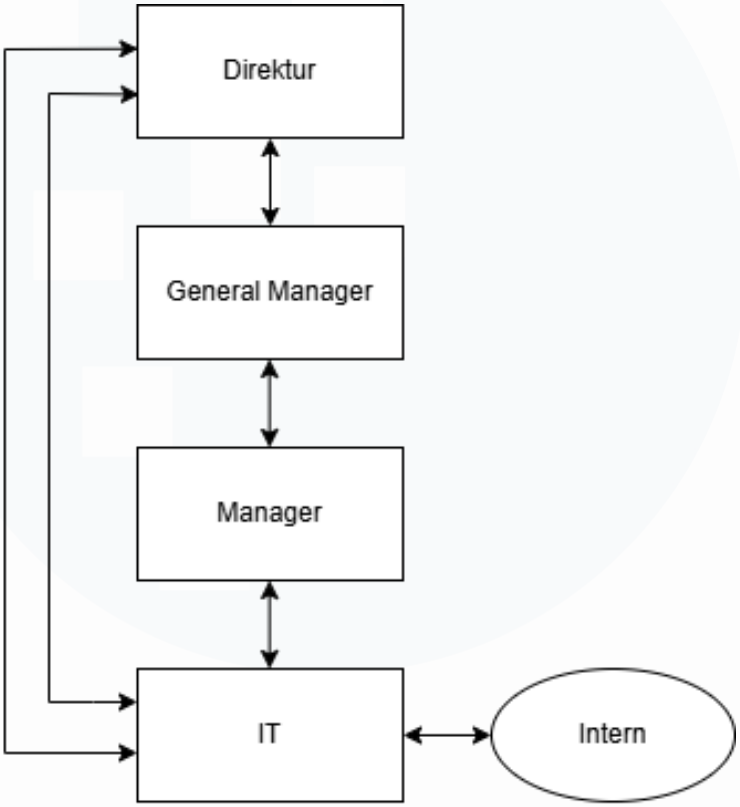
BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi



Gambar 3.1 Kedudukan dan Koordinasi Kerja Magang

Struktur organisasi menunjukkan bahwa pekerja magang (Intern) berada di bawah divisi IT, yang bertugas membimbing, memberikan arahan, dan menerima laporan dari magang. Jika ada kendala atau tugas, pekerja magang akan berkoordinasi dengan divisi IT untuk mendapatkan solusi atau instruksi yang tepat.

Divisi IT berada di bawah Manager, yang bertanggung jawab dalam pengambilan keputusan, dan selanjutnya berkoordinasi dengan *General Manager* sebagai penghubung ke Direktur. *General Manager* menyampaikan perkembangan kerja serta meminta persetujuan kebijakan kepada Direktur, yang memiliki wewenang tertinggi dalam perusahaan.

Struktur ini juga memungkinkan divisi IT dan pekerja magang berkomunikasi langsung dengan Direktur atau *General Manager*, terutama dalam situasi mendesak atau proyek khusus. Dengan pola kerja ini, penyelesaian masalah teknis dan proyek IT dapat dilakukan lebih cepat dan efisien.

3.2 Tugas dan Uraian Kerja Magang

Tabel 3.1 Waktu Pelaksanaan Magang Perusahaan

No	Minggu ke-	
1	1	Onboarding dan Perkenalan Perusahaan
2	2-4	Mempelajari *Workflow* MIRA
3	5-6	Melakukan Perancangan Sistem Aplikasi MIRA
4	7-14	Pengembangan Sistem Secara Paralel *Frontend* dan *Backend*

MIRA adalah aplikasi *Customer Relationship Management* (CRM) yang dibuat untuk membantu tim marketing dan sales PT Maro Anugrah Jaya dalam mengelola data *lead* dan transaksi dengan lebih mudah. Aplikasi ini menggantikan proses manual dengan sistem digital, sehingga pencatatan, pencarian, dan pembaruan data lead bisa dilakukan lebih cepat dan rapi. MIRA memiliki beberapa fitur utama, seperti *Lead Management*, Progress Transaksi, Data Konsumen, Produk & Katalog, Program Promosi, Penghargaan, dan *Task Management*. Namun pada tahap awal pengembangan MIRA, fokus utamanya adalah fitur *Lead Management* sehingga dapat membantu sales dalam mencari, mencatat, dan memperbarui status calon pelanggan. Tugas penulis ialah merancang dan membuat sistem CRM dalam bentuk aplikasi MIRA yang dibuat dalam waktu kurang lebih 14 minggu.

3.2.1 Workflow

1. Pencatatan Lead

- Data calon pembeli (lead) dicatat menggunakan spreadsheet.
- Lead ID dibuat berdasarkan sumber lead:
  - **BS_[Nomor HP]_XXX** → Lead dari sales sendiri.

- **CRM_[Nomor HP]_XXX** → Lead dari tim marketing.
- Data yang dicatat dalam lead mencakup:
  - Nama
  - Kontak
  - Pekerjaan
  - Penghasilan
  - Sumber lead
  - Status (hot, warm, cold)
  - Catatan tindak lanjut

2. Promo untuk Lead

- Lead dapat menerima promo.
- Promo yang sudah tidak berlaku tetap dapat digunakan jika lead sudah membayar booking sebelumnya.

3. Proses Transaksi

- Jika lead membayar booking, statusnya berubah menjadi **customer**.
- Tahapan proses transaksi:
  1. **SPKB (Surat Pemesanan Kavling dan Bangunan) atau booking (UTJ)**.
  2. **Pengajuan KPR** (jika customer memilih KPR).
     - SP3K (Surat Penegasan Persetujuan Penyediaan Kredit) harus diperoleh dalam **10 hari**.
  3. **Akad Jual-Beli**
     - Dilakukan dalam **5 hari** setelah persetujuan KPR atau pembayaran cash.
  4. **Serah Terima Unit**
     - Setelah akad, rumah diserahkan melalui proses **BAST (Berita Acara Serah Terima)**.

4. Proses Refund dan Pembatalan

- Jika ada pembatalan atau pengajuan refund, proses masuk ke tahap **Refund**.

5. Tugas Tim Sales

- Melakukan follow-up terhadap lead.
- Memperbarui data di spreadsheet, termasuk:
  - Tanggal kunjungan
  - Status transaksi
  - Dokumen pendukung

