongan Kode Implementasi Integrasi JSON-RPC dengan Odoo ERP

Integrasi aplikasi mobile dengan sistem Odoo ERP dilakukan melalui protokol *JSON-RPC* seperti pada kode 3.1, yang merupakan antarmuka standar

untuk akses eksternal pada Odoo. *JSON-RPC* mendukung komunikasi ringan dan efisien untuk *RPC* (*Remote Procedure Call*) berbasis *JSON*, yang memungkinkan akses data secara langsung dan *real-time* dari *server* melalui *HTTP* atau *HTTPS* [8]. Aplikasi mengirim permintaan *HTTP POST* ke *endpoint* `/jsonrpc` dengan struktur body berformat JSON yang mencakup nama service (`object`), metode (`execute_kw`), serta parameter berupa kredensial sesi (database, UID, password), nama model (misal: `sale.order`), dan operasi yang diinginkan (`search_read`). Respons dari Odoo berupa data *JSON* yang kemudian diproses oleh aplikasi *mobile* untuk ditampilkan kepada pengguna, sehingga memungkinkan akses *real-time* terhadap data perusahaan langsung dari perangkat lapangan.

B Tampilan Login



Gambar 3.26. Tampilan Login

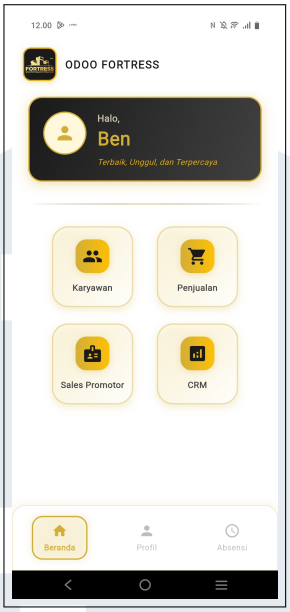
Gambar 3.26 menampilkan tampilan halaman *login* aplikasi *mobile*, di mana pengguna wajib mengisi *field*: *Server URL*, *Database*, *Email*, dan Kata Sandi. Data tersebut harus diverifikasi oleh pihak yang memiliki akses ke *server* perusahaan untuk memastikan keakuratan dan keamanan koneksi. Desain antarmuka menggunakan palet warna hitam, emas, dan kuning yang disesuaikan dengan identitas visual logo Fortress, sesuai persetujuan *supervisor*, demi menjaga konsistensi *branding*.

```dart
Future<void> _saveData() async {
  final prefs = await SharedPreferences.getInstance();
  prefs.setBool("rememberMe", _rememberMe);
  if (_rememberMe) {
    prefs.setString("odooUsername", _odooUsernameController.text);
    prefs.setString("odooPassword", _odooPasswordController.text);
  }
}

Future<void> _loadSavedData() async {
  final prefs = await SharedPreferences.getInstance();
  _rememberMe = prefs.getBool("rememberMe") ?? false;
  if (_rememberMe) {
    _odooUsernameController.text = prefs.getString("odooUsername") ?? "";
    _odooPasswordController.text = prefs.getString("odooPassword") ?? "";
  }
}
```

Kode 3.2: Potongan Kode Penyimpanan Kredensial

Fitur *login* aplikasi *mobile* dirancang untuk memudahkan pengguna dengan menyediakan opsi "*Remember Me*" yang secara otomatis memuat kembali data *login* sebelumnya menggunakan *SharedPreferences* seperti pada potongan Kode 3.2. *SharedPreferences*, yaitu sebagai salah satu metode untuk menyimpan data terkait penggunaan sumber daya perangkat, seperti waktu layar *on* dan informasi jaringan [9]. Saat pengguna mencentang opsi ini, semua kredensial (*server URL*, *database*, *username*, *password*) disimpan dalam format teks terenkripsi, dan akan dimuat ulang saat aplikasi dibuka kembali.

C Tampilan Homepage



Gambar 3.27. Tampilan Homepage

Gambar 3.27 menampilkan tampilan *homepage* aplikasi setelah pengguna berhasil *login*. Tampilan ini dirancang sesuai dengan *mockup UI* yang telah ditentukan sebelumnya, dengan susunan modul yang disusun secara presisi dalam bentuk *grid 2×2* untuk memudahkan navigasi. Pada halaman ini, pengguna dapat memilih berbagai modul fungsional sesuai perannya, serta mengakses menu tambahan di bagian bawah seperti profil dan absensi untuk berpindah ke halaman terkait.

