

# Low Level Design Report

## Ticket Alerting System

Anishya Maurya      Anshah Hegde      Aryan Kumar      Kartik Mittal  
Maloth Vishal Nayak      Pramod Chaitanya Dandu      Raj Aryan

July 17, 2025

## Contents

<b>1</b>	<b>Overview and Component Diagram</b>	<b>3</b>
<b>2</b>	<b>ER Diagram</b>	<b>4</b>
<b>3</b>	<b>Process Flow</b>	<b>5</b>
<b>4</b>	<b>Authentication Module</b>	<b>6</b>
4.1	Overview . . . . .	6
4.2	Login Sequence Diagram . . . . .	6
4.3	Login UI . . . . .	7
4.4	Register Sequence Diagram . . . . .	8
4.5	Register UI . . . . .	8
4.6	Logout Sequence Diagram . . . . .	10
4.7	API Endpoints . . . . .	11
4.8	Summary . . . . .	11
<b>5</b>	<b>Analyst Module</b>	<b>11</b>
5.1	Overview . . . . .	11
5.2	Analyst Sequence Diagram . . . . .	12
5.3	Features & Responsibilities . . . . .	12
5.4	Ticket Assignment Workflow . . . . .	13
5.5	Analyst Actions . . . . .	13
5.6	Analyst Availability & Logout . . . . .	14
5.7	Real-Time Updates . . . . .	14
5.8	API Endpoints . . . . .	14
5.9	Ticket Status Flow . . . . .	14
5.10	System Integration . . . . .	14
5.11	Notes . . . . .	14
5.12	Analyst UI . . . . .	15

<b>6 Admin and Superadmin</b>	<b>19</b>
6.1 Overview . . . . .	19
6.2 Features . . . . .	19
6.3 Responsibilities . . . . .	19
6.4 System Integration and Real-Time Updates . . . . .	19
6.5 Admin Sequence Diagram . . . . .	20
6.6 Admin Workflow . . . . .	21
6.7 Admin Actions . . . . .	21
6.8 API Endpoints . . . . .	22
6.9 Superadmin . . . . .	22
6.10 Admin UI . . . . .	22
<b>7 Maker Controller (Manual Entry)</b>	<b>28</b>
7.1 Overview . . . . .	28
7.2 Key Responsibilities . . . . .	28
7.3 Manual Entry Sequence Diagram . . . . .	29
7.4 Workflow . . . . .	29
7.5 API Endpoints . . . . .	30
7.6 Example Request . . . . .	30
7.7 Summary . . . . .	30
7.8 Manual Entry UI . . . . .	31
<b>8 BullMQ Queue System</b>	<b>31</b>
<b>9 File Upload</b>	<b>32</b>
9.1 Overview . . . . .	32
9.2 API Endpoints . . . . .	32
9.3 User Flow . . . . .	33
9.4 File Upload Sequence Diagram . . . . .	33
9.5 Supported File Types . . . . .	33
9.6 Required Columns . . . . .	33
9.7 Backend Processing . . . . .	34
9.8 Error Handling . . . . .	34
9.9 Security & Permissions . . . . .	34
9.10 Benefits . . . . .	34
9.11 Example Status Messages . . . . .	35
9.12 Extensibility . . . . .	35
9.13 File Upload UI . . . . .	35
9.14 Running the Ticket Alerting System . . . . .	36
9.15 Sample Credentials . . . . .	37
9.16 Sample Credentials . . . . .	37
<b>10 Conclusion</b>	<b>38</b>

# 1 Overview and Component Diagram

This document provides the Low-Level Design (LLD) for the Ticket Alerting System. It covers detailed modules like Authentication, Analyst operations, Admin dashboards, Superadmin permissions, and background queues using BullMQ. The report includes ER diagrams, component diagrams, Sequence Diagrams, UI previews, and API endpoint details.