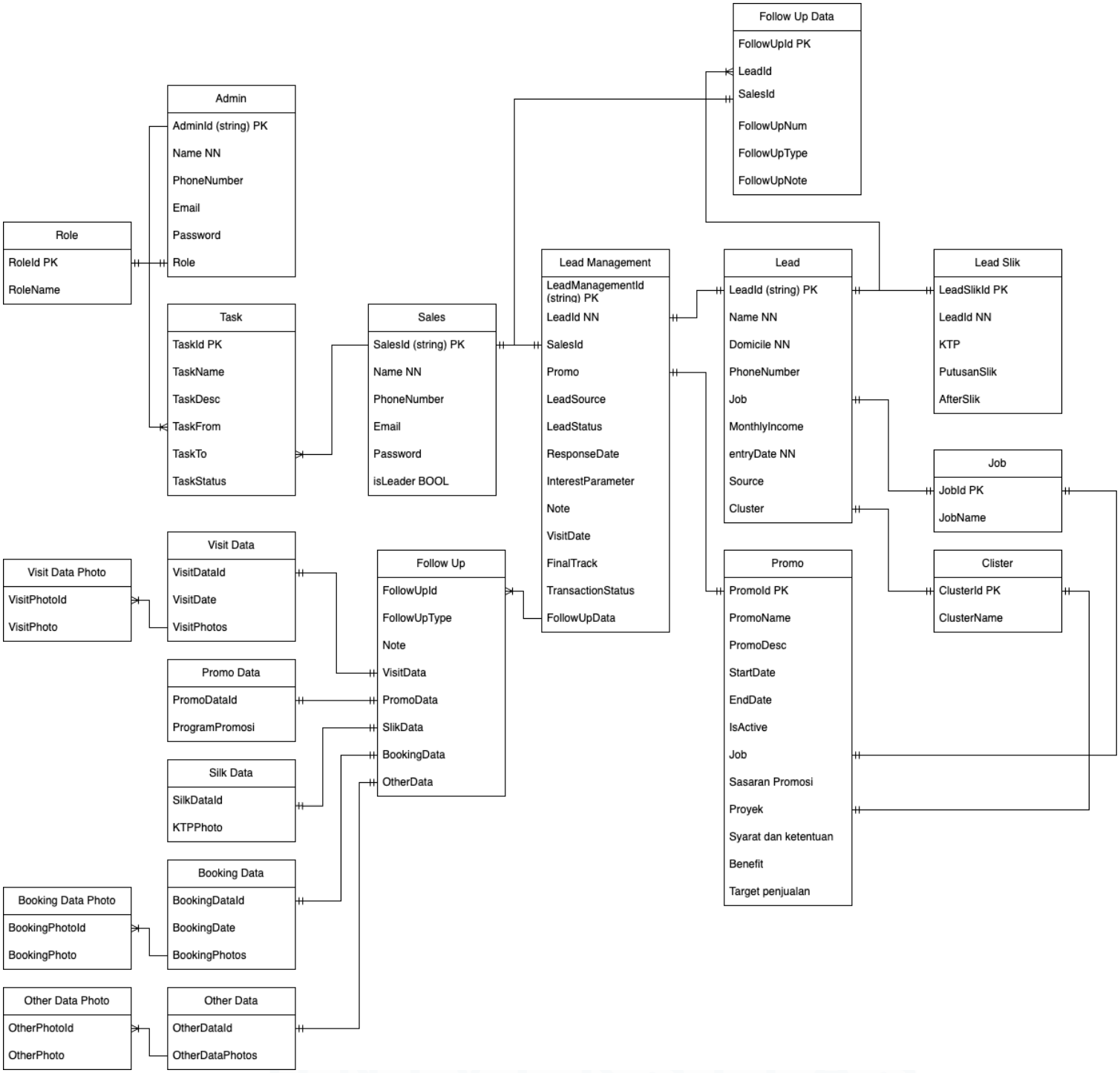
6. Kendala Sistem Saat Ini

- Proses pencatatan masih dilakukan secara manual menggunakan spreadsheet.
- Kurang efisien dan memakan banyak waktu.

3.2.2 Perancangan Sistem

Arsitektur sistem dalam aplikasi MIRA menggunakan konsep client-server dengan tiga lapisan utama, yaitu *frontend*, *backend*, dan *database*. *Frontend* dikembangkan menggunakan React Native dengan Expo, yang bertanggung jawab untuk menampilkan antarmuka pengguna dan mengirimkan permintaan ke backend melalui REST API. Backend dibuat menggunakan Express.js (Node.js), yang berfungsi untuk menerima permintaan dari frontend, memproses data, dan mengirimkan respons dalam format JSON. Sementara itu, database menggunakan MySQL untuk menyimpan data seperti informasi lead, status transaksi, dan pengguna aplikasi.

Dalam sistem MIRA, terdapat beberapa tabel utama yang digunakan untuk mengelola data lead, sales, transaksi, serta tugas yang diberikan kepada tim. Tabel-tabel ini saling terhubung untuk memastikan setiap proses dalam workflow sales dapat berjalan dengan baik. Berikut ERD dari aplikasi MIRA sementara untuk fitur *lead management*:



Gambar 3.2 ERD Fitur Lead Management MIRA

Gambar 3.2 menunjukkan ERD dari fitur *Lead Management* MIRA. Pada ERD yang akan diimplementasikan beberapa tabel yang digunakan pada fitur Lead Management yaitu:

1. Admin

   Tabel Admin digunakan untuk menyimpan informasi tentang admin yang memiliki peran dalam mengelola sistem. Data yang disimpan dalam tabel ini meliputi ID admin (bilangan bulat unik yang bertambah otomatis), nama admin (teks), nomor telepon admin (teks unik), email admin (teks unik untuk login), password (teks sebagai kata sandi), ID peran admin (bilangan bulat yang merujuk ke tabel UserRole), dan foto profil admin (opsional dalam bentuk teks URL). Tabel ini berelasi dengan UserRole untuk menentukan peran admin dan dengan Task untuk mencatat tugas yang diberikan oleh admin.

2. Sales

   Tabel Sales digunakan untuk menyimpan informasi tentang tenaga penjualan yang bertanggung jawab atas pengelolaan lead dan tindak lanjutnya. Data yang disimpan dalam tabel ini meliputi ID sales (bilangan bulat unik yang bertambah otomatis), nama sales (teks), nomor telepon sales (teks unik), email sales (teks unik untuk login), password (teks sebagai kata sandi), foto profil sales (opsional dalam bentuk teks URL), dan ID peran sales (bilangan bulat yang merujuk ke tabel UserRole). Sales juga memiliki hubungan dengan LeadManagement untuk menangani lead dan dengan Task untuk menerima tugas dari admin.

3. User Role

   Tabel UserRole menyimpan informasi tentang peran pengguna dalam sistem, seperti admin atau sales. Data dalam tabel ini meliputi ID peran (bilangan bulat unik yang bertambah otomatis) dan nama peran (teks unik yang menggambarkan peran pengguna, misalnya

"Admin" atau "Sales"). Tabel ini berelasi dengan Admin dan Sales untuk menentukan peran mereka dalam sistem.

4. Lead

   Tabel Lead menyimpan informasi tentang calon pelanggan yang tertarik dengan produk atau layanan. Data yang disimpan meliputi ID lead (bilangan bulat unik yang bertambah otomatis), ID unik lead (teks unik yang dihasilkan dari sumber lead), nama lead (teks), nomor telepon lead (teks unik), domisili lead (teks), ID pekerjaan lead (bilangan bulat yang merujuk ke tabel Job), penghasilan bulanan (enum yang mencakup kategori pendapatan), tanggal masuk lead (tanggal dan waktu), sumber lead (enum yang menunjukkan asal lead seperti kanvasing atau CRM MARO), dan ID cluster (bilangan bulat opsional yang merujuk ke tabel Cluster). Tabel ini berelasi dengan Job untuk informasi pekerjaan, Cluster untuk area perumahan yang diincar, LeadManagement untuk mencatat perkembangan lead, dan LeadSlik untuk menyimpan data pemeriksaan SLIK.

5. Lead Management

   Tabel LeadManagement mengelola perkembangan lead, mencatat status mereka, promo yang digunakan, dan tindak lanjut yang telah dilakukan. Data yang disimpan meliputi ID lead management (bilangan bulat unik yang bertambah otomatis), ID lead (teks unik yang merujuk ke tabel Lead), ID sales (bilangan bulat yang merujuk ke tabel Sales), catatan respons (teks opsional untuk mencatat interaksi awal), ID promo (bilangan bulat opsional yang merujuk ke tabel Promo), status lead (enum seperti Cold, Warm, atau Hot), parameter ketertarikan (enum seperti harga atau lokasi), tanggal kunjungan lokasi (opsional dalam format tanggal dan waktu), status akhir lead (enum seperti Closing atau Losing), dan status transaksi

(enum yang mencerminkan tahap transaksi). Tabel ini berelasi dengan Lead, Sales, Promo, dan FollowUp.

6. Follow Up

   Tabel FollowUp mencatat tindak lanjut yang dilakukan oleh sales terhadap lead. Data yang disimpan meliputi ID follow-up (bilangan bulat unik yang bertambah otomatis), ID lead management (bilangan bulat yang merujuk ke tabel LeadManagement), jenis follow-up (enum seperti kunjungan lokasi atau pengecekan SLIK), dan catatan follow-up (teks opsional). Tabel ini berelasi dengan LeadManagement dan dapat memiliki data tambahan melalui VisitData, PromoData, SlikData, BookingData, dan OtherFollowUpData.

7. Promo

   Tabel Promo menyimpan program promosi yang ditawarkan kepada calon pelanggan. Data yang disimpan meliputi ID promo (bilangan bulat unik yang bertambah otomatis), nama promo (teks), deskripsi promo (teks opsional), tanggal mulai promo (tanggal dan waktu), tanggal berakhir promo (tanggal dan waktu), dan status aktif (boolean yang menunjukkan apakah promo masih berlaku). Tabel ini berelasi dengan LeadManagement karena promo dapat digunakan oleh lead tertentu.

8. Task

   Tabel Task menyimpan tugas yang diberikan oleh admin kepada sales untuk memastikan lead ditangani dengan baik. Data yang disimpan meliputi ID tugas (bilangan bulat unik yang bertambah otomatis), nama tugas (teks), deskripsi tugas (teks), ID pemberi tugas (bilangan bulat yang merujuk ke tabel Admin), ID penerima tugas (bilangan bulat yang merujuk ke tabel Sales), batas waktu

tugas (tanggal dan waktu), dan status tugas (enum seperti Not Yet, In Progress, atau Completed). Tabel ini berelasi dengan Admin sebagai pemberi tugas dan Sales sebagai penerima tugas.

