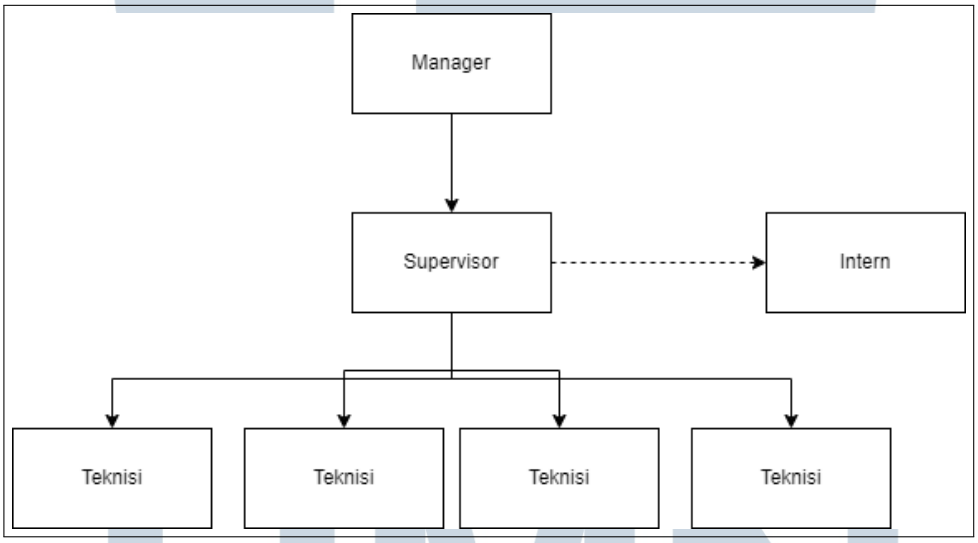
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

PT. Satyaraya Keramindoindah (PT. SRKI) merupakan perusahaan yang bergerak di bidang manufaktur dan distribusi keramik. Dalam struktur organisasi, divisi TI memiliki peran strategis dalam mendukung kelancaran operasional perusahaan, terutama terkait pengelolaan perangkat keras, jaringan komputer, dan sistem teknologi informasi.



Gambar 3.1. Struktur organisasi perusahaan PT. Satyaraya Keramindoindah

Berdasarkan Gambar 3.1, tim TI terdiri dari *Manager*, *Supervisor*, dan 4 Teknisi. Selama praktik kerja magang, penugasan diberikan dan dibimbing langsung oleh Bapak Albertus Febi selaku *Supervisor* tepatnya untuk fokus pada pengembangan aplikasi helpdesk berbasis web.

3.2 Tugas yang Dilakukan

Selama empat bulan praktik kerja magang, adapun tugas-tugas yang dilaksanakan adalah sebagai berikut.

1. Mempelajari dan membiasakan lingkungan kerja serta observasi teknisi yang praktik.
2. Pengembangan Aplikasi *Helpdesk* yang dipandu oleh *Supervisor*.

3. Merancang Flowchart, ERD, Use Case dan Data Flow Diagram Aplikasi Helpdesk.
4. Menyusun *Framework* untuk *backend* dan *frontend* Aplikasi yang akan di pakai
5. Presentasi progress kepada *Supervisor* dan uji coba ke *client server*
6. Proses *deployment*
7. Perbaikan *bug* pasca *deployment*

3.3 Uraian Pelaksanaan Magang

Berikut adalah detail terkait kegiatan yang dilakukan selama praktik kerja magang. Adapun mulai dari pembentukan *flowchart* sampai tahap *testing* dan *deplyoment*.

Tabel 3.1. Uraian pelaksanaan praktik kerja magang

Minggu Ke -	Pekerjaan yang dilakukan
1	• Pengenalan lingkungan kerja PT. SRKI secara spesifik bagian TI. • Memahami kebutuhan teknisi TI terkait sistem *helpdesk*. • Merancang *flowchart, ERD, DFD (Data Flow Diagram)*, dan *Use Case Diagram*.
2	• Menentukan *framework* dan bahasa pemrograman yang akan digunakan (Express.js dan React.js). • Mempelajari materi yang diperlukan untuk pengembangan aplikasi. • Mulai implementasi *database* dan memanggil dengan menggunakan *framework Express*.
3	• Menyusun *database* dengan menggunakan MySQL serta mengembangkan backend lebih lanjut. • Menyusun *backend* sistem krerasi tiket • Berdiskusi dengan *Supervisor* mengenai kode backend serta *Supervisor* memberi feedback mengenai program *backend*.
4	• Testing koneksi *backend* untuk memastikan bahwa sistem operasional. • Presentasi *layout UI* kepada rekna-rekan teknisi dan *Supervisor* serta membahas fitur-fitur yang perlu di tambahkan.

Minggu Ke -	Pekerjaan yang dilakukan
5	• Menambahkan fitur Auto-close tiket, di mana jika user lupa mengkonfirmasi, tiket akan tutup secara otomatis dalam kurun 3 hari. • Merapikan *layout UI* serta presentasi UI dan fitur-fitur yang telah di tambahkan.
6	• Penyesuaian *User Interface* dan mengurangi layout atau design yang tidak perlu. • Mengurangi fitur *key performance index* dalam akun manager. • Uji coba aplikasi secara keseluruhan menggunakan localhost.
7	• Diskusi dengan *supervisor* soal fitur-fitur sistem tiket yang perlu ditambahkan dan diperbaiki. • Presentasi sistem aplikasi kepada rekan-rekan teknisi beserta *Supervisor* memberikan *feedback*.
8	• Penyesuaian *feedback* dari *Supervisor*. • Penambahan fitur total tiket dan perbaikan sistem *auto-close*. • Review ulang kode dan memperbaiki bug yang ada.
9	• Diskusi dengan *supervisor* soal tata cara koneksi ke *client server*. Uji coba aplikasi dijalankan dengan client server.