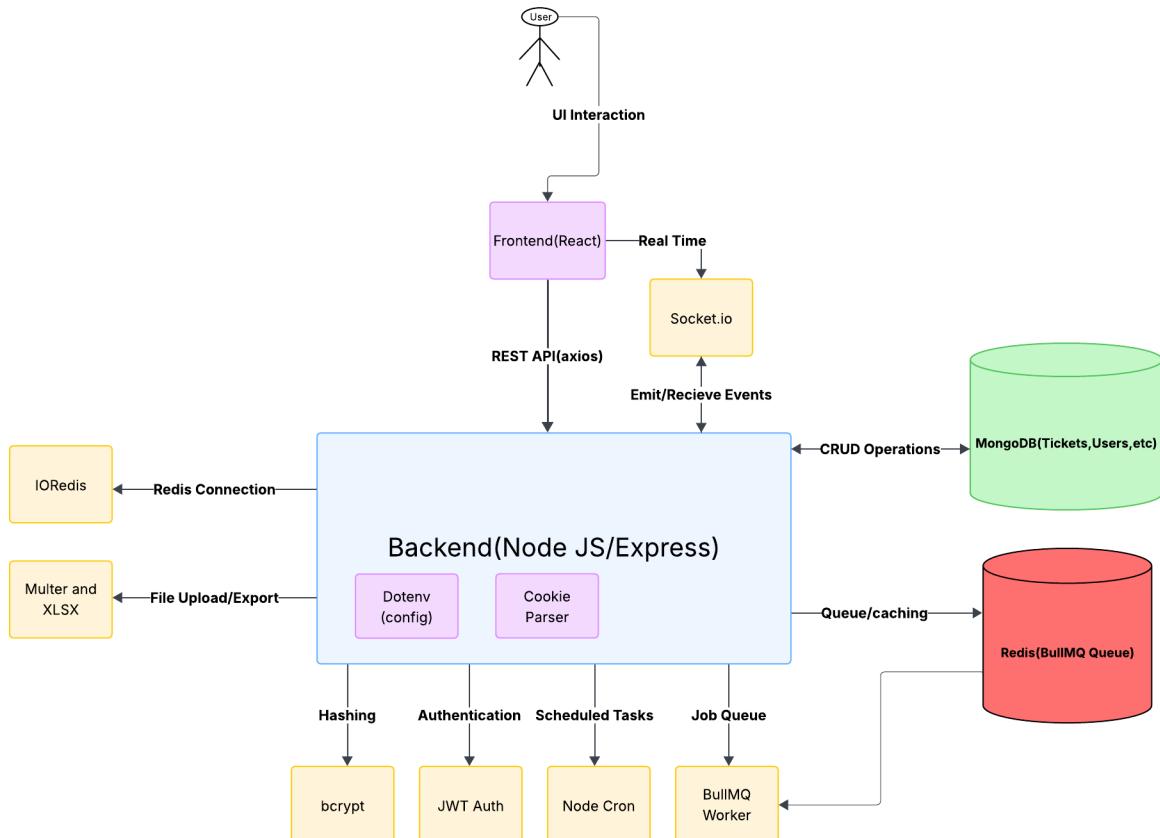
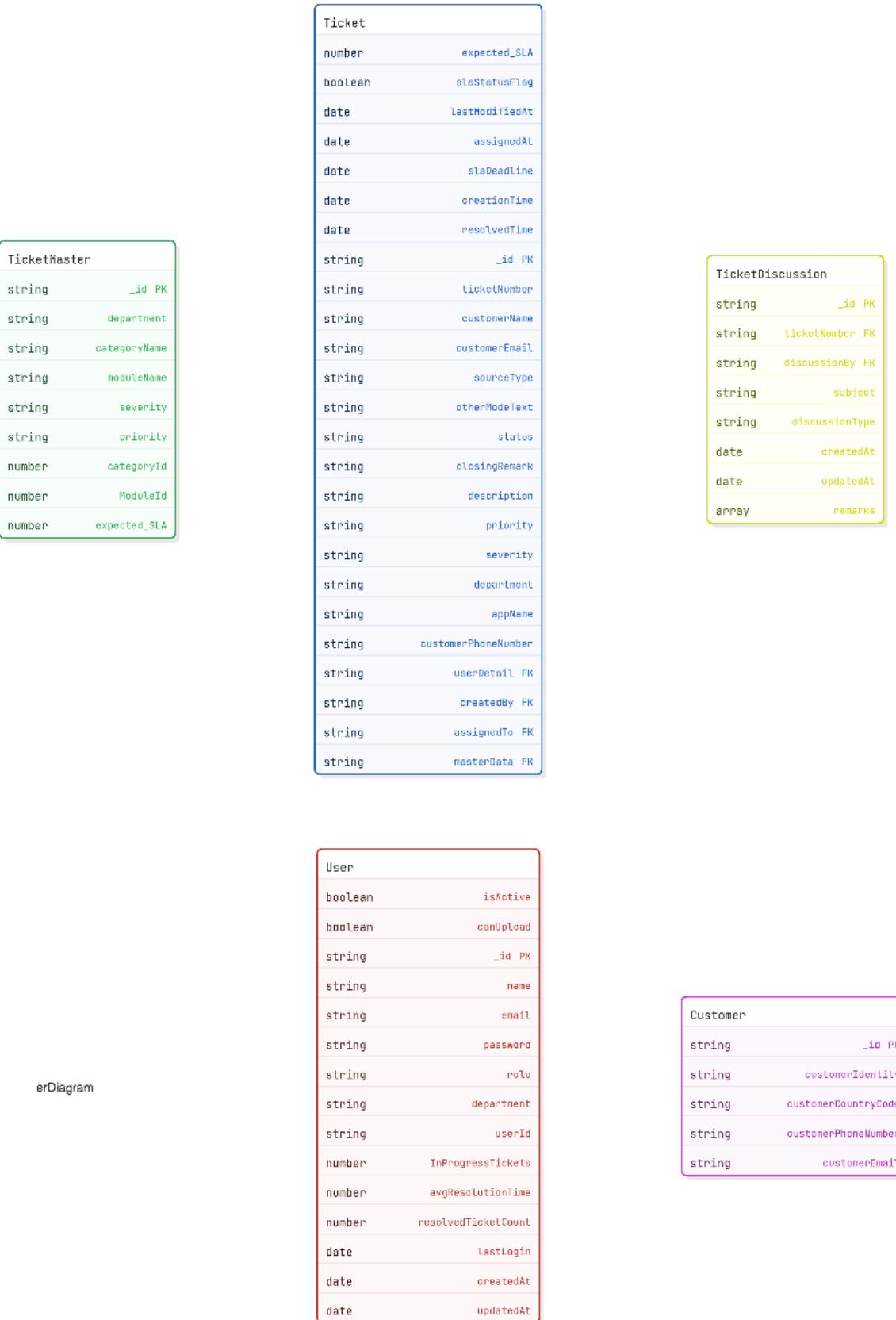


Figure 1: Component Diagram

A component diagram visually represents the structure of a system by showing its major components, their relationships, and interactions. It explains how modules are organized and communicate within the architecture.

## 2 ER Diagram



An ER (Entity-Relationship) diagram illustrates the entities in a system and the relationships between them, typically used for designing relational databases. In our project, we use MongoDB, which is a NoSQL, non-relational database, so the ER diagram conceptually guides our data modeling but does not enforce strict relational structures.