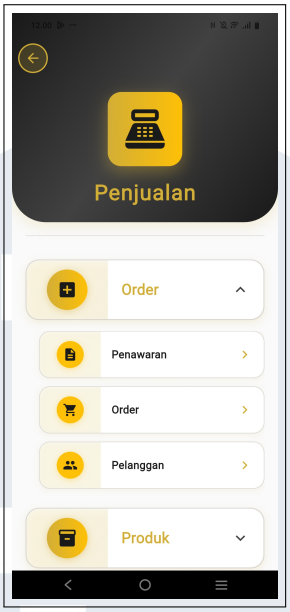
```
onTap: (index) {
  if (index == 2) {                    // Navigasi ke halaman Absensi
    Navigator.push(
      context,
      PageRouteBuilder(
        pageBuilder: (context, animation, secondaryAnimation) =>
    AttendancePage(
          serverUrl: widget.serverUrl,
          db: widget.db,
          uid: widget.uid,
          password: widget.password,
        ),
```

```
      ),
    );
  } else if (index == 1) {             // Navigasi ke halaman Profil
    Navigator.push(
      context,
      PageRouteBuilder(
        pageBuilder: (context, animation, secondaryAnimation) => ProfilePage(
          serverUrl: widget.serverUrl,
          db: widget.db,
          uid: widget.uid,
          password: widget.password,
        ),
      ),
    );
  }
},
```

Kode 3.3: Potongan Kode Navigasi Bottom Bar

Navigasi pada *bottom bar* dirancang untuk memudahkan pengguna dalam beralih antar halaman utama, seperti halaman *Attendance* dan *Profile*, tanpa harus kembali ke halaman utama terlebih dahulu. Ketika sebuah *item* pada *bottom bar* dipilih, sistem akan memicu fungsi `Navigator.push` dengan menggunakan *PageRouteBuilder* seperti pada Kode 3.3, untuk membuka halaman tujuan secara langsung, sambil membawa data sesi pengguna seperti *URL server*, *UID*, dan *password* agar koneksi ke sistem Odoo tetap berjalan tanpa perlu *login* ulang. Pendekatan ini memastikan bahwa setiap halaman yang dibuka memiliki akses lengkap terhadap informasi pengguna dan kredensial yang diperlukan untuk berinteraksi dengan *backend* Odoo.

D Tampilan Menu Modul



Gambar 3.28. Tampilan Menu Modul

Gambar 3.28 menampilkan halaman *submenu* yang muncul setelah pengguna memilih salah satu modul utama dari halaman *homepage*. Halaman ini menyajikan daftar fitur atau tugas spesifik yang terkait dengan modul yang dipilih, di mana setiap *item* menu dilengkapi dengan ikon navigasi di sisi kanan. Jika ikon tersebut berupa panah ke bawah (*expand indicator*), artinya *item* tersebut memiliki *submenu* lebih lanjut; sebaliknya, jika tidak terdapat ikon tersebut, maka *item* tersebut langsung membuka halaman tujuan tanpa navigasi tambahan.

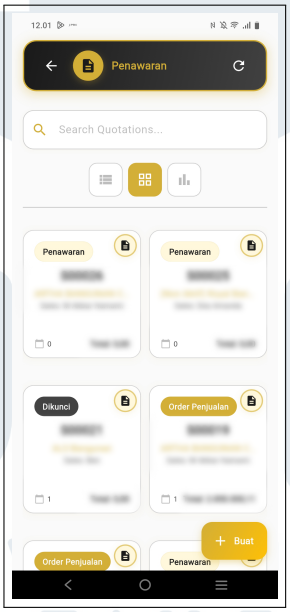
```
void _toggleOrders() {
  setState(() {
    isOrdersExpanded = !isOrdersExpanded;
    if (isOrdersExpanded) {
      _ordersController.forward();
      if (isProductsExpanded) {
        isProductsExpanded = false;
        _productsController.reverse();
      }
    } else {
      _ordersController.reverse();
    }
```

```
13    });
14  }
```

Kode 3.4: Potongan Kode Fungsi Toggle Submenu

Fungsi seperti `_toggleOrders()` pada Kode 3.4 digunakan untuk mengelola status ekspansi *submenu* pada modul *Sales*. Ketika salah satu menu (misalnya "*Orders*") diklik, sistem akan mengekspansi *submenu* terkait dengan efek animasi, dan jika menu lain sedang terbuka, maka akan ditutup secara otomatis. Pendekatan ini memastikan bahwa hanya satu *submenu* yang aktif dalam satu waktu, sehingga tampilan tetap bersih dan mudah dinavigasi oleh pengguna lapangan.

E Tampilan List View dalam Menu

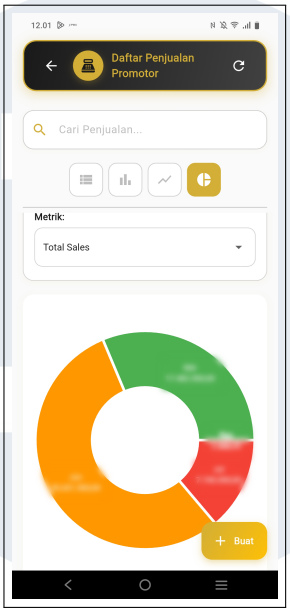


Gambar 3.29. Tampilan List dalam Menu

Gambar 3.29 menampilkan halaman daftar *item* yang sesuai dengan modul yang dipilih sebelumnya, seperti Penawaran, Penjualan, atau menu lainnya. Halaman ini menyediakan beberapa opsi tampilan seperti *List View*, *Grid View*, dan *Chart View* yang dapat dipilih pengguna untuk menyesuaikan preferensi visual dan kebutuhan analisis data. Pendekatan ini dirancang berdasarkan masukan dari *supervisor*, bertujuan memudahkan tim *sales* dalam memantau data secara fleksibel,

menghindari tampilan statis, dan meningkatkan efisiensi pengambilan keputusan di lapangan.