3.2.3 Pengembangan Frontend

Aplikasi MIRA dikembangkan menggunakan React Native dengan Expo karena memungkinkan pembuatan aplikasi mobile untuk Android dan iOS. React Native dipilih karena memiliki performa yang baik dan komunitas yang besar, sehingga mudah untuk mencari solusi jika ada masalah. Sementara itu, Expo digunakan untuk mempermudah proses pengembangan, karena menyediakan banyak fitur bawaan seperti push notification, akses kamera, dan manajemen update aplikasi tanpa perlu konfigurasi tambahan. Pada aplikasi MIRA, digunakan Expo Router sebagai navigasi antar *screen* dan digunakan juga Axios yang mengatur pengambilan dan pengiriman data dari server.

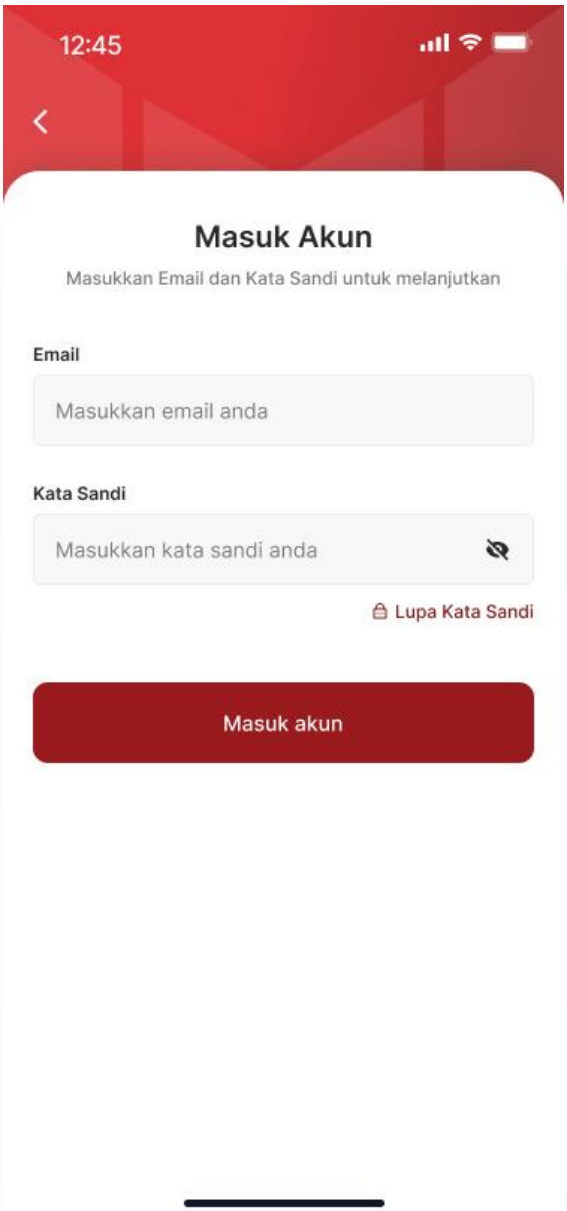
Pada project MIRA, terdapat UI/UX yang mendesain aplikasi agar sederhana dan mudah digunakan oleh tim sales PT. Maro Anugrah Jaya. Berikut beberapa screen dan komponen yang akan dibuat:

1. *Welcoming Screen*

Gambar 3.3 *Welcome Screen*

Pada gambar 3.3 menampilkan *Welcome Screen* yang muncul saat aplikasi dibuka sebelum pengguna diarahkan ke halaman login.

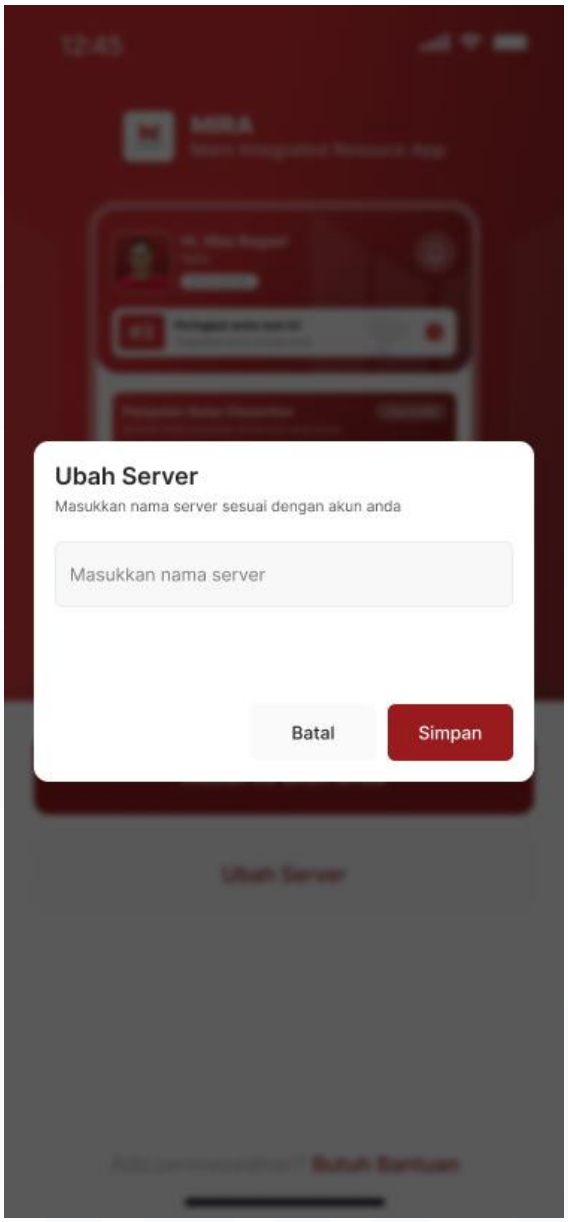
2. *Login Screen*



Gambar 3.4 *Login Screen*

Pada gambar 3.4 menampilkan *Login Screen* untuk pengguna masuk ke aplikasi dengan memasukkan email/nomor HP dan password. Terdapat tombol untuk masuk serta opsi Lupa Password jika pengguna lupa kata sandinya.

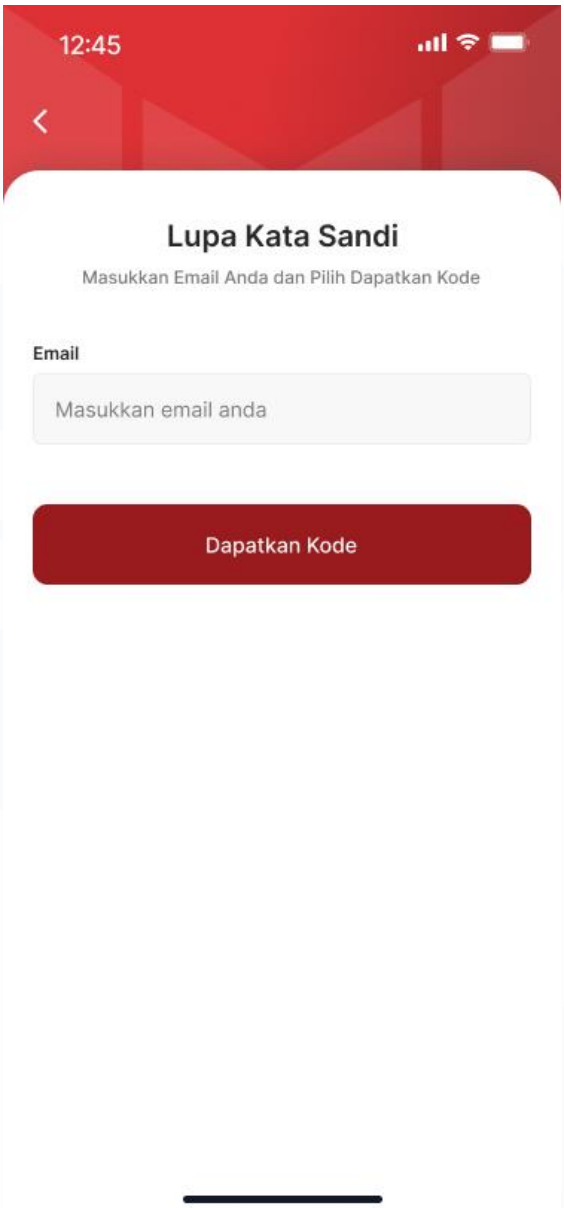
3. *Change Server Modal*



Gambar 3.5 *Change Server Modal*

Pada gambar 3.5 menampilkan *pop-up* atau *modal* yang memungkinkan pengguna mengubah server tujuan untuk koneksi aplikasi.