### 3 Process Flow

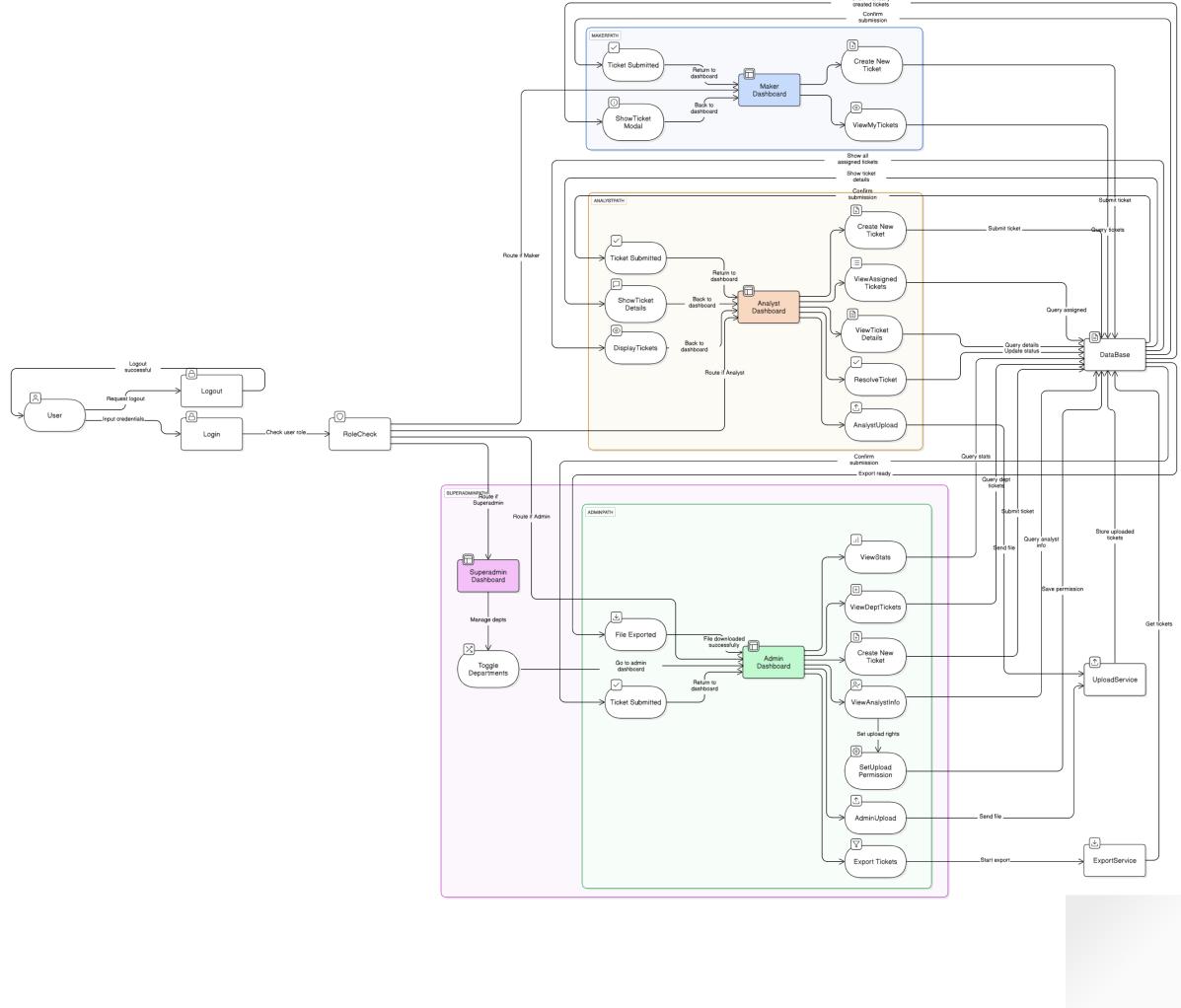


Figure 3: Process Flow diagram

A process flow diagram visually maps out the sequence of business activities, decisions, and paths within a system. It explains how our business processes operate from start to finish, showing the flow of tasks, user actions, and system responses to ensure clarity in how the overall workflow is managed and executed.

## 4 Authentication Module

### 4.1 Overview

The Authentication Controller manages user registration, login, and logout processes in the Ticket Alerting System. It ensures secure user management, session handling, and role-based logic for analysts and other users.

### 4.2 Login Sequence Diagram

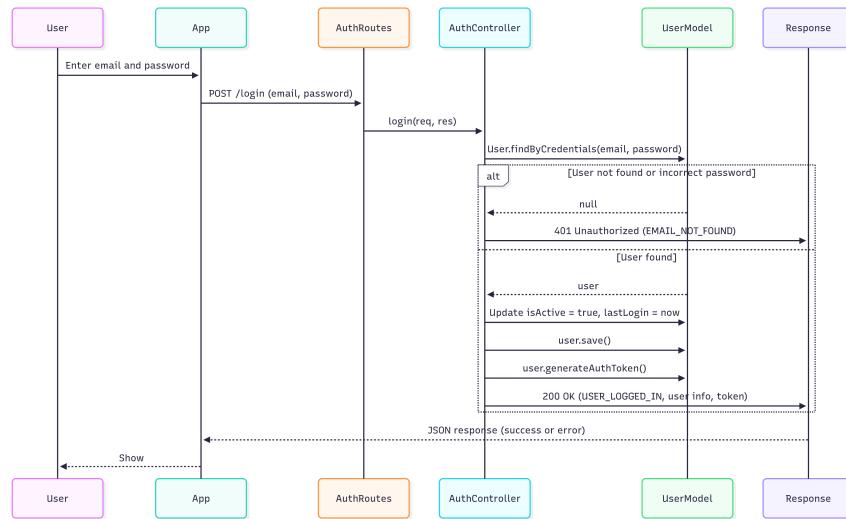


Figure 4: Login Sequence Diagram

A sequence diagram visually details the step-by-step interactions between system components or users over time. It explains how messages, requests, and responses flow in a specific scenario, clarifying the order and logic of operations.