F Tampilan Pie Chart View dalam Menu



Gambar 3.30. Tampilan Pie Chart dalam Menu

Gambar 3.30 menampilkan tampilan visualisasi data dalam bentuk *pie chart* yang tersedia hanya pada modul-modul tertentu, seperti Daftar Penjualan Promotor. Fitur ini dirancang berdasarkan masukan dari *supervisor*, bertujuan membantu tim lapangan memahami performa penjualan secara visual melalui proporsi atau ukuran potongan grafik. Saat pengguna mengklik ikon *pie chart*, sistem akan menampilkan daftar terkait yang mendasari visualisasi tersebut, sehingga pengguna dapat melihat detail data di balik setiap sektor grafik.

```
Widget _buildPieChart() {
  final entries = groupedData[selectedPromoter] ?? [];
  if (entries.isEmpty) {
    return const Center(child: Text("No data for chart"));
  }
    ...
  final sections = entriesGrouped.asMap().entries.map((entry) {
    final value = getValue(entry.value.value);
    return PieChartSectionData(
```

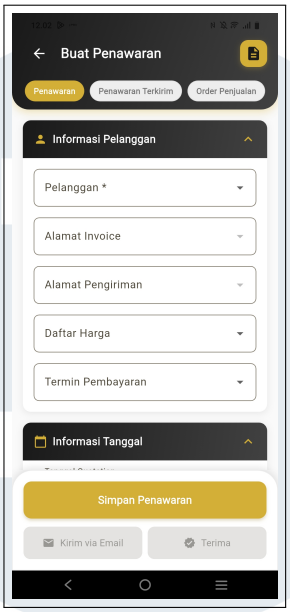
```
      value: value,
      title: "${entry.value.key}
${formatOdooNumber(value, userLangFormat!)}",
      color: [
        AppColors.secondaryGold,
        AppColors.accentYellow,
        Colors.red,
        Colors.orange,
        Colors.green,
        Colors.blue,
      ][entry.key % 6],
      radius: 80,
    );
  }).toList();

  return PieChart(
    PieChartData(
      sections: sections,
      centerSpaceRadius: 80,
      pieTouchData: PieTouchData(
        touchCallback: (event, response) {
          if (event is FlTapUpEvent && response?.touchedSection != null) {
            final index = response!.touchedSection!.touchedSectionIndex;
            final salesList = entriesGrouped[index].value;
            if (salesList.isNotEmpty) {
              _showPromoterSales(selectedPromoter!, salesList);
            }
          }
        },
      ),
    ),
  );
}
```

Kode 3.5: Potongan Kode Implementasi Pie Chart

Visualisasi data seperti yang terdapat pada Kode 3.5, khususnya dalam fungsi `_buildPieChart()`, menggunakan implementasi paket *fl_chart*. Paket *fl_chart* memungkinkan pembuatan grafik yang interaktif dan kustomisasi tinggi, sehingga sangat cocok untuk menampilkan data analitik secara visual yang informatif dan responsif [10]. Setiap sektor pada grafik memiliki warna unik dan menampilkan nilai yang diformat sesuai dengan pengaturan lokal. Saat pengguna menyentuh sektor tertentu, fungsi `_showItemDetails()` akan dipanggil untuk

menampilkan daftar *item* yang mendasari nilai pada sektor yang dipilih.

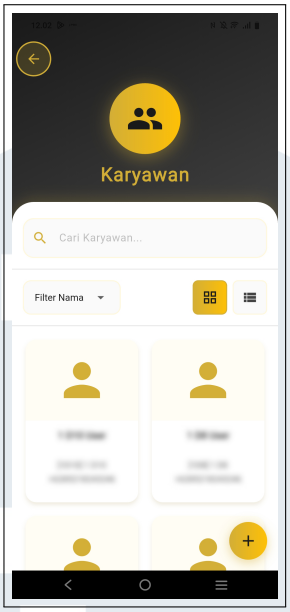
G Tampilan New/Edit Data



Gambar 3.31. Tampilan New/Edit Data

Gambar 3.31 menampilkan halaman input data yang muncul ketika pengguna memilih salah satu *item* dari daftar pada modul tertentu, atau saat memilih opsi *New* untuk membuat entri baru. Tampilan *form* ini dirancang secara dinamis, jika pengguna membuka data yang sudah ada, semua *field* akan diisi otomatis sesuai dengan data yang tersimpan, sedangkan jika pengguna membuat data baru, semua *field* akan kosong dan siap diisi. Desain ini memastikan konsistensi antarmuka, sehingga pengguna tidak perlu belajar dua tampilan berbeda hanya satu *form* yang digunakan untuk kedua fungsi (*new* dan edit), dengan penyesuaian konteks berdasarkan status data.