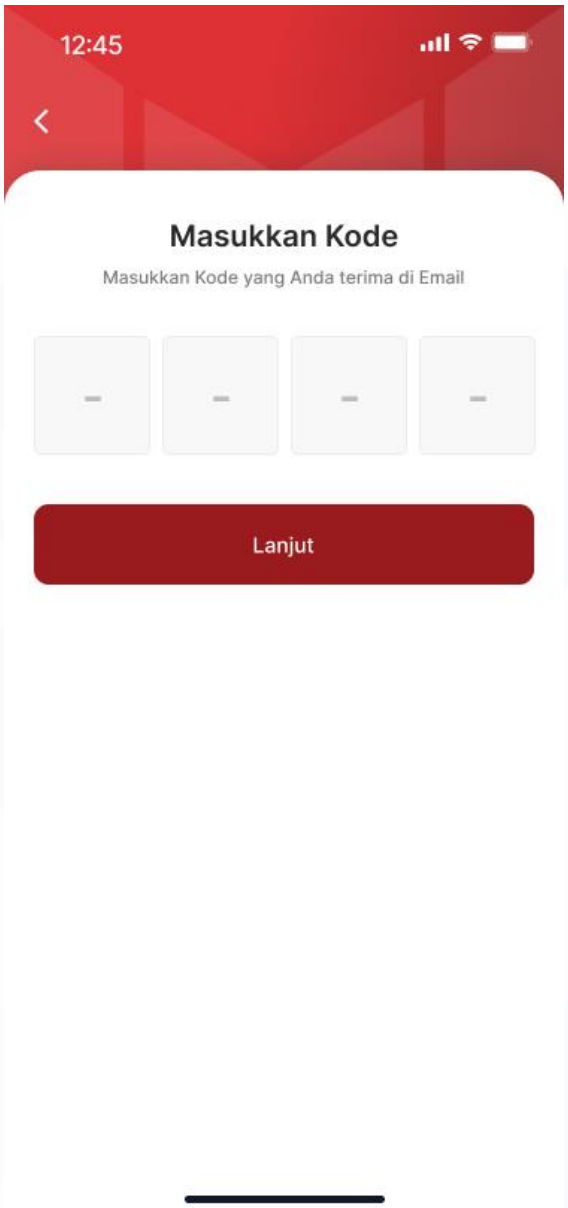
4. *Forget Password Screen*



Gambar 3.6 *Forget Password Screen*

Pada gambar 3.6 menampilkan halaman yang memungkinkan pengguna meminta reset password dengan memasukkan email atau nomor HP yang terdaftar.

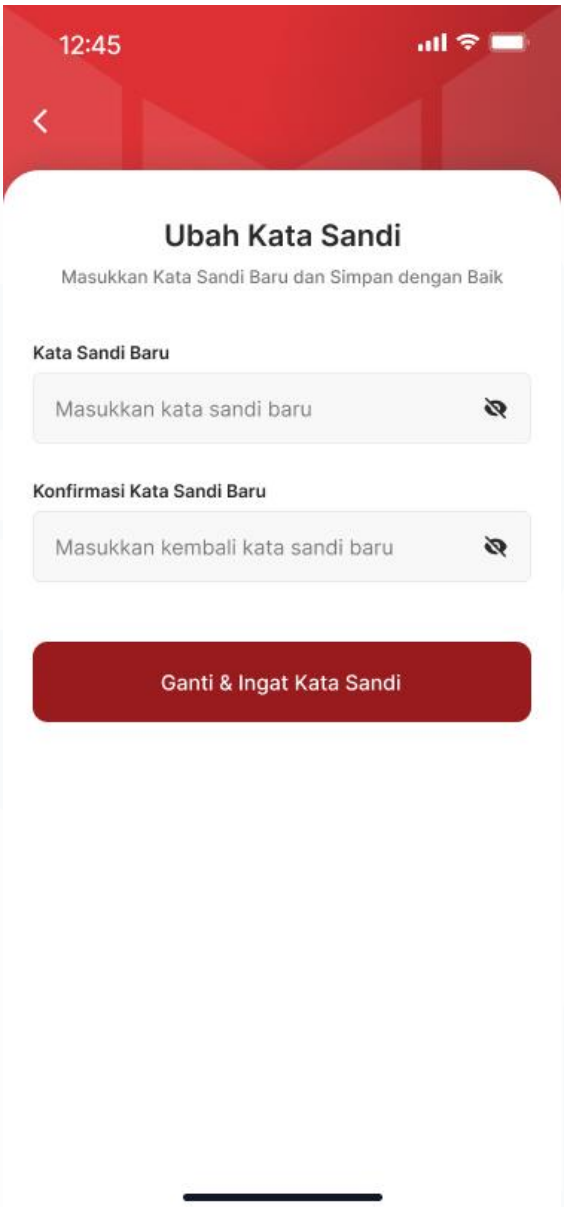
5. *Forget Password Code Verification Screen*



Gambar 3.7 *Reset Code Screen*

Pada gambar 3.7 menampilkan halaman untuk memasukkan kode verifikasi yang dikirim ke email atau nomor HP pengguna sebagai langkah keamanan sebelum melakukan reset password.

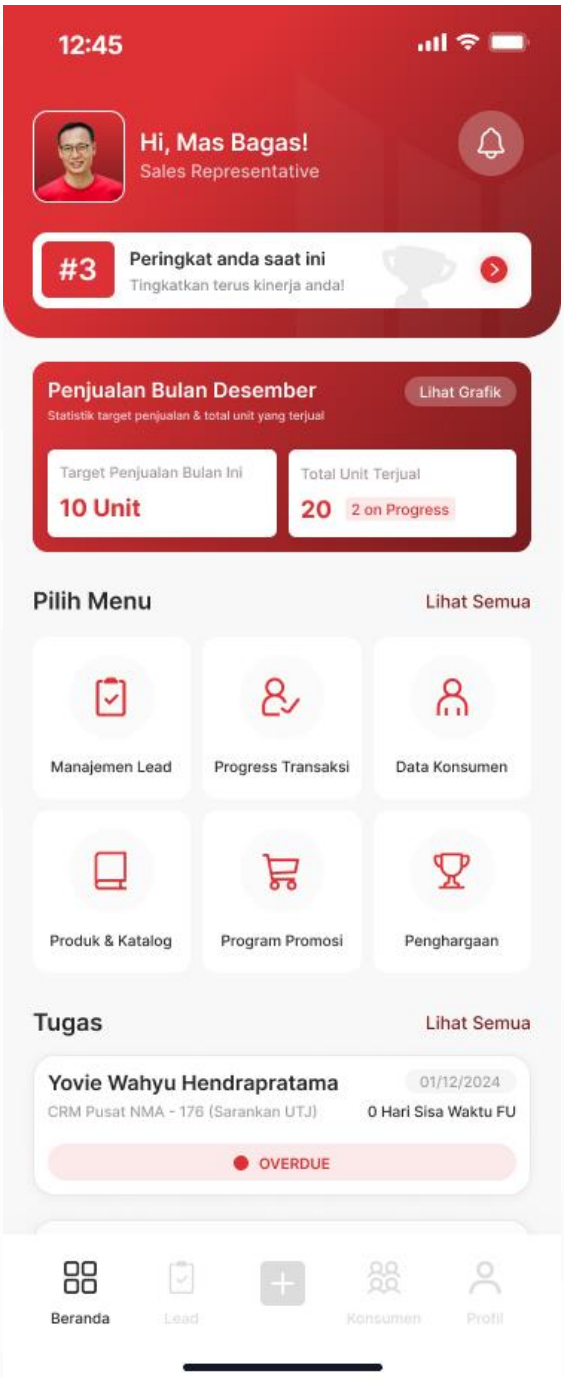
6. *Reset Password Screen*



Gambar 3.8 *Reset Password Screen*

Pada gambar 3.8 menampilkan halaman untuk mengatur ulang kata sandi baru setelah berhasil melewati verifikasi kode keamanan.