### 4.3 Login UI

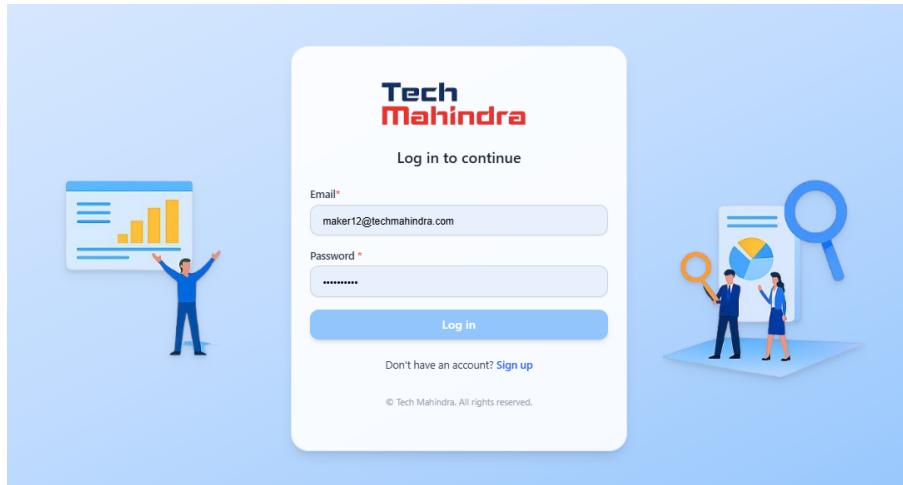


Figure 5: Login UI

**Endpoint:** POST /login

**How it works:**

- Receives email and password from the request body.
- Authenticates the user using the `findByCredentials` method.
- If authentication fails, responds with an unauthorized error.
- If successful:
  - Updates the user's `isActive` status and `lastLogin` timestamp.
  - Saves the updated user information.
  - Generates a JWT token and sets it as an HTTP-only cookie.
  - Responds with user details and the token.

**Key Points:**

- Secure session management using JWT cookies.
- Tracks user activity and last login time.
- Provides clear error messages for failed logins.