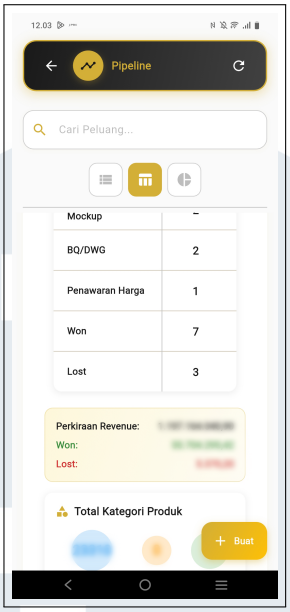
H Tampilan Daftar Employees



Gambar 3.32. Tampilan Daftar Employees

Gambar 3.32 menampilkan halaman daftar karyawan yang dirancang untuk memberikan gambaran umum mengenai setiap karyawan melalui tampilan berbentuk *card*. Setiap kartu menampilkan informasi singkat, seperti nama, divisi, serta kontak (telepon atau email), sehingga memudahkan pengguna dalam mencari dan mengidentifikasi karyawan secara cepat. Ketika pengguna memilih salah satu *card*, sistem akan membuka halaman detail untuk memungkinkan pengeditan data karyawan. Selain itu, pengguna juga dapat menambahkan data karyawan baru melalui tombol "*New*" yang tersedia di pojok kanan bawah.

I Tampilan List View Pipeline



Gambar 3.33. Tampilan List View Pipeline

Gambar 3.33 menunjukkan tampilan *view list* halaman *pipeline* yang muncul saat pengguna pertama kali membuka halaman tersebut. Sesuai permintaan tim Koordinator *Sales*, tampilan ini menampilkan tabel *stage* yang berisi jumlah total data di setiap tahapan *pipeline*, serta informasi mengenai perkiraan pendapatan, jumlah peluang yang berhasil (*won*), dan jumlah peluang yang gagal (*lost*). Tampilan ini juga mencakup jumlah produk untuk masing-masing kategori produk, yang dianggap penting untuk mendukung evaluasi kinerja penjualan.

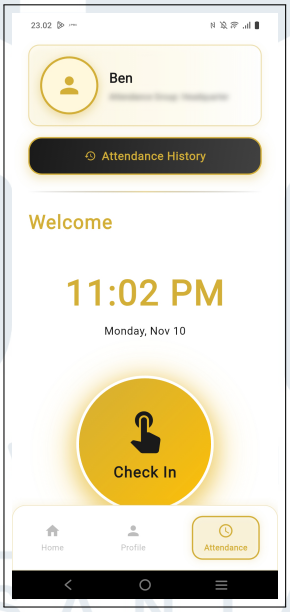
```
for (var opp in filtered) {
  String stage = safeGetString(opp["stage_id"]);
  stageCount[stage] = (stageCount[stage] ?? 0) + 1;
}

...

for (var line in productLines) {
  String mainCategory = _getMainCategoryFromName(line["product_categ_id"]);
  if (mainCategory == "Pintu Baja" || mainCategory == "Smart Lock" ||
    mainCategory == "Mevvah") {
    summary[mainCategory] += line["quantity"];
```

```
12    }
13  }
```

Kode 3.6: Potongan Kode Perhitungan Jumlah Stage dan Ringkasan Kategori Produk

Pada potongan Kode 3.6 dalam metode `_buildNewStageTable()` dan `_calculateTotalCategorySummary()`, terlihat bagaimana aplikasi menampilkan jumlah *stage* dan ringkasan kategori produk yang telah ditentukan: Pintu Baja, Smart Lock, dan Mevvah. Jumlah *stage* dihitung melalui iterasi pada daftar *opportunities* yang telah difilter berdasarkan *stage_id*, sementara total per kategori produk dihitung dengan menelusuri hierarki *product_categ_id* dari setiap *product line*, lalu mengelompokkannya ke dalam tiga kategori utama tersebut. Hasil penghitungan ini kemudian ditampilkan secara *real-time* di tabel dan *widget* ringkasan menggunakan `setState()`.

J Tampilan Attendances

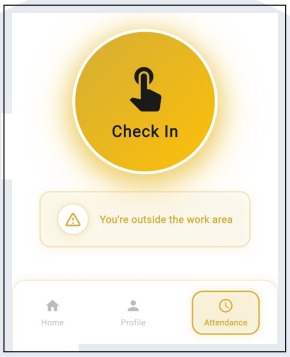


Gambar 3.34. Tampilan Attendances

Gambar 3.34 menampilkan halaman absensi karyawan pada *menu Attendances*. Tampilan ini menyajikan waktu *real-time* serta tombol *Check In* untuk melakukan absensi. Selain itu, terdapat indikator visual berupa kotak status yang secara otomatis memberitahu apakah pengguna berada dalam area perusahaan

(berdasarkan lokasi *GPS* atau *geofencing*), sehingga memastikan status untuk proses absensi.