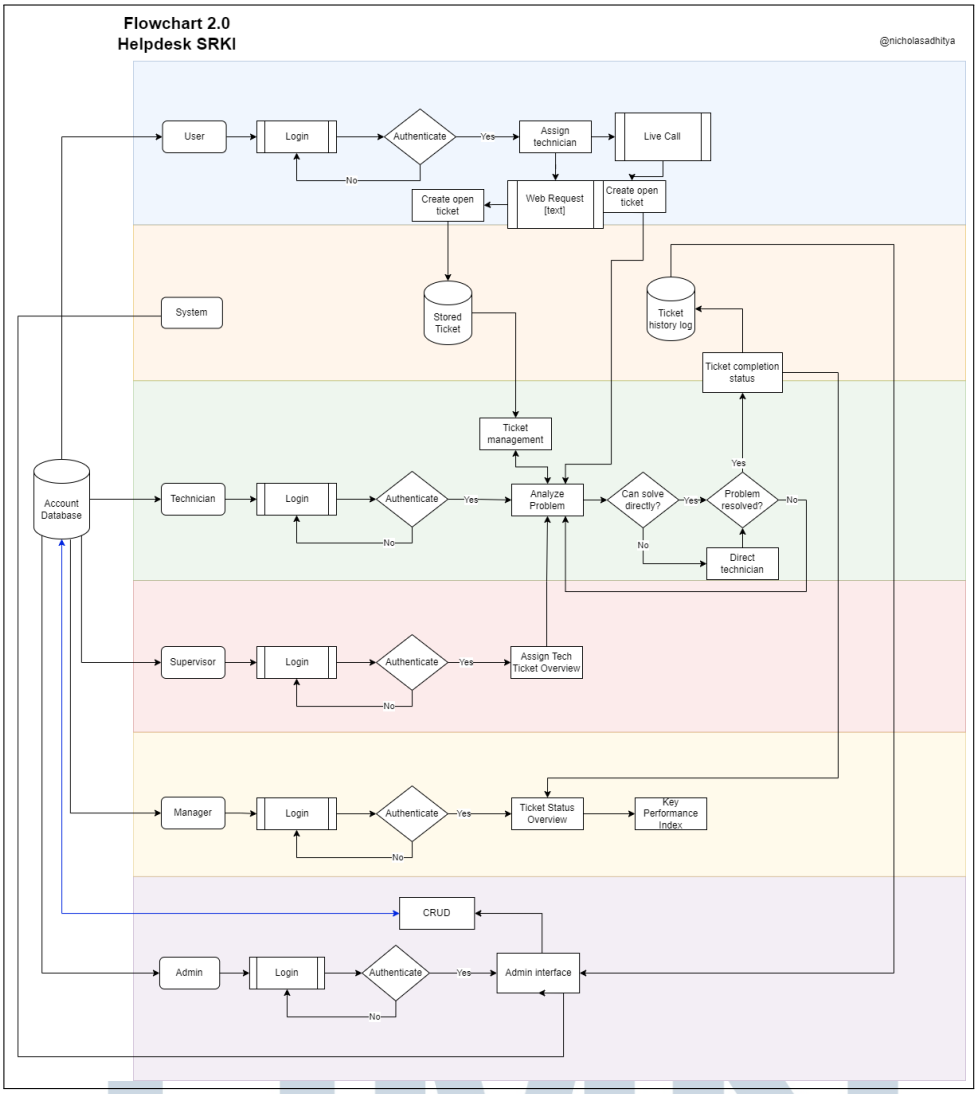
Minggu Ke -	Pekerjaan yang dilakukan
10	• *Deployment* dan Uji Coba Aplikasi Helpdesk beserta penjelasan ulang fitur-fitur aplikasi kepada rekan-rekan TI dan feedback untuk fitur yang masih perlu di tambahkan.

3.3.1 Uraian Masalah dan Kebutuhan

Sebelum membuat aplikasi *helpdesk* untuk membantu kelancaran komunikasi teknisi TI dengan departemen lain. Hubungan antara departemen TI dengan departemen lain di PT. SRKI saat ini masih dilakukan secara manual, dengan komunikasi yang bergantung pada panggilan telepon atau email untuk menyampaikan masalah yang perlu ditangani. Metode ini sering kali menyulitkan dalam melacak perkembangan penyelesaian masalah yang dilaporkan. Maka dari itu, *supervisor* memutuskan untuk menginisiasi pengembangan sebuah sistem yang memungkinkan pelacakan masalah-masalah tersebut dalam bentuk tiket. Sistem ini diharapkan dapat mempermudah komunikasi antar departemen sekaligus meningkatkan efisiensi dalam menyelesaikan permasalahan yang ada.

3.3.2 *Flowchart* Aplikasi Helpdesk PT. SRKI

Berikut merupakan *flowchart* yang digunakan sebagai dasar alur pada Aplikasi Helpdesk PT. SRKI.



Gambar 3.2. *Flowchart* Aplikasi *Helpdesk*

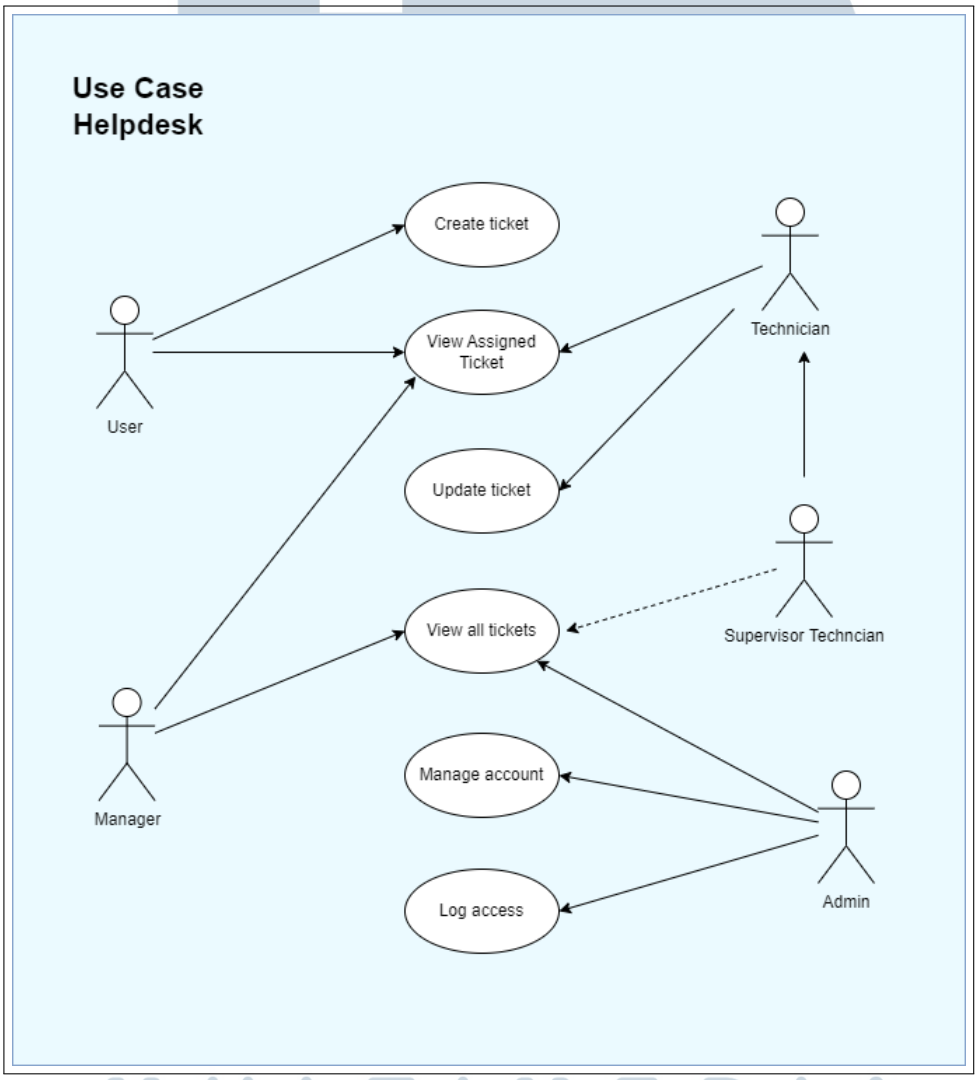
Tabel Gambar 3.1, menjelaskan *flowchart* aplikasi *helpdesk* PT. SRKI. Berikut merupakan alur proses keseluruhan dari aplikasi Helpdesk ini.