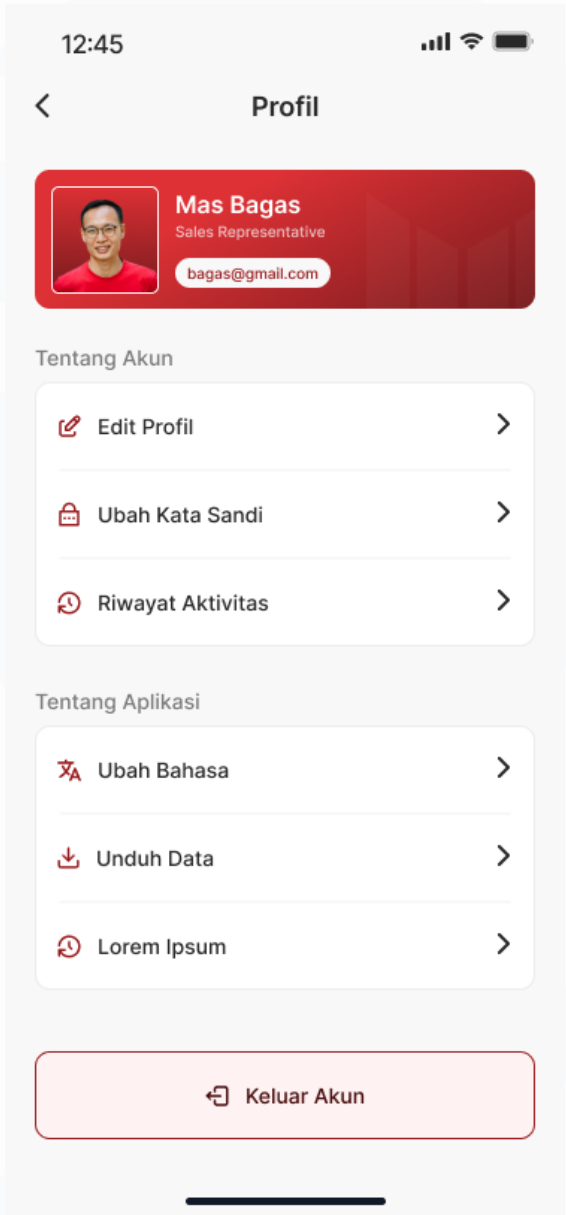
7. Beranda *Screen*



Gambar 3.9 *Beranda Screen*

Pada gambar 3.9 menampilkan halaman utama setelah pengguna berhasil login. Berisi ringkasan fitur utama dan data penting seperti jumlah lead, progress transaksi, dan promosi yang sedang berlangsung.

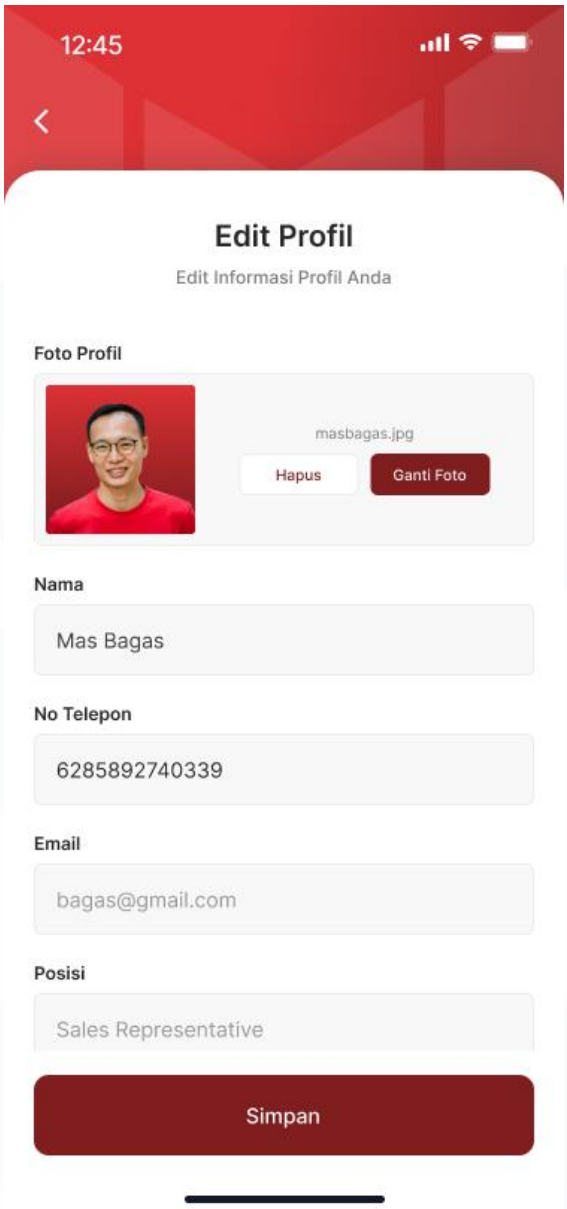
8. *Profile Screen*



Gambar 3.10 *Profile Screen*

Pada gambar 3.10 menampilkan halaman yang menampilkan informasi pribadi pengguna, seperti nama, email, nomor HP, dan peran dalam aplikasi.

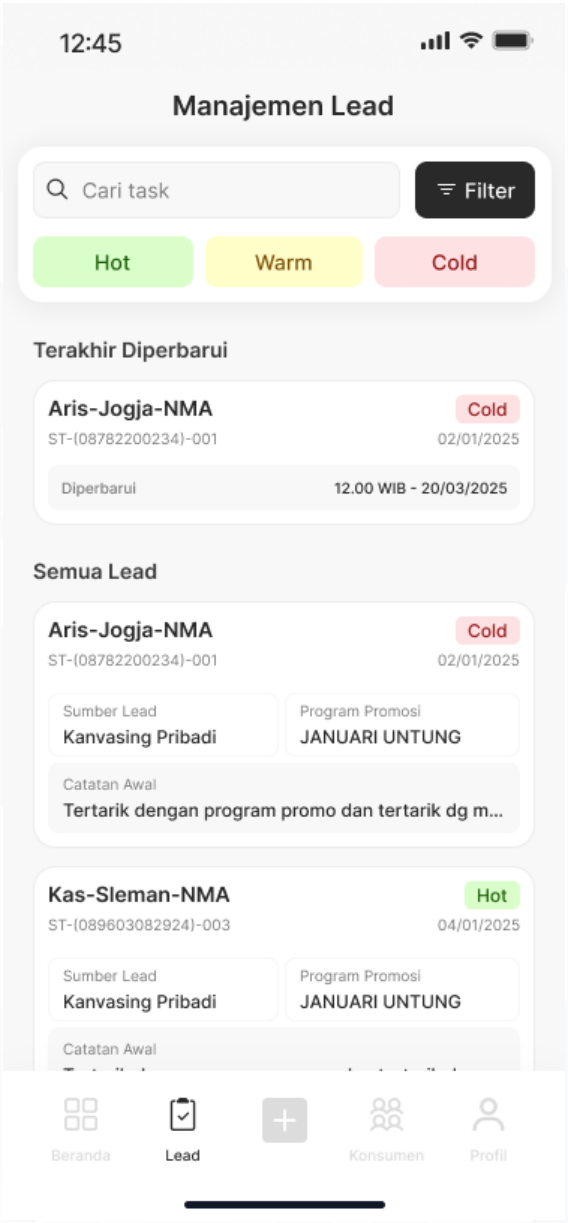
9. *Edit Profile Screen*



Gambar 3.11 *Edit Profile Screen*

Pada gambar 3.11 menampilkan halaman untuk mengubah informasi profil pengguna, seperti foto profil, nama, nomor HP, dan password.

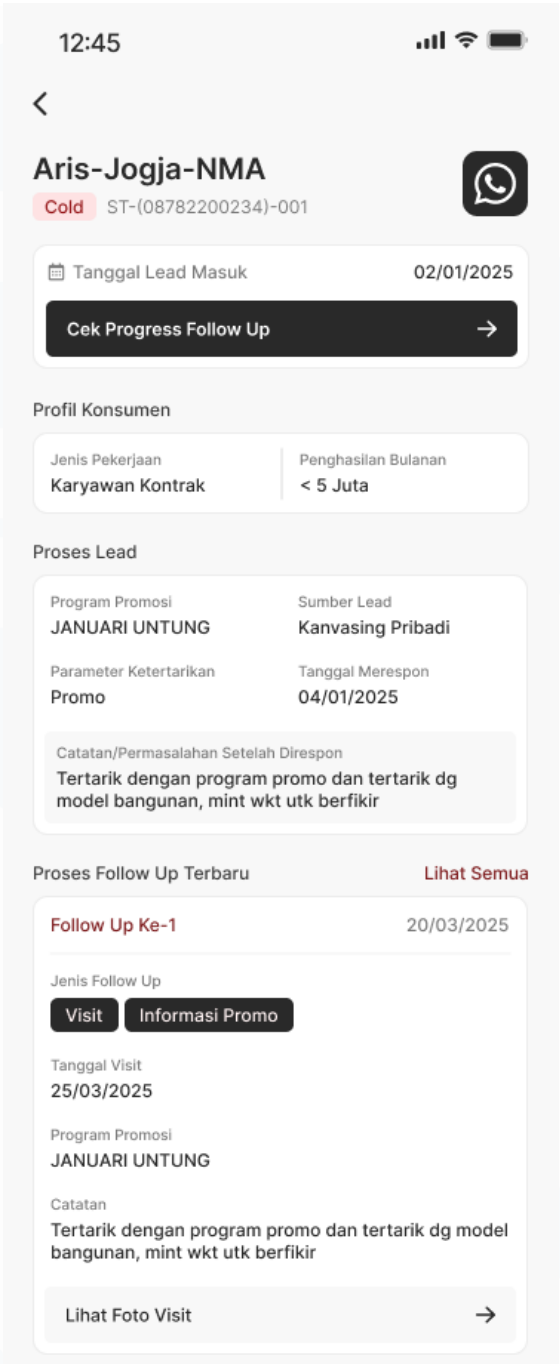
10. Manajemen Lead Screen



Gambar 3.12 *Lead Management Screen*

Pada gambar 3.12 menampilkan halaman yang menampilkan daftar lead dalam bentuk list dengan fitur search dan filter untuk memudahkan pencarian lead berdasarkan status atau sumber.