K Implementasi Modul Geofence pada Attendance



Gambar 3.35. Status Lokasi Absensi

Pada halaman absensi di tampilan *Attendances* aplikasi *mobile*, terdapat indikator status lokasi yang memberitahu apakah pengguna berada di dalam area perusahaan atau tidak seperti pada Gambar 3.35. Sebelumnya, diperlukan modul *geofence* untuk menentukan batas lokasi perusahaan. Karena modul *geofence* pada sistem Odoo merupakan modul berbayar, maka dibuatlah modul *geofence* versi sederhana yang menyerupai fungsionalitas aslinya. Modul ini dirancang sesuai dengan *operating unit*, karena *operating unit* juga merupakan modul inti yang digunakan sebagai dasar pengelompokan data.

Pengguna diharapkan terlebih dahulu menentukan *operating unit* yang digunakan, kemudian menambahkan titik koordinat (*latitude* dan *longitude*) dari lokasi perusahaan, yang dapat diperoleh melalui layanan peta seperti *Google Maps*. Modul *geofence* ini dibangun sebagai model *inheritance* yang terintegrasi dengan modul terkait, seperti *hr.attendance* dan *operating.unit*, untuk memastikan sinkronisasi data dan fungsi absensi berbasis lokasi berjalan secara efektif.

```
def distance(lat1, lon1, lat2, lon2):
    R = 6371000  # Radius bumi dalam meter
    phi1, phi2 = math.radians(lat1), math.radians(lat2)
    delta_phi = math.radians(lat2 - lat1)
    delta_lambda = math.radians(lon2 - lon1)
    a = math.sin(delta_phi / 2)**2 + math.cos(phi1) * math.cos(phi2) * math.
    sin(delta_lambda / 2)**2
```

```
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
    return R * c
```

Kode 3.7: Potongan Kode Fungsi Perhitungan Jarak

.

Fungsi untuk validasi lokasi menggunakan pendekatan formula *Haversine* untuk menghitung jarak antara koordinat GPS karyawan dengan pusat area *geofence*. Fungsi *Haversine* memperkirakan jarak secara garis lurus antara dua titik berdasarkan koordinat *GPS* dan menjadi dasar dalam menghitung jarak dalam sistem geografis [11].

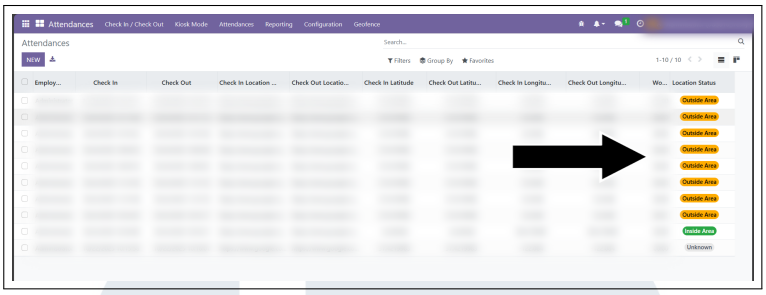
Pada potongan Kode 3.7, fungsi `distance()` menerima empat parameter berupa koordinat *latitude* dan *longitude* dari dua titik: koordinat karyawan (*lat1*, *lon1*) yang diperoleh dari *GPS* perangkat *mobile*, dan koordinat pusat area *geofence* (*lat2*, *lon2*) yang tersimpan di *database*. Fungsi ini mengembalikan jarak dalam satuan meter yang kemudian dibandingkan dengan nilai radius yang dikonfigurasi untuk menentukan apakah karyawan berada di dalam atau di luar area yang diizinkan.

```
@api.model_create_multi
def create(self, vals_list):
    # Ambil koordinat dari context (dikirim dari frontend)
    lat = self._context.get('latitude')
    lon = self._context.get('longitude')

    # Jika tidak ada koordinat, langsung create tanpa validasi
    if not lat or not lon:
        return super().create(vals_list)

.....

    # Hitung jarak menggunakan koordinat karyawan vs pusat geofence
    dist = distance(lat, lon, geofence.latitude, geofence.longitude)
    location_status = 'inside' if dist <= geofence.radius else 'outside'
```