1. User melaporkan masalah melalui tiket.
2. Tiket masuk ke dalam sistem, yang meneruskan ke *supervisor* untuk pengalokasian teknisi.
3. Teknisi menangani tiket, memperbarui status, dan menyelesaikan tugas.
4. User memverifikasi penyelesaian dengan menutup tiket, atau tiket ditutup otomatis oleh sistem.
5. *Admin* dan *manager* dapat memantau data dan progres tiket untuk keperluan operasional atau strategis.

6. Alur ini memastikan setiap peran memiliki tanggung jawab spesifik dan saling terhubung, menciptakan sistem kerja yang terorganisir dan efisien.

3.3.3 *Use Case Diagram* dan Fitur Aplikasi *Helpdesk* PT. SRKI

A. Use Case Diagram



Gambar 3.3. *Use Case Diagram* Aplikasi Helpdesk

Gambar 3.3 di atas merupakan *Use Case Diagram Helpdesk* yang menunjukkan bagaimana pengguna (aktor) berinteraksi dengan sistem *helpdesk*.

1. *User*

   Berperan untuk membuat tiket beserta melihat status tiket untuk konfirmasi selesai.

2. *Supervisor*

   Supervisor masih satu *role* dengan *technician* tetapi dapat melihat seluruh tiket yang ada dan memberi *assign* teknisi ke teknisi lain secara langsung jika teknisi yang di tuju absen.

3. *Technician*

   Berperan menanggapi tiket yang telah dikirim oleh user dan update status tiket.

4. *Manager*

   Berperan melihat seluruh tiket dan teknisi yang mengerjakan sebagai referensi untuk laporan Key Performance Index.

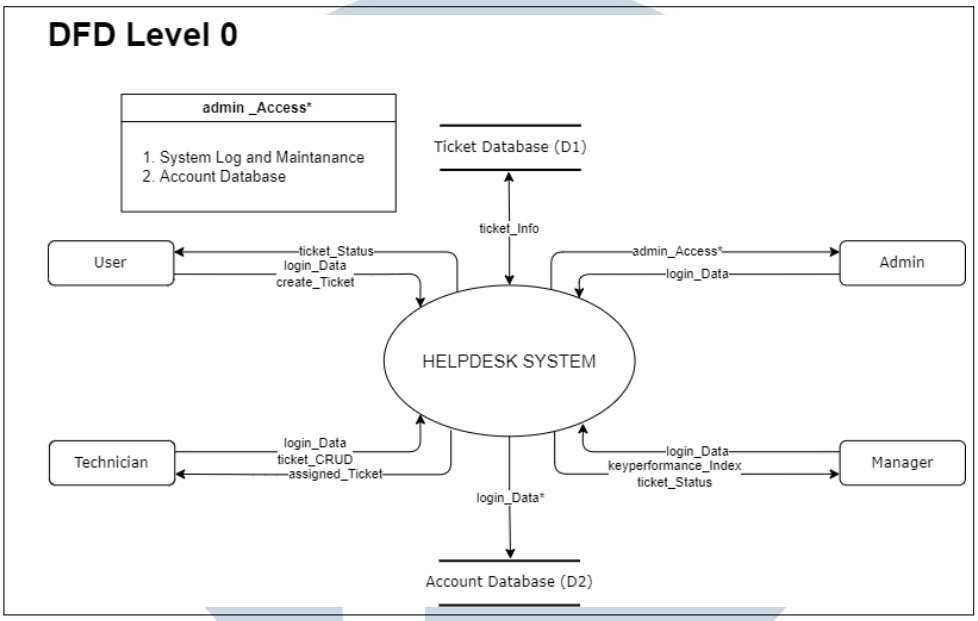
5. *Admin*

   Berperan sebagai kelancaran aplikasi dan melihat log tiket. Menambahkan serta mengedit semua akun user.

B. Fitur Helpdesk

Website helpdesk yang dikembangkan memiliki berbagai fitur yang dirancang untuk meningkatkan efisiensi dan transparansi dalam penanganan masalah. Salah satu fitur utama adalah tracking tanggal, yang mencatat setiap langkah atau perubahan status tiket, termasuk tanggal tiket dibuat, ditangani, dan diselesaikan, sehingga mempermudah pengawasan. Fitur konfirmasi status memungkinkan pengguna untuk melihat status terkini dari tiket mereka, seperti ”Dalam Proses” atau ”Selesai.” Selain itu, ada opsi untuk menetapkan tingkat urgensi, sehingga masalah yang lebih kritis dapat ditangani dengan prioritas lebih tinggi. Fitur *auto close* juga disediakan, di mana tiket akan otomatis ditutup setelah 3 hari jika pengguna lupa menutup tiket yang sudah selesai. Aplikasi ini juga mendukung *chat reply*, memungkinkan komunikasi langsung antara pengguna dan teknisi untuk mendiskusikan detail masalah atau progres penyelesaian. Terakhir, fitur *attachment* memungkinkan pengguna mengunggah *file* pendukung, seperti *screenshot* atau dokumen, sehingga teknisi dapat memahami permasalahan dengan lebih jelas.