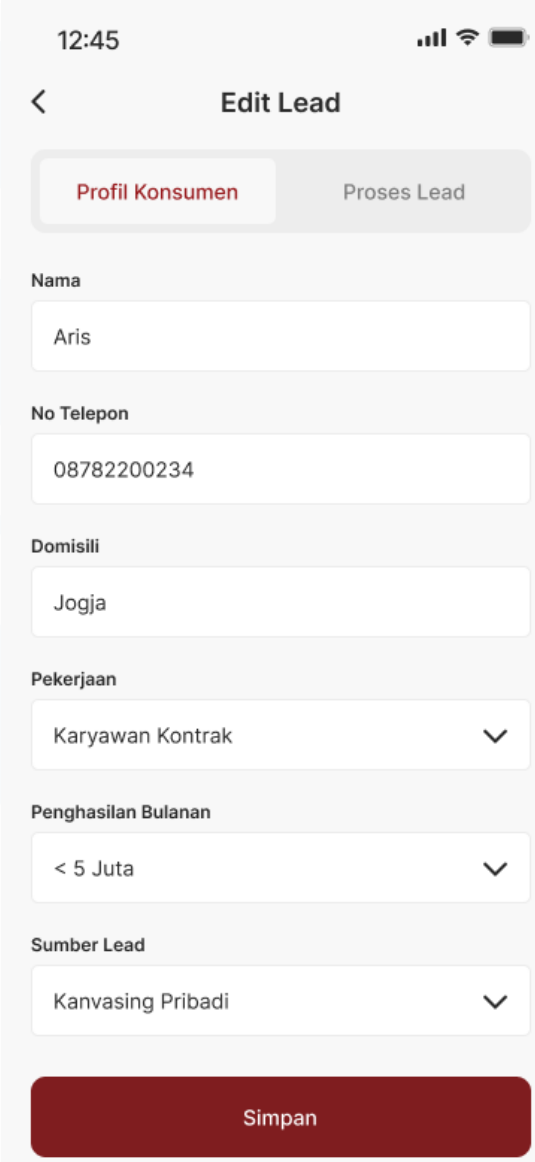
11. Detail Lead Screen



Gambar 3.13 *Lead Management Detail Screen*

Pada gambar 3.13 menampilkan halaman yang menampilkan informasi lengkap tentang lead tertentu, seperti nama, kontak, sumber lead, status, serta histori follow-up yang dilakukan.

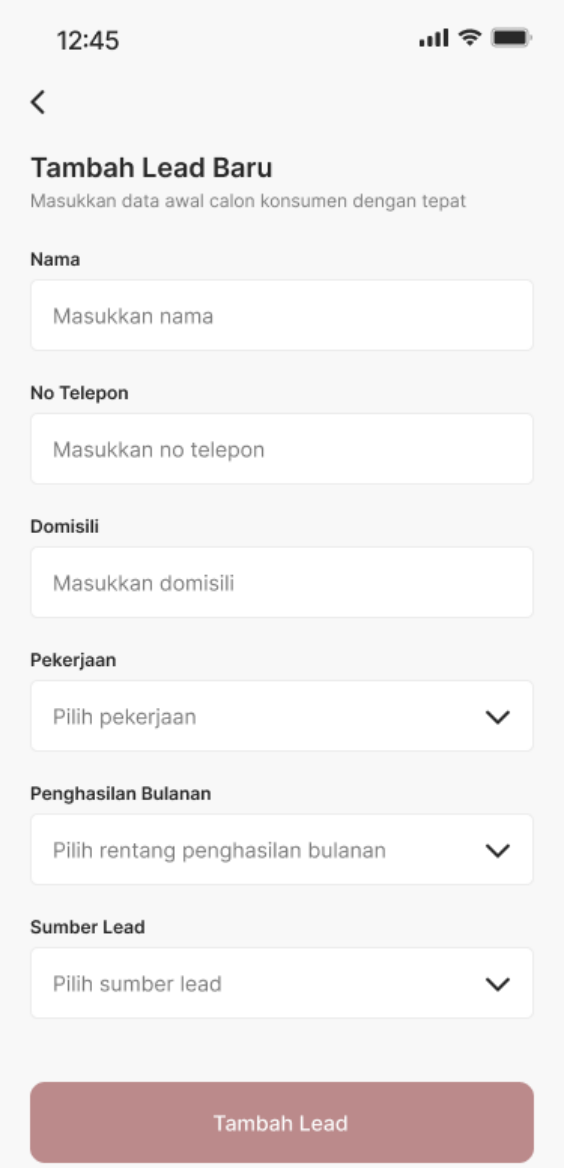
12. Edit Lead Screen



Gambar 3.14 *Edit Lead Management Screen*

Pada gambar 3.14 menampilkan halaman untuk mengedit lead dengan mengisi formulir yang berisi nama, kontak, sumber lead, status, dan informasi lainnya.

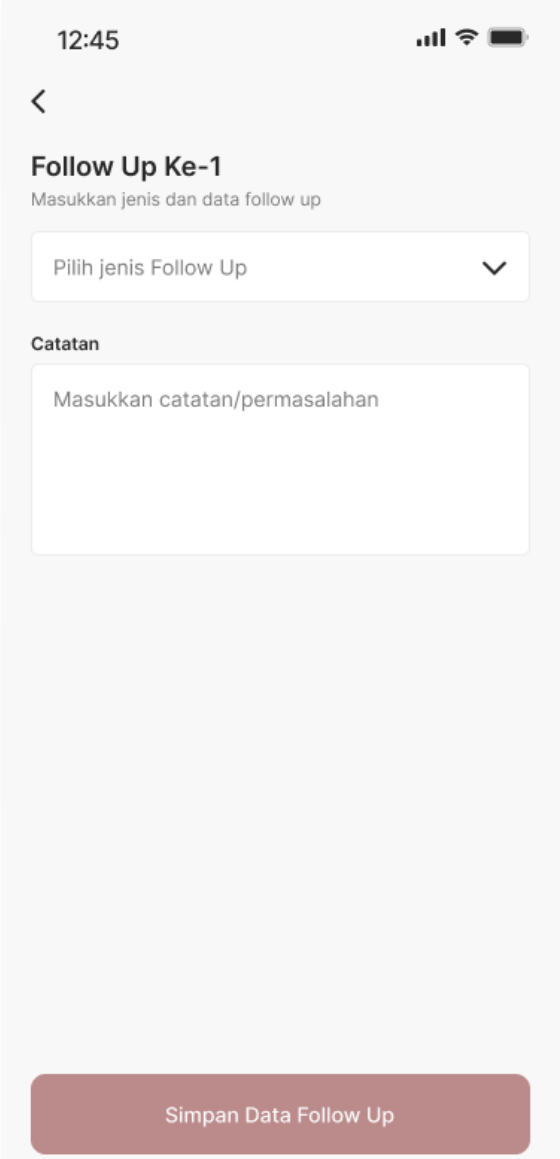
13. Tambah Lead Screen



Gambar 3.15 *Add New Lead Screen*

Pada gambar 3.15 menampilkan halaman untuk menambahkan lead baru dengan mengisi formulir yang berisi nama, kontak, sumber lead, status, dan informasi lainnya.

14. Add Follow Up Screen



Gambar 3.16 *Add Follow Up Screen*

Pada gambar 3.16 menampilkan halaman untuk menambahkan follow-up terhadap lead, misalnya mengisi catatan hasil komunikasi, status kunjungan, atau tindakan yang harus diambil selanjutnya.

15. List Follow Up Screen

Gambar 3.17 *Empty State Follow Up Screen*

Pada gambar 3.17 menampilkan halaman yang menampilkan daftar follow-up yang telah dilakukan terhadap lead tertentu, termasuk catatan, tanggal *follow-up*, dan statusnya.