Kode 3.8: Potongan Kode Validasi Geofencing pada Method Create

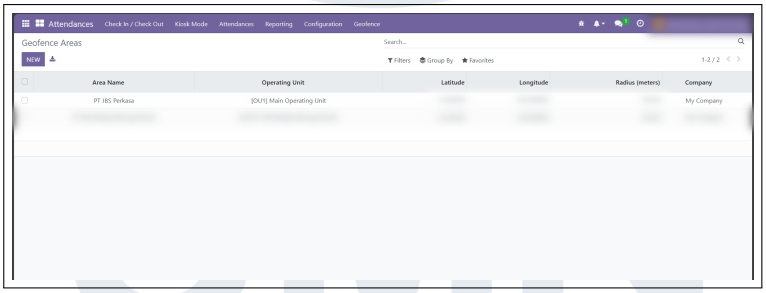
Fungsi *create* ini merupakan *override method* bawaan Odoo yang digunakan untuk memvalidasi lokasi karyawan saat membuat *record* baru berdasarkan koordinat *GPS* yang dikirim dari *frontend*. Fungsi ini mengambil koordinat *latitude* dan *longitude* dari *context*, kemudian menghitung jarak antara posisi karyawan dengan titik pusat *geofence* menggunakan formula *Haversine*. Berdasarkan hasil

perhitungan jarak tersebut, sistem menentukan status lokasi apakah karyawan berada di dalam (*inside*) atau di luar (*outside*) radius *geofence* yang telah ditentukan.



Gambar 3.36. Kolom Status Lokasi pada Modul Attendances Odoo

Gambar 3.36 menampilkan tampilan tabel pada modul *Attendances* di Odoo ERP, yang telah diperbarui dengan penambahan kolom baru: "*Location Status*". Kolom ini berfungsi untuk menyimpan dan menampilkan status lokasi karyawan (misalnya: *Inside Area*, *Outside Area*, atau *Unknown*) saat melakukan absensi melalui aplikasi *mobile*. Data dalam kolom ini diisi secara otomatis oleh aplikasi berdasarkan hasil perhitungan *geofence*.



Gambar 3.37. Halaman Konfigurasi Geofence pada Modul Attendances

Gambar 3.37 menampilkan halaman konfigurasi *Geofence Areas* pada Modul *Attendances* Odoo ERP, yang digunakan untuk mendefinisikan wilayah absensi berdasarkan *Operating Unit*. Sebelum aplikasi mobile dapat mengecek lokasi kehadiran, pengguna (biasanya administrator) harus terlebih dahulu memasukkan data geografis untuk setiap *Operating Unit*, termasuk koordinat *latitude* dan *longitude*, serta radius area yang diizinkan. Data ini kemudian digunakan oleh aplikasi mobile untuk memverifikasi apakah karyawan berada di dalam area yang ditentukan saat melakukan *check-in* atau *check-out*.

3.5.4 Pengujian Aplikasi Mobile Cross-Platform

Proses pengujian dilakukan untuk mengevaluasi fungsionalitas aplikasi, termasuk penggunaan setiap modul yang tersedia, serta verifikasi bahwa fitur-fitur seperti pengiriman email, perubahan status data, dan penambahan data ke dalam daftar berjalan sesuai harapan. Pengujian dilakukan dengan menggunakan metode *Black box testing*. Metode pengujian *Black box testing* merupakan proses yang dilakukan tanpa pengetahuan mengenai detail internal sistem yang diuji dan *testers* hanya dapat mengumpulkan informasi melalui saluran publik dan melakukan pengujian dari luar sistem, tanpa akses ke kode sumber maupun dokumentasi internal [12].

Tabel 3.4. Hasil Pengujian *Black Box Testing* Aplikasi Mobile

No	Skenario Pengujian	Fitur yang Diuji	Ekspektasi Hasil	Hasil Pengujian	Status
1	Input kredensial valid	*Login*	Aplikasi berhasil *login* dan masuk ke halaman *homepage*	User berhasil login dan *redirect* ke *homepage*	Berhasil
2	Input kredensial tidak valid	*Login*	Muncul notifikasi error ”*Login* Gagal”	Notifikasi error ”*Username*, *Password*, atau Nama *Database* Salah” ditampilkan	Berhasil
3	Centang *Remember Me*	*Login*	Kredensial tersimpan untuk *login* berikutnya	Kredensial tersimpan dan *auto-fill* saat buka aplikasi	Berhasil
4	*Uncheck Remember Me*	Login	Kredensial tidak tersimpan	Kredensial terhapus dan tidak *auto-fill*	Berhasil

Bersambung ke halaman berikutnya

Tabel 3.4. Hasil Pengujian *Black Box Testing* Aplikasi Mobile (lanjutan)