3.3.4 *Data Flow Diagram* Aplikasi Helpdesk PT. SRKI



Gambar 3.4. *Data Flow Diagram* Level 0 Aplikasi Helpdesk

Gambar 3.3 merupakan *data flow diagram* level 0 dari aplikasi helpdesk. Semua aliran data akan diatur oleh sistem dan disimpan dalam *database* yang sesuai.

A. Data Flow Diagram Level 0

Berikut merupakan penjelasan DFD Level 0:

1. Dari sisi user, data yang masuk merupakan data *login* (*username* dan *password*)dan kreasi tiket. Jika user telah masuk dan terverifikasi oleh sistem akan menampilkan tiket-tiket yang telah dibuat oleh user beserta informasi status tiket.
2. Sisi *technician* data yang masuk adalah data *login* beserta *update* status tiket yang di mana akan di simpan dalam sistem. *Technician* akan mendapatkan tiket yang telah di buat dari user client.
3. Sisi manager juga memberikan data login dan jika telah terverifikasi, sistem akan menampilkan seluruh tiket beserta status informasinya.

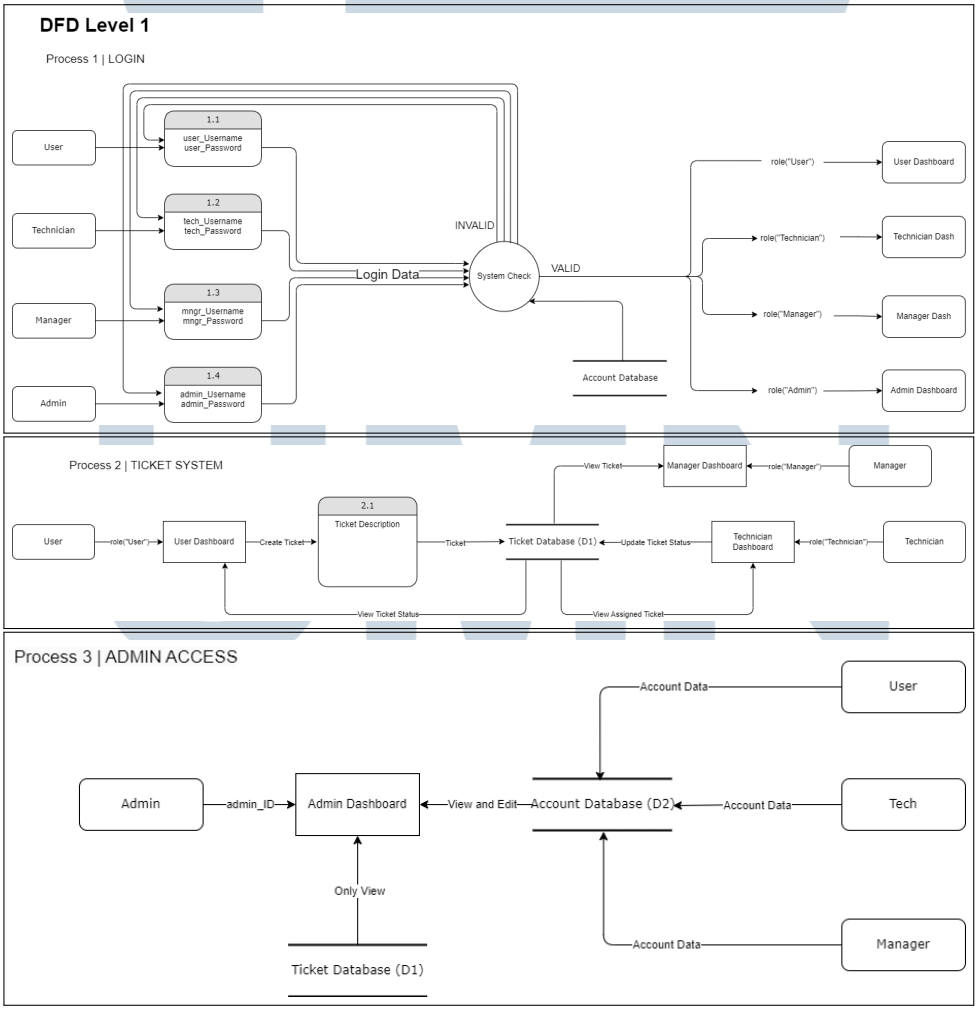
4. Admin memberikan data *login* kepada sistem dan jika telah terverifikasi, sistem akan memberikan seluruh akses admin seperti *Account Database* di mana *admin* bisa memproses seluruh akun dan melihat sistem *log*.

5. Seluruh data *login* akan di verifikasi oleh sistem yang mengecek kredensial di *Account Database*.

6. Seluruh data mengenai tiket yang telah di buat oleh user client akan di simpan dalam *Ticket Database*.

B. Data Flow Diagram Level 1

Gambar di bawah merupakan *Data Flow Diagram* level 1.



Gambar 3.5. *Data Flow Diagram* Level 1 urut dari proses *login*, sistem tiket, dan sistem *admin*

1. Proses 1 Login

   Pada Proses 1 Login, semua user yang akan masuk ke aplikasi akan disambut dengan login. Data yang diinput merupakan username dan password yang nantinya akan di verifikasi oleh sistem dengan mencocokan role kedalam sistem basis data yang memuat semua akun. Jika tidak ada kesalahan input, sistem akan membagi sesuai dengan peran dari yang terdaftar. Misalkan jika peran tertulis sebagai user, maka akan diarahkan ke tampilan untuk user.