## 4.4 Register Sequence Diagram

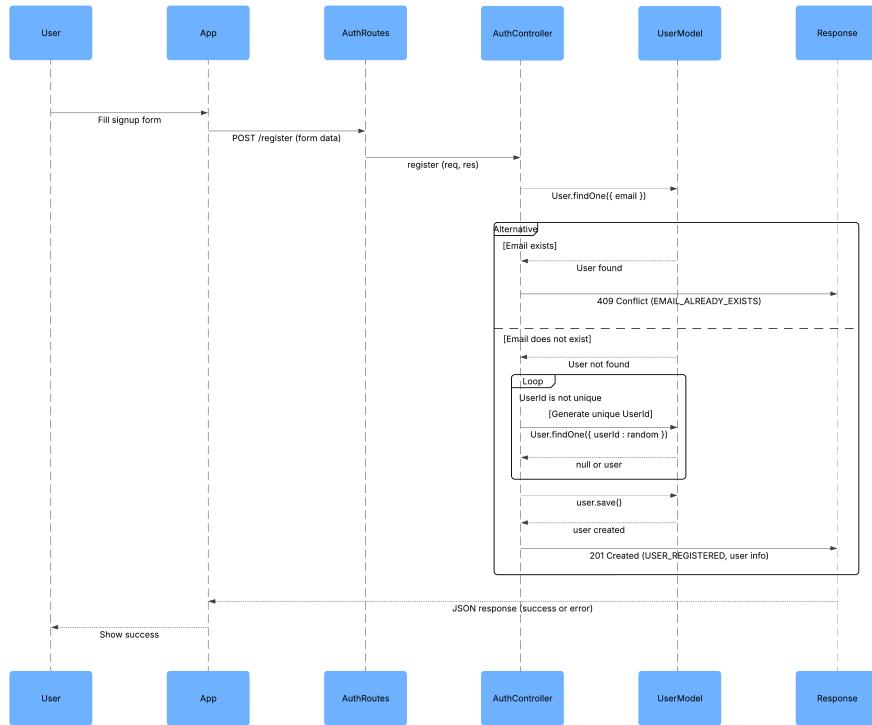


Figure 6: Register Sequence Diagram

## 4.5 Register UI

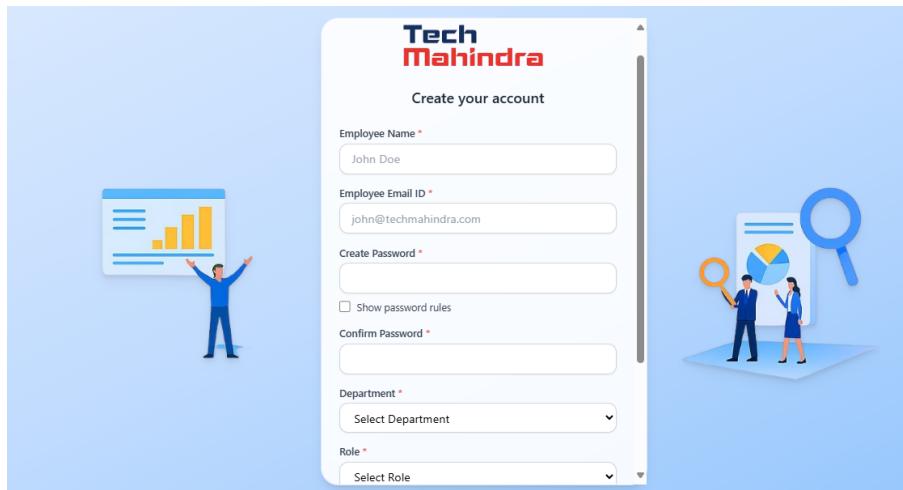


Figure 7: Register UI

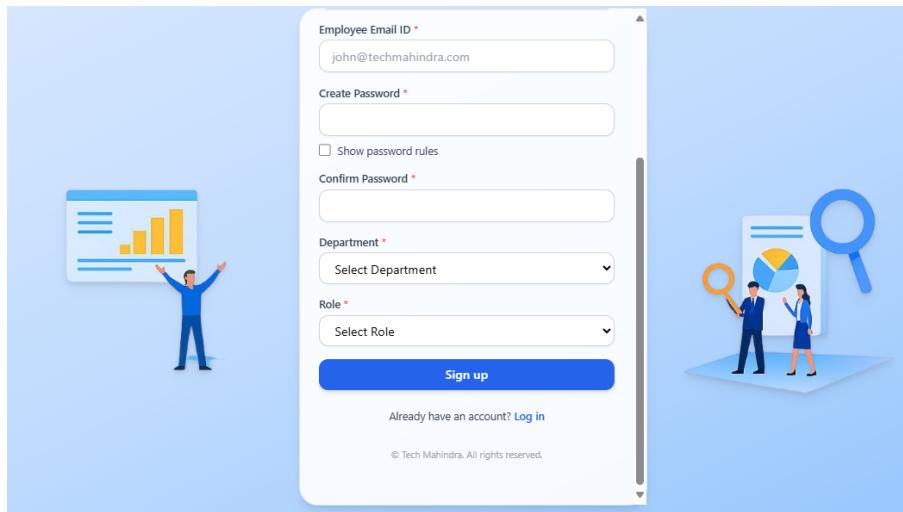


Figure 8: Register UI

**Endpoint:** POST /register

**How it works:**

- Receives user details (name, email, password, department, role) from the request body.
- Checks if a user with the provided email already exists.
- Generates a unique user ID.
- Creates and saves a new user in the database.
- Generates a JWT authentication token for the user.
- Responds with a success message, user details, and the token.

**Key Points:**

- Prevents duplicate registrations using email checks.
- Ensures each user has a unique identifier.
- Immediately authenticates the user upon successful registration.