No	Skenario Pengujian	Fitur yang Diuji	Ekspektasi Hasil	Hasil Pengujian	Status
5	Akses halaman *homepage*	*Homepage*	Menampilkan menu utama dan informasi user	Menu utama dan nama user ditampilkan dengan benar	Berhasil
6	Navigasi menu *bottom navbar*	*Homepage*	Berpindah antar menu	Perpindahan antar menu berfungsi	Berhasil
7	Melihat daftar *quotations*	*Sales - Quotations*	Menampilkan list *quotations* dari Odoo	List *quotations* ditampilkan dengan data lengkap	Berhasil
8	Membuat *quotation* baru	*Sales - Quotations*	Form input *quotation* dan data tersimpan ke Odoo	*Quotation* baru berhasil dibuat dan tersimpan	Berhasil
9	Mengirim Data *Quotation* lewat tombol *Send by Email*	*Sales - Quotations*	Data akan terkirim ke email tujuan dengan lampiran *PDF*	Data terkirim ke email tujuan dengan lampiran *PDF*	Berhasil
10	Menambah produk ke *quotation*	*Sales - Quotations*	Produk ditambahkan dengan kalkulasi harga otomatis	Produk berhasil ditambahkan dan harga terhitung	Berhasil
11	Menghapus produk dari *quotation*	*Sales - Quotations*	Produk terhapus dan total harga ter-update	Produk terhapus dan kalkulasi harga ter-update	Berhasil

Bersambung ke halaman berikutnya

Tabel 3.4. Hasil Pengujian *Black Box Testing* Aplikasi Mobile (lanjutan)

No	Skenario Pengujian	Fitur yang Diuji	Ekspektasi Hasil	Hasil Pengujian	Status
12	Melihat detail *quotation*	*Sales - Quotations*	Menampilkan detail lengkap *quotation*	Detail *quotation* ditampilkan dengan akurat	Berhasil
13	Melihat daftar *sales orders*	*Sales - Orders*	Menampilkan list *sales orders* dari Odoo	List *sales orders* ditampilkan dengan status masing-masing	Berhasil
14	Filter *orders* berdasarkan status	*Sales - Orders*	Menampilkan *orders* sesuai filter yang dipilih	Filter berfungsi dan menampilkan data sesuai kriteria	Berhasil
15	Melihat detail *order*	*Sales - Orders*	Menampilkan informasi lengkap order	Detail order ditampilkan dengan lengkap	Berhasil
16	Melihat daftar produk	*Products*	Menampilkan list produk dengan harga	List produk ditampilkan dengan informasi lengkap	Berhasil
17	Pencarian produk	*Products*	Menampilkan hasil pencarian produk	Hasil pencarian produk ditampilkan dengan akurat	Berhasil

Bersambung ke halaman berikutnya

Tabel 3.4. Hasil Pengujian *Black Box Testing* Aplikasi Mobile (lanjutan)

No	Skenario Pengujian	Fitur yang Diuji	Ekspektasi Hasil	Hasil Pengujian	Status
18	Melihat daftar *customer*	*Customers*	Menampilkan list *customer* dari Odoo	List *customer* ditampilkan dengan data kontak	Berhasil
19	Melihat detail *customer*	*Customers*	Menampilkan informasi lengkap *customer*	Detail *customer* ditampilkan dengan lengkap	Berhasil
20	Melihat detail penjualan promotor	*Sales Promotor* - Penjualan	Menampilkan detail promotor dalam berbagai pilihan *view*	Detail penjualan ditampilkan dengan lengkap dengan berbagai pilihan *view*	Berhasil
21	Merubah status data penjualan promotor	*Sales Promotor* - Penjualan	Status dapat diubah dari *DRAFT* menjadi *CHECKED*	Status berubah dari *DRAFT* menjadi *CHECKED*	Berhasil
22	Melihat daftar *pipeline*	*CRM - My Pipeline*	Menampilkan list *opportunities*	List *opportunities* ditampilkan dengan *stage* masing-masing	Berhasil

Bersambung ke halaman berikutnya

Tabel 3.4. Hasil Pengujian *Black Box Testing* Aplikasi Mobile (lanjutan)

No	Skenario Pengujian	Fitur yang Diuji	Ekspektasi Hasil	Hasil Pengujian	Status
23	Membuat *opportunity* baru	*CRM - My Pipeline*	Form input *opportunity* dan data tersimpan	*Opportunity* baru berhasil dibuat dan tersimpan	Berhasil
24	Visualisasi *chart pipeline*	*CRM - My Pipeline*	Menampilkan *bar chart* dan *pie chart*	*Chart* ditampilkan dengan data yang akurat	Berhasil
25	Melihat daftar penjualan	*Sales Promotor*	Menampilkan list penjualan promotor	List penjualan ditampilkan dengan target dan realisasi	Berhasil
26	Input data penjualan	*Sales Promotor*	Data penjualan tersimpan ke sistem	Data penjualan berhasil disimpan	Berhasil
27	Melihat data toko	*Sales Promotor*	Menampilkan informasi toko dan promotor	Data toko dan promotor ditampilkan dengan lengkap	Berhasil
28	*Filter* promotor berdasarkan *branch*	Sales Promotor	Menampilkan promotor sesuai cabang	*Filter* berfungsi dan data sesuai cabang ditampilkan	Berhasil
29	*Check-in* attendance dalam radius	*Attendance*	Status *attendance* ”Di dalam area” dan data tersimpan	Status ”Di dalam area” ditampilkan dan data tersimpan	Berhasil