2. Proses 2 Sistem Tiket

   User membuat tiket yang nantinya akan di simpan dalam *database* (D1) merujuk pada Gambar 3.5. Data-data terkait adalah *ticketID, userID, technicianID* (untuk memilih teknisi), *ticketDesc, ticketStatus, dan datetimeCreated*. Nantinya *Technician* akan menerima data tersebut dan menanggapi dengan *update* status tiket. Kemudian sistem akan *update* secara *real-time* dan menampilkan status tiket terbaru kepada user. User *Manager* dapat hanya dapat melihat seluruh tiket yang telah dibuat dan tidak dapat merubah atau menghapus tiket.

3. Proses 3 Akses *Admin*

   Setelah *admin* login, dapat melihat seluruh tiket yang ada, baik yang aktif maupun tutup. Data dari database akun juga dapat di akses oleh *admin* untuk mengubah, membuat, dan menghapus akun seluruh pengguna.

3.3.5 *Framework* Pemogramman yang digunakan

Aplikasi ini di rancang dengan menggunakan *JavaScript* sebagai bahasa pemrograman utama untuk pengembangan dalam bentuk website. *JavaScript* adalah bahasa pemrograman yang digunakan untuk membuat halaman web lebih interaktif dan dinamis. Bahasa ini berjalan di sisi klien, yang memungkinkan pengembang untuk menambahkan fungsionalitas seperti validasi formulir, manipulasi elemen DOM, animasi, serta komunikasi asinkron menggunakan API seperti AJAX [3]. Bahasa pemrograman ini juga memudahkan teknisi yang menjadi *admin* aplikasi karena juga mempunyai pengalaman yang cukup untuk bahasa pemrograman ini.

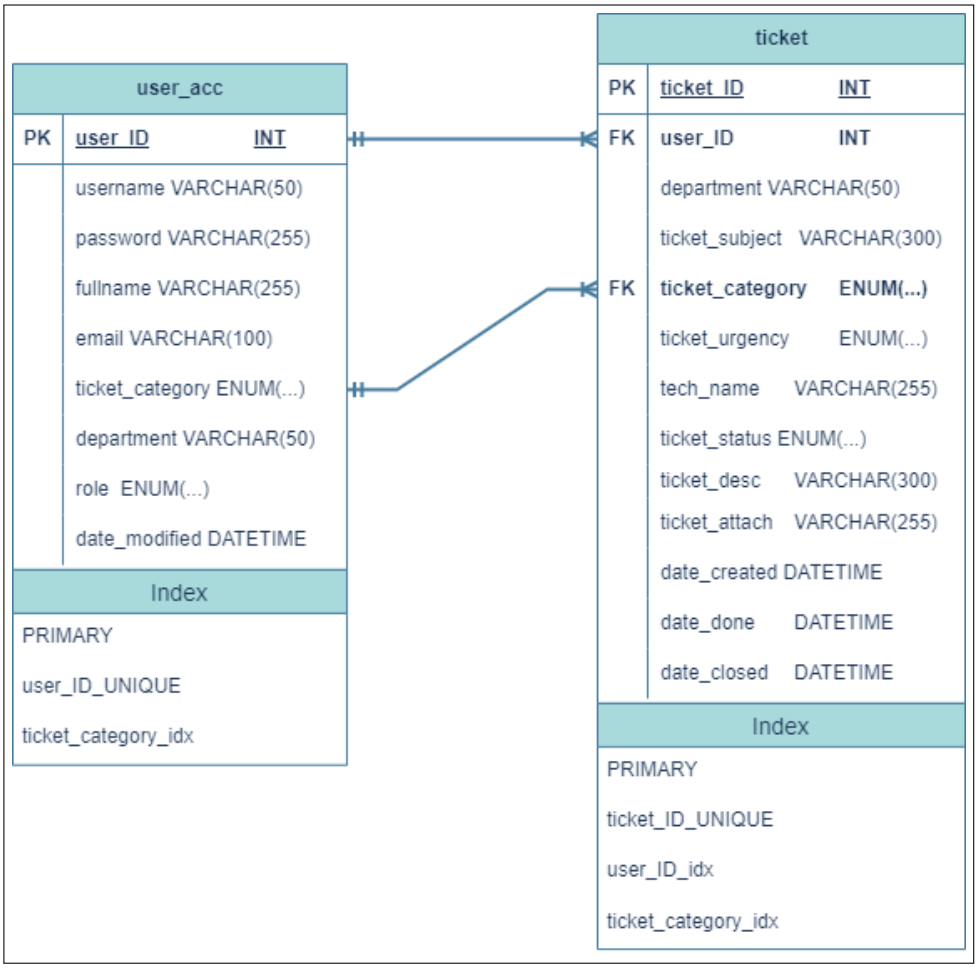
Sistem backend dijalankan di server menggunakan *Express.js*, sebuah *framework* untuk *Node.js* yang memudahkan pembuatan aplikasi web dan *API*. *Express.js* mendukung manajemen rute dan middleware secara efisien,

memungkinkan pengelolaan berbagai permintaan HTTP dan respons secara terstruktur [4]. Seperti halnya *Node.js*, *Express.js* juga memanfaatkan *non-blocking I/O* untuk menangani banyak koneksi secara bersamaan, yang meningkatkan performa dan skalabilitas aplikasi. Selain itu, *Express.js* mempermudah implementasi *real-time communication*, yang cocok untuk aplikasi helpdesk yang memerlukan interaksi langsung antara klien dan teknisi.

Sistem *frontend* menggunakan *framework React.js* karena kemampuannya dalam membangun antarmuka pengguna (UI) yang dinamis, efisien, dan berbasis komponen. *React* menggunakan *Virtual DOM*, yang memungkinkan *rendering UI* lebih cepat dibandingkan manipulasi *DOM* secara langsung [5].

3.3.6 Perancangan Database MySQL

Aplikasi menggunakan MySQL sebagai sistem basis data dikarenakan keunggulannya dalam kecepatan, efisiensi, dan kemudahan penggunaan. Selain itu, MySQL juga memiliki kompatibilitas baik dengan *Express.js*, serta menyediakan fitur keamanan yang dapat diandalkan untuk melindungi data sensitif [6]. Data dari MySQL di panggil menggunakan *Axios* yang merupakan library API dari *Express.js*.



Gambar 3.6. *Entity Relational Database* Aplikasi Helpdesk

Berikut penjelasan *Entity Relational Database* pada Gambar 3.6.

A. Tabel User

Tabel ini menyimpan informasi akun pengguna yang mencakup user biasa, teknisi, dan admin. Kolom-kolom yang ada pada tabel ini dirancang untuk memenuhi kebutuhan autentikasi, identifikasi, dan penugasan teknisi berdasarkan kategori.