Pada pengembangan *frontend* penulis menggunakan folder structure sebagai berikut:

MIRA-APP
- /app
  - /(App)
    - /Mira
  - /(Auth)
- /assets
  - /font
  - /images
  - /languages
- /components
- /constants
- /hooks
- /utils

Berikut penjelasan tentang setiap folder yang digunakan

1. App

   Folder app berisi file utama aplikasi, seperti halaman (screens) dan navigasi. Di dalamnya terdapat berbagai komponen yang menyusun tampilan dan fungsionalitas utama dari aplikasi. Penulis juga membuat 2 sub-folder didalam folder app untuk membedakan screen yang dapat diakses pengguna yang telah login (App) dan yang belum login (Auth)
2. Assets

   Folder ini menyimpan berbagai aset statis seperti gambar, ikon, font dan terjemahan bahasa yang digunakan dalam aplikasi untuk meningkatkan tampilan antarmuka pengguna.

3. Components

   Folder ini digunakan untuk menyimpan beberapa komponen UI yang dapat digunakan kembali, seperti tombol, kartu, input field, dan modal. Dengan adanya folder ini, pengembangan menjadi lebih modular dan efisien.

4. Constants

   Folder ini digunakan untuk menyimpan nilai tetap (*constant*) yang digunakan di berbagai bagian aplikasi, seperti daftar status lead, konfigurasi API, atau warna tema agar lebih mudah dikelola dan diubah.

5. Hooks

   Folder ini digunakan untuk menyimpan custom hooks yang dibuat untuk mengelola logika khusus dalam aplikasi, seperti pengambilan data dari API, manajemen state, atau interaksi dengan perangkat. Dengan adanya hooks, kode menjadi lebih bersih dan terorganisir.

6. Utils

   Folder ini berisi fungsi-fungsi bantu (*utility functions*) yang digunakan di berbagai bagian aplikasi, seperti api, response formatter, state, store, zod dan lain-lain.

Berikut adalah penjelasan singkat mengenai library yang digunakan dalam pengembangan aplikasi MIRA:

1. Tamagui [1]

Tamagui adalah sebuah UI library yang memungkinkan pengembang untuk membangun tampilan aplikasi yang cepat, responsif, dan ringan. Library ini mendukung *cross-platform* development, sehingga desain aplikasi bisa lebih seragam di berbagai perangkat, termasuk *mobile* dan *web*.

2. Zustand [2]

Zustand adalah *state management library* yang lebih ringan dan sederhana dibandingkan Redux. Dengan API yang minimalis, Zustand

memudahkan pengembang dalam mengelola *global state* tanpa perlu banyak boilerplate code. Selain itu, Zustand juga memiliki fitur *reactive updates* yang efisien.

3. Expo Router [3]

Expo Router adalah sistem navigasi berbasis *file routing* yang diadaptasi dari Next.js untuk aplikasi *React Native*. Library ini mempermudah pengelolaan navigasi dalam aplikasi dengan struktur yang lebih terorganisir dan mendukung konsep *deep linking* serta *server-side rendering* (SSR) untuk aplikasi *web*.

4. Axios [4]

Axios adalah pustaka HTTP berbasis *Promise* yang digunakan untuk melakukan *request* ke backend. Dibandingkan dengan Fetch API bawaan JavaScript, Axios lebih fleksibel karena memiliki fitur tambahan seperti *automatic request cancellation*, *interceptors*, serta dukungan untuk *transforming responses*.

5. React Hook Form [5]

React Hook Form adalah pustaka yang digunakan untuk menangani *form validation* dan manajemen input dalam aplikasi React. Library ini dioptimalkan untuk mengurangi *re-rendering*, meningkatkan performa, serta memberikan pengalaman pengguna yang lebih baik dalam mengisi formulir.

6. React Query [6]

React Query adalah pustaka untuk mengelola pengambilan dan penyimpanan data dari API secara otomatis. Dengan fitur *caching*, *synchronization*, dan *background data fetching*, React Query dapat meningkatkan performa aplikasi dengan mengurangi jumlah permintaan API yang tidak perlu.

7. Zod [7]

Zod adalah pustaka untuk validasi skema data berbasis TypeScript. Dengan Zod, pengembang dapat memastikan bahwa data yang dikirimkan sesuai dengan format yang diharapkan sebelum diproses

lebih lanjut. Ini sangat berguna dalam meningkatkan keamanan dan kestabilan aplikasi.

3.2.4 Pengembangan *Backend*

Dalam pengembangan *backend* aplikasi MIRA, penulis menggunakan beberapa teknologi dan library untuk memastikan sistem berjalan dengan baik. *Backend* dikembangkan menggunakan RESTful API, yang memungkinkan komunikasi yang efisien antara frontend dan *backend* melalui endpoint yang jelas. Express.js digunakan sebagai framework backend karena ringan dan mudah digunakan untuk membangun API. Untuk mengelola *database*, Prisma ORM digunakan yang mempermudah interaksi dengan *database* MySQL. Bcryptjs digunakan untuk mengenkripsi password, sementara jsonwebtoken (JWT) digunakan untuk sistem autentikasi pengguna. Selain itu, Multer membantu dalam menangani unggahan file, dan Cors digunakan untuk mengizinkan komunikasi antara backend dan frontend. Dalam pengembangan, Nodemon digunakan untuk memantau perubahan pada kode sehingga server bisa diperbarui secara otomatis tanpa harus restart manual.

Pada bagian *backend*, digunakan folder structure seperti berikut:

```
/MIRA-BACKEND
  /prisma
  /src
      /config
      /constants
      /controllers
      /lib
      /middleware
      /routes
      /schemas
```

Berikut penjelasan tentang setiap folder yang digunakan:

1. Config

   Folder ini berisi file konfigurasi yang digunakan dalam aplikasi seperti konfigurasi JWT Token yang akan digunakan.

2. Constants

   Berisi nilai-nilai tetap yang digunakan dalam aplikasi, seperti status transaksi, atau pesan kesalahan standar sehingga kode menjadi lebih rapi..

3. Controllers

   Folder ini menyimpan file yang berisi logika utama dari setiap fitur di aplikasi, seperti mengelola data pengguna, transaksi, dan proses autentikasi. Controller bertugas menangani permintaan dari client dan memberikan respons yang sesuai.

4. Lib

   Berisi fungsi atau modul tambahan yang digunakan dalam aplikasi, seperti helper functions atau konfigurasi library tertentu agar bisa digunakan ulang di berbagai bagian kode.

5. Middleware

   Folder ini berisi middleware, yaitu fungsi perantara yang dijalankan sebelum request diproses oleh controller. Contohnya, autentikasi dengan JWT, validasi data, atau pengaturan CORS.

6. Routes

   Berisi semua rute atau endpoint API yang digunakan dalam aplikasi. Setiap file di dalamnya menangani berbagai kategori fitur, seperti rute untuk autentikasi, manajemen lead, atau transaksi.

7. Schemas

   Folder ini berisi skema validasi data menggunakan Zod. Setiap request yang masuk akan divalidasi sesuai dengan skema yang telah ditentukan untuk memastikan data yang diterima sesuai dengan standar yang diharapkan.

Pada pengembangan backend, ada beberapa API yang akan digunakan yaitu::

1. Auth

   menangani proses autentikasi dan manajemen pengguna dalam sistem. Controller ini menyediakan beberapa API penting:

   a. Register API (POST /api/auth/register): Memungkinkan pendaftaran pengguna baru dengan peran yang berbeda (Admin atau Sales). Fungsi ini memvalidasi data pengguna, memeriksa keberadaan email, mengenkripsi password, dan membuat token JWT untuk autentikasi.

   b. Login API (POST /api/auth/login): Menangani proses login pengguna dengan memverifikasi kredensial (email dan password). Setelah verifikasi berhasil, API ini mengembalikan token JWT dan informasi pengguna termasuk gambar profil dalam format base64.

   c. Logout API (POST /api/auth/logout): Menangani proses logout pengguna dari sistem.

   d. Change Password API (POST /api/auth/changePassword): Memungkinkan pengguna untuk mengubah password mereka dengan memverifikasi password lama sebelum menggantinya dengan password baru yang terenkripsi.

2. Admin Controllerm

   Admin Controller berfokus pada manajemen pengguna admin dalam sistem. Controller ini menyediakan API berikut:

   a. Get Admins API (GET /api/admins): Mengambil daftar semua admin dalam sistem beserta informasi peran mereka.

   b. Create Admin API (POST /api/admins): Memungkinkan pembuatan akun admin baru dengan validasi untuk memastikan

email belum digunakan sebelumnya dan mengenkripsi password untuk keamanan.