## 4.6 Logout Sequence Diagram

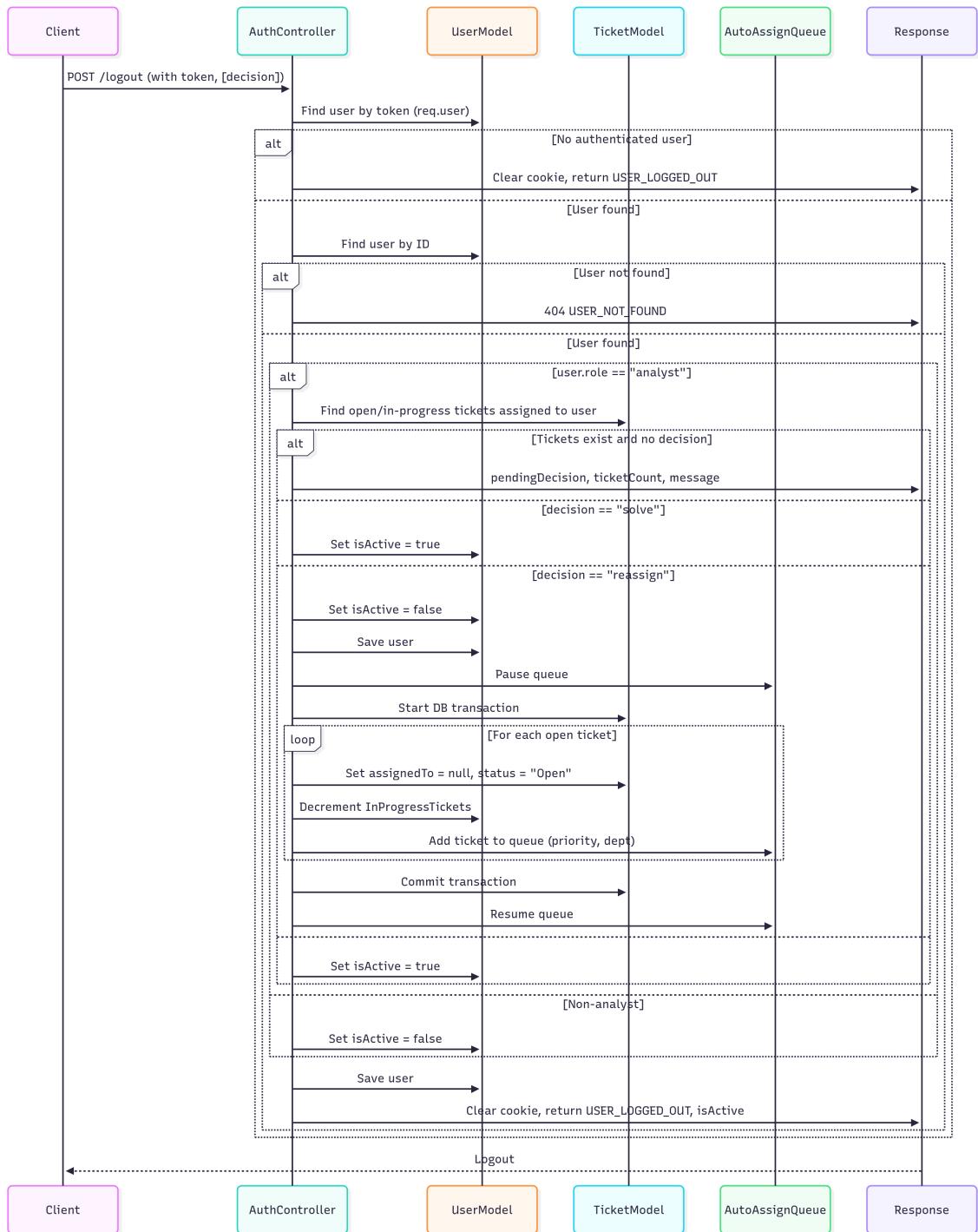


Figure 9: Logout Sequence Diagram

**Endpoint:** `POST /logout`

**How it works:**

- Checks if the user is authenticated.
- For analysts:

- Checks for any open or in-progress tickets assigned to the analyst.
- If tickets exist and no decision is provided, prompts the analyst to choose between “solve later” or “reassign”.
- If “reassign” is chosen:
  - \* Marks the analyst as inactive.
  - \* Reassigns all open tickets back to the auto-assign queue within a transaction.
  - \* Updates ticket and user records accordingly.
- If “solve later” is chosen, keeps the analyst active.
- For non-analyst users, simply marks the user as inactive.
- Clears the authentication cookie and responds with a logout confirmation.

#### **Key Points:**

- Special handling for analysts to ensure ticket continuity.
- Uses transactions for safe ticket reassignment.
- Ensures secure session termination for all users.

## **4.7 API Endpoints**

Endpoint	Description
POST /register	Registers a new user
POST /login	Authenticates and logs in a user
POST /logout	Logs out the user and handles ticket reassignment if analyst

## **4.8 Summary**

The Authentication Controller provides robust user management by:

- Validating and registering new users.
- Authenticating users and managing sessions securely.
- Handling analyst-specific logout scenarios to maintain ticket workflow integrity.
- Supporting role-based logic and secure cookie-based authentication.

## **5 Analyst Module**

### **5.1 Overview**

The Analyst is a core user role in the Ticket Alerting System, responsible for handling, resolving, and tracking support tickets. Analysts interact with the system through ticket assignment, resolution, and real-time updates, ensuring efficient ticket management and SLA compliance.