Struktur Tabel:

1. userID (Primary Key): Identitas unik untuk setiap pengguna.
2. username (VARCHAR): Nama pengguna untuk *login* ke sistem.
3. password (VARCHAR): Kata sandi terenkripsi untuk keamanan akun.
4. fullname (VARCHAR): Nama lengkap pengguna.

5. email (VARCHAR): Alamat email pengguna.
6. category (ENUM): Kategori teknisi, seperti TI, HR, dan lainnya. Nilai ini diisi jika pengguna adalah teknisi. department (VARCHAR): Departemen tempat pengguna bekerja.
7. role (ENUM): Peran pengguna dalam sistem *(user, technician, admin)*.
8. datemodified (DATETIME): Tanggal terakhir akun dimodifikasi.

Indeks:

- Primary Key: userID.
- Indeks Unik: username idx untuk memastikan username tidak duplikat.
- Indeks: ticketcategory idx untuk mempercepat pencarian teknisi berdasarkan kategori.

B. Tabel ticket

Tabel ini berfungsi untuk menyimpan semua data terkait tiket bantuan atau laporan masalah yang dibuat oleh pengguna.

Struktur Tabel:

1. ticketID (Primary Key): Identitas unik untuk setiap tiket.
2. userID (Foreign Key): Menghubungkan tiket dengan pengguna yang membuatnya, merujuk ke userID di tabel useracc.
3. department (VARCHAR): Departemen terkait dengan tiket.
4. ticketSubject (VARCHAR): Subjek tiket untuk mendeskripsikan inti masalah.
5. ticketCategory (ENUM): Kategori tiket, seperti TI atau HR.
6. ticketUrgency (ENUM): Tingkat urgensi tiket (Low, Medium, High).
7. techName (VARCHAR): Nama teknisi yang menangani tiket.
8. ticketStatus (ENUM): Status tiket, seperti *"Open", "In Progress", "Done",* dan *"Closed."*
9. ticketDesc (VARCHAR): Deskripsi lengkap masalah atau permintaan.

10. ticketAttach (VARCHAR): *File* lampiran, jika ada, untuk mendukung deskripsi tiket.
11. dateCreated (DATETIME): Tanggal tiket dibuat.
12. dateDone (DATETIME): Tanggal tiket ditandai selesai oleh teknisi.
13. dateClosed (DATETIME): Tanggal tiket ditutup oleh user atau sistem.

Indeks:

- Primary Key: ticketID. Indeks:
- userIDIdx untuk mempercepat pencarian tiket berdasarkan pengguna.
- ticketcategoryIdx untuk pencarian tiket berdasarkan kategori.

C. Relasi Antar Tabel

1. Relasi useracc ke ticket:

   Relasi ini menggunakan *Foreign Key userID* pada tabel ticket yang merujuk ke userID di tabel useracc. Tipe Relasi: *One-to-Many*. Satu pengguna dapat membuat banyak tiket.

2. Relasi ticketCategory ke useracc.category:

   Relasi ini memastikan bahwa teknisi hanya dapat menangani tiket berdasarkan kategori yang sesuai dengan keahlian mereka. Misalnya, teknisi dengan kategori TI hanya akan menerima tiket dengan kategori TI.

D. Fungsi dan Kegunaan Database

1. Manajemen Akun Pengguna:

   Admin dapat membuat, mengedit, dan mengelola akun untuk user biasa dan teknisi. Setiap pengguna memiliki peran (*user, technician,* atau *admin* yang menentukan akses mereka dalam sistem.

2. Pembuatan dan Pelacakan Tiket:

   Pengguna dapat membuat tiket untuk melaporkan masalah atau meminta bantuan. Setiap tiket memiliki informasi lengkap, seperti subjek, deskripsi, kategori, urgensi, status, dan lampiran.

3. Penugasan Tiket ke Teknisi:

   Tiket dapat ditugaskan ke teknisi berdasarkan kategori masalah. Teknisi hanya dapat melihat tiket yang sesuai dengan keahlian mereka.

4. Pelacakan Status Tiket:

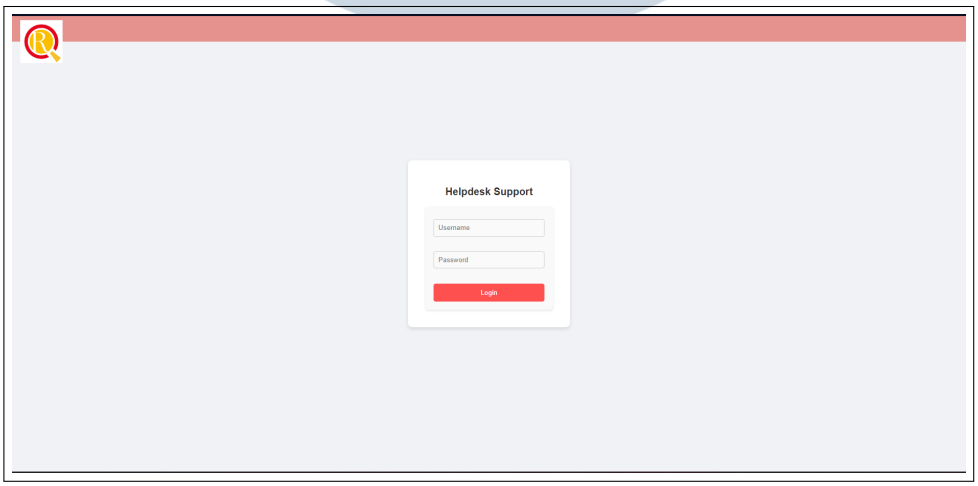
   Status tiket berubah secara bertahap dari *Open* ke *Done* atau *Closed*, memungkinkan pengguna dan *admin* melacak perkembangan penyelesaian masalah.

5. Sistem Log dan Riwayat:

   Tiket yang selesai atau ditutup tidak dihapus, tetapi disimpan sebagai *log* untuk referensi di masa mendatang.