Bersambung ke halaman berikutnya

Tabel 3.4. Hasil Pengujian *Black Box Testing* Aplikasi Mobile (lanjutan)

No	Skenario Pengujian	Fitur yang Diuji	Ekspektasi Hasil	Hasil Pengujian	Status
30	*Check-in attendance* luar radius	*Attendance*	Status *attendance* ”Di luar area” dan notifikasi peringatan	Status ”Di luar area” ditampilkan dengan notifikasi	Berhasil
31	*Check-out attendance*	*Attendance*	Data *check-out* tersimpan dengan durasi kerja	Data *check-out* tersimpan dan durasi ditampilkan	Berhasil
32	Melihat histori *attendance*	*Attendance*	Menampilkan riwayat *check-in/out*	Histori *attendance* ditampilkan dengan lengkap	Berhasil
33	Melihat profil *user*	*Profile*	Menampilkan informasi *user*	Data profil *user* ditampilkan dengan lengkap	Berhasil
34	Sinkronisasi data *real-time*	Integrasi Odoo	Data yang diinput di *mobile* tersimpan di Odoo	Data berhasil tersinkronisasi ke database Odoo	Berhasil

Bersambung ke halaman berikutnya

Tabel 3.4. Hasil Pengujian *Black Box Testing* Aplikasi Mobile (lanjutan)

No	Skenario Pengujian	Fitur yang Diuji	Ekspektasi Hasil	Hasil Pengujian	Status
35	List View Pipeline	Pipeline	Data stage table, data pipeline lost/win, dan data jumlah produk dari kategori produk yang ditentukan muncul	Data muncul dengan total akumulasi yang tepat dari masing-masing stage serta total produk dari masing-masing kategori produk	Berhasil

Berdasarkan hasil pengujian *black box* yang dilakukan terhadap aplikasi *mobile cross-platform*, seluruh fitur fungsional pada setiap modul telah berhasil berjalan sesuai harapan. Tidak ditemukan *bug* atau *error* yang berasal dari sisi aplikasi, kecuali masalah yang berasal langsung dari sistem Odoo ERP. Fitur absensi (*Attendance*) saat ini masih dalam tahap diskusi internal terkait kemungkinan migrasi dari aplikasi *DingTalk* yang sebelumnya digunakan untuk pencatatan kehadiran. Aplikasi akan tetap dilakukan pemeliharaan secara berkala untuk memastikan kinerja optimal dan adaptasi terhadap kebutuhan pengguna di masa mendatang.

3.6 Kendala dan Solusi yang Ditemukan

A Kendala

Selama pelaksanaan kerja magang di PT. Jaya Bersama Saputra Perkasa, beberapa kendala yang memengaruhi efisiensi dan kualitas kerja, antara lain:

1. Kesulitan dalam mencari referensi teknis terkait desain *UI* dan implementasi fitur khusus, seperti visualisasi data interaktif dan modul *geofence* versi sederhana. Hal ini disebabkan oleh kurangnya dokumentasi atau tutorial

spesifik untuk pendekatan yang digunakan, serta seringnya perubahan kebutuhan fungsional dari *supervisor*.

2. Ketidakstabilan koneksi internet di ruangan kantor, yang menyebabkan jaringan sering *lag*, terputus, atau bahkan tidak dapat digunakan. Kondisi ini menghambat kerja pengembangan dan pengujian aplikasi.

3. Terjadinya *error white screen* pada Odoo saat pembuatan atau instalasi modul *custom*. *Error* ini cukup menyulitkan karena tidak menampilkan pesan *error* yang jelas, sehingga sulit mengidentifikasi akar masalah apakah berasal dari modul *custom* yang sedang dikembangkan, *dependencies* yang tidak kompatibel, atau konfigurasi sistem yang salah.

B Solusi

Untuk mengatasi kendala-kendala tersebut, beberapa solusi yang diterapkan adalah:

1. Penggunaan dokumentasi resmi dan eksplorasi langsung terhadap *API* Odoo serta pustaka *Flutter* yang relevan. Pendekatan ini membantu dalam memahami mekanisme integrasi dan membangun fitur-fitur khusus secara mandiri berdasarkan kebutuhan spesifik. Selain itu, pencarian referensi desain *UI* dilakukan melalui contoh aplikasi lain di internet yang memiliki tampilan sederhana namun menarik, sesuai dengan preferensi *supervisor*, untuk dijadikan inspirasi dalam merancang antarmuka aplikasi.

2. Penggunaan koneksi data seluler atau *hotspot* lain sebagai alternatif saat jaringan kantor bermasalah. Pendekatan ini memungkinkan proses pengembangan tetap berjalan meskipun kondisi jaringan utama tidak mendukung.

3. Melakukan debugging sistematis dengan memeriksa *log* Odoo melalui command line, menjalankan mode development dengan *–log-level=debug*, dan mematikan modul satu per satu untuk identifikasi konflik.