## 5.2 Analyst Sequence Diagram

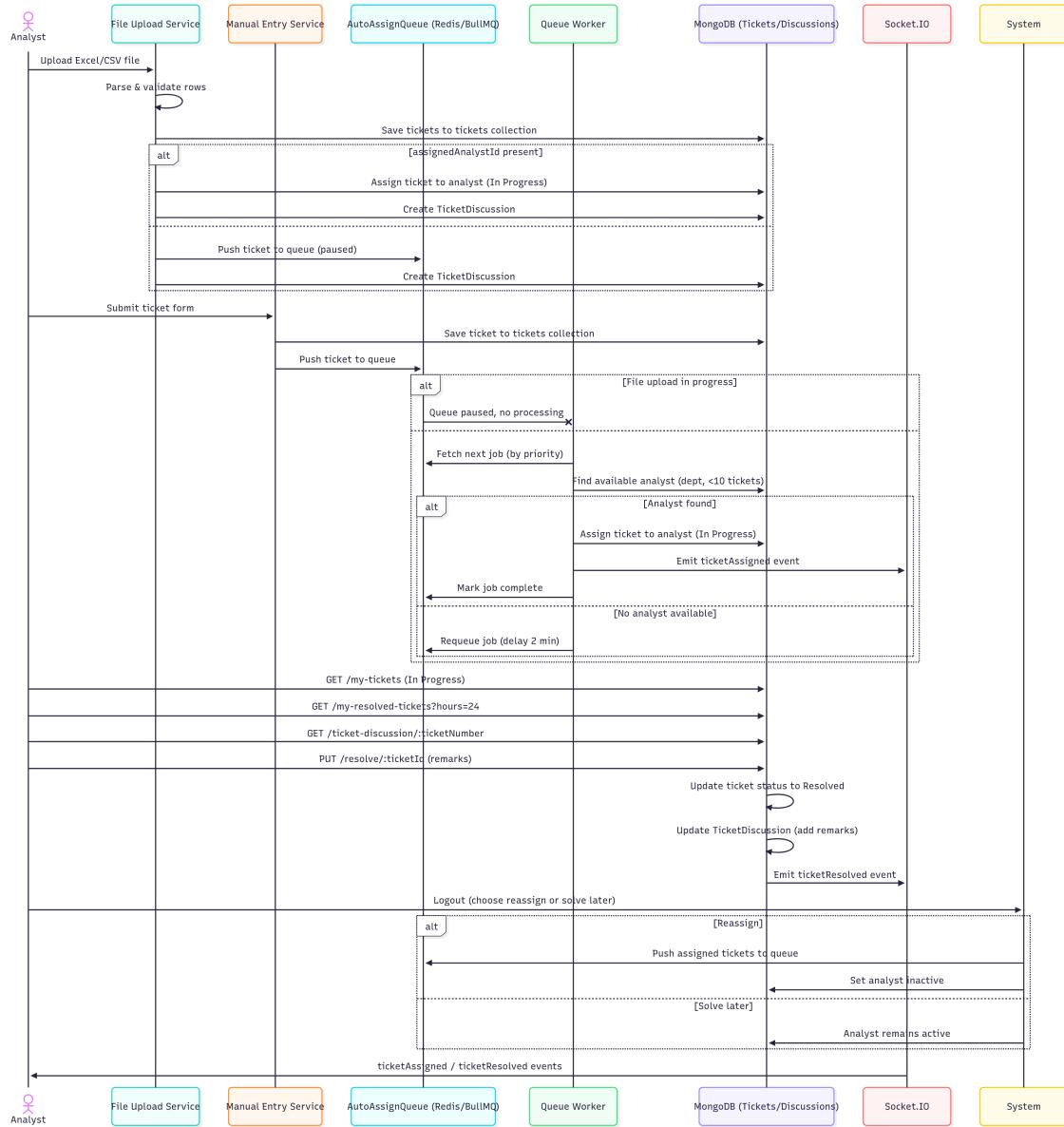


Figure 10: Analyst Workflow Sequence Diagram

## 5.3 Features & Responsibilities

- Receive tickets automatically assigned based on department, workload, and ticket priority.
- Resolve tickets and provide resolution remarks.
- View assigned and resolved tickets with filtering and sorting.
- Participate in ticket discussions for collaborative resolution.
- Real-time updates for ticket assignment and resolution.
- Workload balancing when analysts join or leave.

## 5.4 Ticket Assignment Workflow

### 1. Ticket Creation

- **File Upload:** Tickets are uploaded via `/api/upload` using Excel/CSV files. Each row is validated and saved to the MongoDB `tickets` collection. If `assignedAnalystId` is present, the ticket is directly assigned to that analyst; otherwise, it is pushed to the AutoAssignQueue. During file upload, the queue is paused to prevent premature assignment.
- **Manual Entry:** Tickets can be created via manual entry (`/manualentry` route). After validation, tickets are saved to the database and pushed to the AutoAssignQueue.

### 2. AutoAssignQueue & Assignment Logic

- **Queue Priority:** Tickets are prioritized as P1 (highest), P2, then P3. If no priority is provided, P3 is used by default.
- **Queue Worker:** Processes jobs when the queue is not paused. Assigns tickets to analysts based on:
  - Same department as the ticket.
  - Fewer than 10 active tickets.
  - Least urgent workload (based on priority and SLA).
- **Requeueing:** If no analyst is available or a temporary error occurs, the job is requeued after a delay. Permanent data errors cause the job to fail.

### 3. Workload Balancing

- If an analyst logs in and no tickets are available in the queue, but other analysts in the department have tickets, a rebalancing event is triggered. Tickets are reassigned from analysts with the highest workload to the new analyst.

## 5.5 Analyst Actions

- **View Assigned Tickets:** GET `/my-tickets`  
Returns all tickets currently assigned to the analyst and in progress.
- **View Resolved Tickets:** GET `/my-resolved-tickets?hours=<t>`  
Returns tickets resolved by the analyst in the last t hours.
- **View Ticket Discussions:** GET `/ticket-discussions/:ticketNumber`  
Shows the conversation and remarks history for a specific ticket.
- **Resolve Ticket:** PUT `/resolve/:ticketId`  
Marks the ticket as resolved, updates the ticket status, and adds remarks to the discussion.

## 5.6 Analyst Availability & Logout

When logging out, analysts are prompted to either:

- **Reassign tickets:** All assigned tickets are returned to the queue, and the analyst becomes inactive.
- **Solve later:** Analyst remains active and retains their tickets.

If an analyst is inactive for 8 hours, their tickets are automatically reassigned.

## 5.7 Real-Time Updates

Socket.IO events notify analysts of ticket assignments (`ticketAssigned`) and resolutions (`ticketResolved`). Each analyst receives updates only for their assigned tickets.

## 5.8 API Endpoints

- GET `/my-tickets`: Get all tickets assigned to analyst.
- GET `/my-resolved-tickets?hours=t`: Tickets resolved in last t hours.
- PUT `/resolve/:ticketId`: Resolve a ticket with remarks.
- GET `/ticket-discussions/:ticketNumber`: Get discussion thread.

## 5.9 Ticket Status Flow

- **Open:** Ticket created, not yet assigned.
- **In Progress:** Ticket assigned to an analyst.
- **Resolved:** Ticket resolved by analyst.
- **Closed:** Ticket closed (not pushed to queue).

## 5.10 System Integration

- **AutoAssignQueue:** Handles ticket assignment logic and requeueing.
- **MongoDB:** Stores tickets and discussion history.
- **Socket.IO:** Provides real-time updates to analysts.
- **Change Streams:** Used for workload rebalancing when analysts join.

## 5.11 Notes

- Only open tickets are pushed to the queue.
- Each ticket creation also creates a corresponding entry in the `TicketDiscussion` collection.
- Analysts can have a maximum of 10 active tickets at a time (except for direct assignment via file upload).