3. Lead Controller

   Lead Controller menangani manajemen prospek (lead) dalam sistem CRM. Controller ini menyediakan beberapa API penting:

   a. Create Lead API (POST /api/leads/createLead): Membuat lead baru dengan informasi seperti nama, domisili, nomor telepon, pekerjaan, dan pendapatan bulanan. API ini juga menghasilkan ID lead unik dan memvalidasi data seperti keberadaan pekerjaan dan cluster.
   b. Get Leads API (GET /api/leads/getLeads): Mengambil daftar semua lead dalam sistem beserta informasi terkait seperti manajemen lead, pekerjaan, dan cluster.
   c. Get Lead By ID API (GET /api/leads/getLeadById/:leadId): Mengambil informasi detail tentang lead berdasarkan ID lead.
   d. Update Lead API (PUT /leads/updateLead/:id): Memperbarui informasi lead yang ada dengan validasi untuk memastikan integritas data.
   e. Get Leads By Sales API (GET /api/leads/getLeadBySales): Mengambil daftar lead yang ditugaskan kepada sales tertentu, memungkinkan sales untuk melihat lead yang menjadi tanggung jawab mereka.

4. Lead Management Controller

   Lead Management Controller menangani pengelolaan lead yang lebih kompleks, termasuk penugasan dan pelacakan status. Controller ini menyediakan API berikut:

   a. Get All Lead Management API (GET /api/leadManagement/getAllLeadManagements): Mengambil semua data manajemen lead dalam sistem dengan informasi terkait seperti lead, sales, promo, dan follow-up.

b. Get Lead Management By Sales API (GET /api/leadManagement/getAllLeadManagementBySales): Mengambil data manajemen lead yang ditugaskan kepada sales tertentu.

c. Get Latest Lead Managements API (GET /api/leadManagement/getLatestLeadManagement): Mengambil data manajemen lead terbaru untuk sales tertentu, membantu sales fokus pada lead yang paling aktif.

d. Create Lead Management API (POST /api/leadManagement/createLeadMangement): Membuat entri manajemen lead baru dengan menghubungkan lead ke sales dan promo tertentu.

e. Update Lead Management API (PATCH /api/leadManagement/updateLeadManagement/:id): Memperbarui informasi manajemen lead seperti status, minat, dan catatan.

5. Follow Up Controller

Follow Up Controller menangani aktivitas tindak lanjut (follow-up) terhadap lead. Controller ini menyediakan API berikut:

a. Create Follow Up API (POST /api/followUps/createNewFollowUp): Membuat catatan follow-up baru dengan berbagai tipe seperti kunjungan lokasi, informasi promo, cek SLIK, proses booking, atau follow-up lainnya.

b. Get Follow Ups API (GET /api/followUps/getFollowUp): Mengambil daftar follow-up untuk manajemen lead tertentu.Get Follow Up By ID API: Mengambil detail follow-up berdasarkan ID.

c. Update Follow Up API (PUT /api/followUps/getFollowUpById/:id): Memperbarui informasi follow-up yang ada.
d. Delete Follow Up API (DELETE /api/followUps/deleteFollowUp/:id): Menghapus catatan follow-up dari sistem.

6. Job Controller

Job Controller menangani manajemen jenis pekerjaan dalam sistem. Controller ini menyediakan API berikut:

a. Create Job API (POST /api/jobs/createJob): Membuat jenis pekerjaan baru dengan label dalam bahasa Inggris dan Indonesia.
b. Get All Jobs API (GET /api/jobs/getAllJobs): Mengambil daftar semua jenis pekerjaan dalam sistem.
c. Update Job API (POST /api/jobs/updateJob/:id): Memperbarui informasi jenis pekerjaan yang ada.
d. Delete Job API (DELETE /api/jobs/deleteJob/:id): Menghapus jenis pekerjaan dari sistem dengan validasi untuk memastikan tidak ada lead atau promo yang menggunakan jenis pekerjaan tersebut.

7. Profile Controller

Profile Controller menangani manajemen profil pengguna. Controller ini menyediakan API berikut:

a. Get Profile API (GET /api/profile/getProfile): Mengambil informasi profil pengguna berdasarkan token JWT.
b. Update Profile API (UPDATE /api/profile/updateProfile): Memperbarui informasi profil pengguna seperti nama, nomor telepon, email, dan gambar profil.

8. Promo Controller

Promo Controller menangani manajemen promosi dalam sistem. Controller ini menyediakan API berikut:

a. Get Promos API (GET /api/promos/getPromos): Mengambil daftar semua promosi dalam sistem.
b. Create Promo API (POST /api/promos/createPromo): Membuat promosi baru dengan informasi seperti nama, deskripsi, tanggal mulai dan berakhir, serta jenis pekerjaan yang ditargetkan.
c. Delete Promo API (DELETE /api/promos/deletePromo/:id): Menghapus promosi dari sistem dengan validasi untuk memastikan promosi tidak sedang digunakan.
d. Toggle Promo Status API (PATCH /api/promos/togglePromoStatus/:id): Mengaktifkan atau menonaktifkan promosi.

9. Task Controller

Task Controller menangani manajemen tugas antara admin dan sales. Controller ini menyediakan API berikut:

a. Create Task API (POST /api/tasks/createNewTask): Membuat tugas baru dari admin ke sales dengan informasi seperti nama tugas, deskripsi, dan tenggat waktu.
b. Get Tasks API (GET /api/tasks/getTasks): Mengambil daftar tugas berdasarkan peran pengguna (admin melihat tugas yang mereka buat, sales melihat tugas yang ditugaskan kepada mereka).
c. Update Task Status API (PATCH /api/tasks/updateTaskStatus/:id): Memperbarui status tugas dengan validasi untuk memastikan hanya sales yang ditugaskan yang dapat memperbarui status.

10. Cluster Controller

Cluster Controller menangani manajemen cluster atau kelompok properti dalam sistem. Controller ini menyediakan API berikut:

a. Create Cluster API (POST /api/cluster/createNewCluster): Membuat cluster baru dengan validasi untuk memastikan nama cluster unik.
b. Get All Clusters API (GET /api/cluster/getAllClusters): Mengambil daftar semua cluster dalam sistem.
c. Get Cluster By ID API (GET /api/cluster/getClusterById/:id): Mengambil informasi detail tentang cluster berdasarkan ID, termasuk lead yang terkait.
d. Update Cluster API (PUT /api/cluster/updateCluster/id): Memperbarui informasi cluster yang ada dengan validasi untuk memastikan nama cluster tetap unik.
e. Delete Cluster API (DELETE /api/cluster/deleteCluster/:id): Menghapus cluster dari sistem dengan validasi untuk memastikan tidak ada lead yang terkait dengan cluster tersebut.

3.3 Kendala yang Ditemukan

Selama pengembangan MIRA, penulis menghadapi beberapa kendala. Kendala-kendala ini muncul selama proses pengembangan dan menjadi bagian dari perjalanan dalam membangun sistem MIRA. Berikut adalah beberapa kendala yang dihadapi:

1. Komunikasi antara divisi Marketing dan IT terbatas jarak dan waktu karena divisi Marketing berada di kantor pusat di Yogyakarta dan divisi marketing memiliki waktu yang terbatas. Hal ini mengakibatkan proses pengembangan MIRA sedikit terhambat karena penyampaian informasi hanya bisa dilakukan pada saat divisi marketing kosong melalui google meet.
2. Dalam proses pengembangan MIRA, kebutuhan sistem bisa berubah seiring waktu karena berbagai faktor seperti masukan dari pengguna, perubahan strategi bisnis sehingga penulis harus sering melakukan revisi atau penyesuaian yang mempengaruhi jadwal proyek.

3.4 Solusi atas Kendala yang Ditemukan

Selama proses pengembangan MIRA, terdapat beberapa kendala yang dihadapi. Untuk mengatasi kendala-kendala tersebut, berikut adalah solusi yang dapat diterapkan:

1. Meningkatkan komunikasi secara fleksibel dan terdokumentasi. Menjadwalkan meeting rutin dengan waktu yang lebih fleksibel (misalnya 2-3 kali seminggu) agar komunikasi lebih terstruktur dan tidak bergantung pada waktu luang divisi Marketing. Selain itu, menggunakan platform komunikasi asinkron seperti Slack, Trello, atau Notion untuk mendokumentasikan pertanyaan dan update proyek, sehingga divisi Marketing bisa memberikan masukan kapan saja tanpa harus melakukan meeting langsung.
2. Mendokumentasikan dan memprioritaskan perubahan kebutuhan sistem. Setiap revisi atau perubahan fitur harus dicatat dengan detail dalam dokumen pengembangan agar tim IT tidak kebingungan dan bisa menyesuaikan jadwal dengan lebih baik. Selain itu, perlu dilakukan evaluasi untuk menentukan fitur mana yang harus segera diimplementasikan dan mana yang bisa ditunda agar proyek tetap berjalan sesuai jadwal.