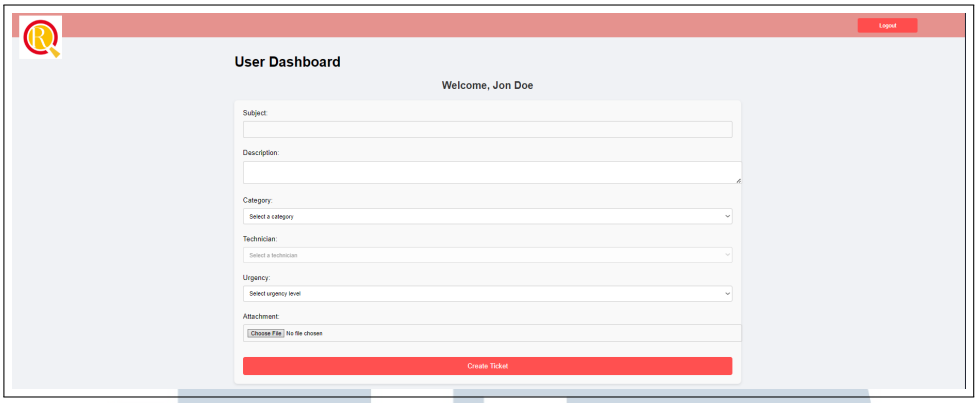
3.3.7 Perancangan Aplikasi Helpdesk PT. SRKI

A. Design Intermuka User Helpdesk PT. SRKI



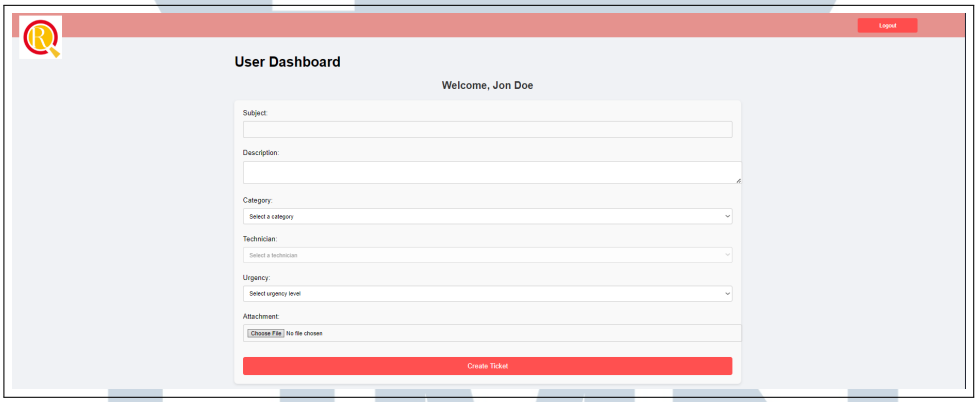
Gambar 3.7. *UI Login*

Pada Gambar 3.7, merupakan Intermuka *User* pada *login*. Semua pengguna akan masuk terlebih dahulu untuk mengetahui peran akun yang sesuai dan terdaftar pada akun *database*.



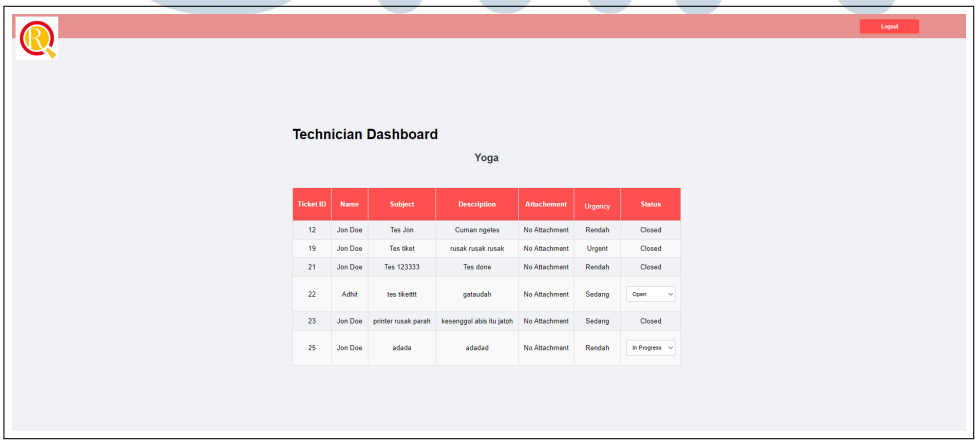
Gambar 3.8. *UI User Client* untuk Membuat Tiket

Jika sebuah user klien berhasil masuk pada Gambar 3.8 merupakan intermuka untuk membuat tiket. Terdapat tombol logout untuk keluar dan kembali pada halaman *login*.



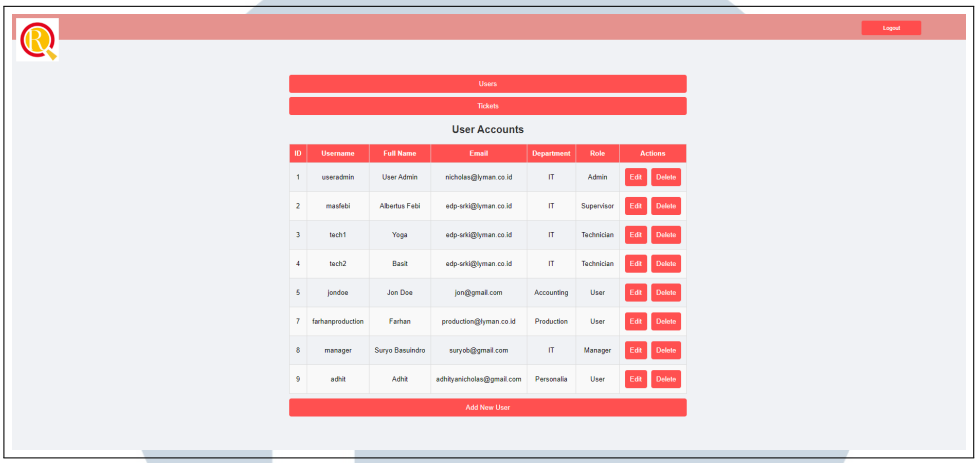
Gambar 3.9. *UI User Client* untuk Melihat Tiket

Berikut Gambar 3.9 merupakan intermuka klien untuk melihat dan melacak status tiket.



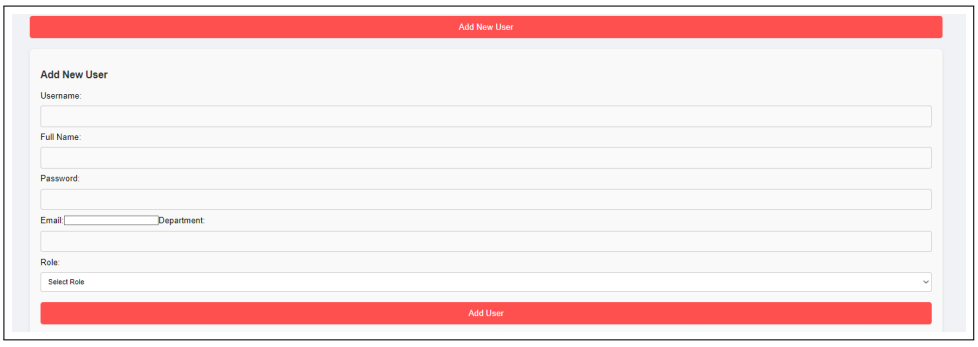
Gambar 3.10. *UI Tech* melihat Tiket

Berikut Gambar 3.10 merupakan intermuka untuk teknisi yang akan mendapatkan tiket yang telah dipilih dari user klien. Teknisi dapat merubah status dari ”Open”, ”In Progress”, dan ”Done”.



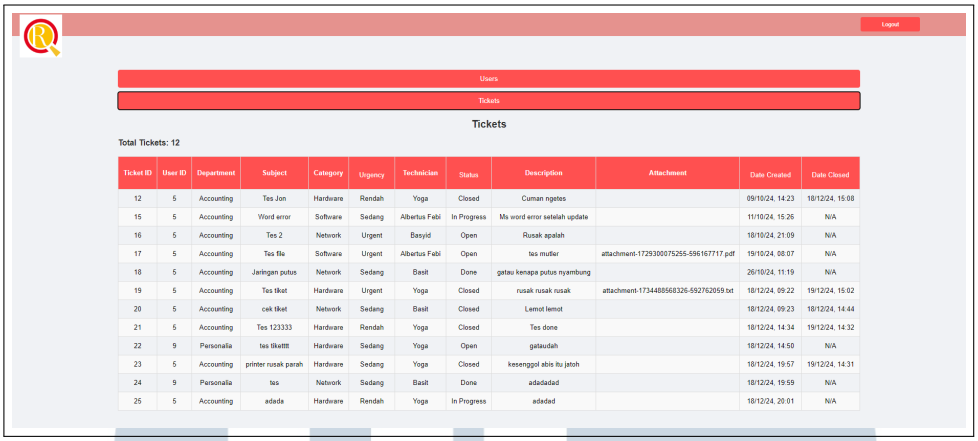
Gambar 3.11. *UI Admin* untuk Melihat User

Gambar 3.11 merupakan intermuka pada user Admin untuk melihat seluruh user yang ada. Admin dapat merubah, menghapus dan menambah akun (*CRUD*). *Admin* dapat menambahkan akun dengan meng-klik tombol *”Add New User”* yang akan menampilkan tampilan form untuk penambahan akun baru.



Gambar 3.12. *UI Admin* untuk Menambah Akun Baru

Setelah *Admin* meng-klik tombol untuk menambahkan akun, akan muncul tampilan form di mana *Admin* dapat memasukkan nama, kata sandi, email, departemen, serta peran (*role* yang digunakan untuk membedakan tampilan antarmuka berdasarkan fungsi masing-masing peran.



Gambar 3.13. *UI Admin* untuk Melihat Tiket

Gambar 3.13 merupakan intermuka pada user *Admin* untuk melihat seluruh tiket yang tersedia. Baik yang masih berstatus terbuka sampai sudah tutup. *Admin* juga dapat melihat total tiket yang tersedia dalam sistem basis data. Ini berguna sebagai referensi *manager* untuk melakukan laporan Key Performance Index.

B. Implementasi ke Server Client

Aplikasi helpdesk yang dikembangkan menggunakan arsitektur *port* terpisah, di mana *frontend React* berjalan pada *port* 3000 dan *backend* berjalan pada *port* 5000, mengadopsi prinsip pemisahan tanggung jawab antara *client-side* dan *server-side*. Komunikasi antara kedua port ini dilakukan menggunakan Axios, sebuah pustaka *HTTP client* yang memfasilitasi pengiriman permintaan API dari frontend ke backend.

Dalam skenario ini, CORS (*Cross-Origin Resource Sharing*) berperan penting untuk mengizinkan permintaan API lintas asal (cross-origin). *Cross-Origin Resource Sharing* (CORS) adalah mekanisme yang memungkinkan sumber daya di satu domain diakses oleh domain lain, yang secara default dibatasi oleh kebijakan Same-Origin Policy [7]. Middleware CORS di backend dikonfigurasi untuk mengizinkan akses dari frontend pada port 3000, sehingga permintaan API dapat dilakukan tanpa ditolak oleh browser.

3.4 Kendala dan Solusi yang Ditemukan

Adapun beberapa kendala dan solusi yang ditemukan selama pengembangan aplikasi helpdesk.

1. Pengembangan helpdesk dilakukan secara mandiri, sehingga memakan waktu lebih lama dan membuat beberapa fitur belum selesai dikembangkan. Solusinya adalah dengan memprioritaskan fitur inti sistem manajemen tiket agar sistem dapat berfungsi secara optimal sebelum mengembangkan fitur-fitur tambahan.
2. Keterbatasan waktu dan pengalaman juga menjadi kendala karena pengembangan dilakukan oleh intern yang masih belajar. Cara mengatasinya adalah dengan memanfaatkan sumber belajar tambahan, berdiskusi dengan *supervisor*, dan mengelola waktu secara lebih efektif.
3. Masalah teknis seperti integrasi *database* dan pembagian peran teknisi sering menjadi hambatan, namun dapat diatasi dengan debugging sistematis dan mencatat solusi untuk digunakan kembali jika masalah serupa muncul.
4. Masalah teknis dalam jaringan koneksi *server client* menjadi hambatan saat ini dikarenakan sistem *Cross Origin Request Sharing* yang mengambil data dari server lokal bukan server terbuka.
5. *JavaScript* kurang mendukung sistem operasi lama dan masih banyak sistem operasi pabrik menggunakan *Windows XP dan 7*. Ini akan berdampak pada *library javascript* soal pengelolaan data yang beberapa sudah memberhentikan fungsi pada sistem operasi lama.