## 5.12 Analyst UI

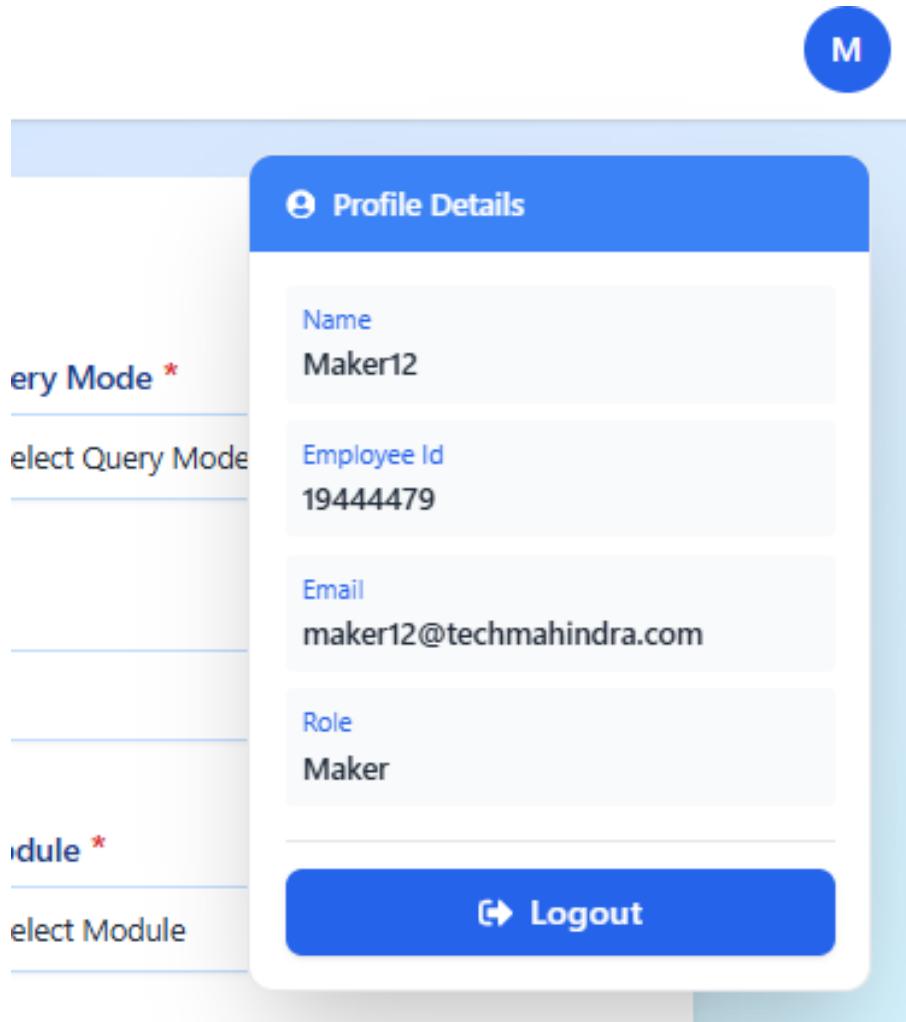


Figure 11: Analyst Profile UI

The screenshot shows the Analyst Dashboard. At the top, there is a header with the Tech Mahindra logo, a department selection dropdown (GLS), and a search bar. Below the header, the dashboard is divided into several sections:

- Analyst Dashboard**: Shows ticket counts: In Progress (0) and Resolved (0). It also includes a search bar and a priority dropdown.
- Quick Stats**: Displays counts for In Progress (0) and Resolved (0) tickets.
- Avg. Resolution Time**: Shows 0s based on 0 resolved tickets.
- SLA Overview**: Shows counts for Urgent (< 2h), Breached, and Good Standing.
- Priority Breakdown**: A chart showing the distribution of tickets by priority: High/P1 (red), Med/P2 (orange), and Low/P3 (blue).

The main content area displays a message: "No tickets assigned to you currently. You have no in-progress tickets."

Figure 12: Analyst Dashboard UI

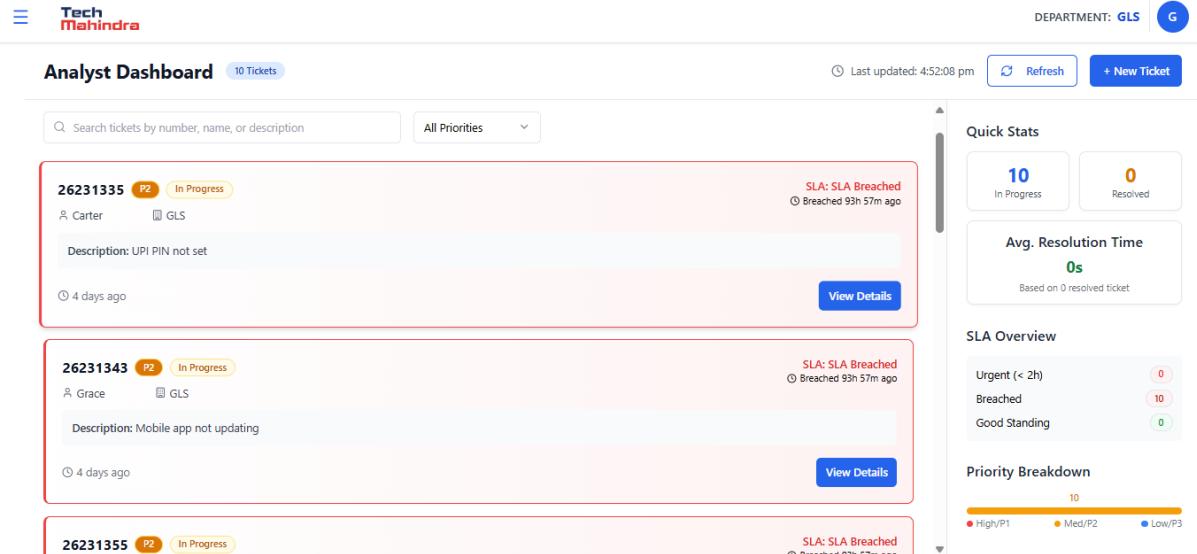


Figure 13: Analyst Dashboard UI

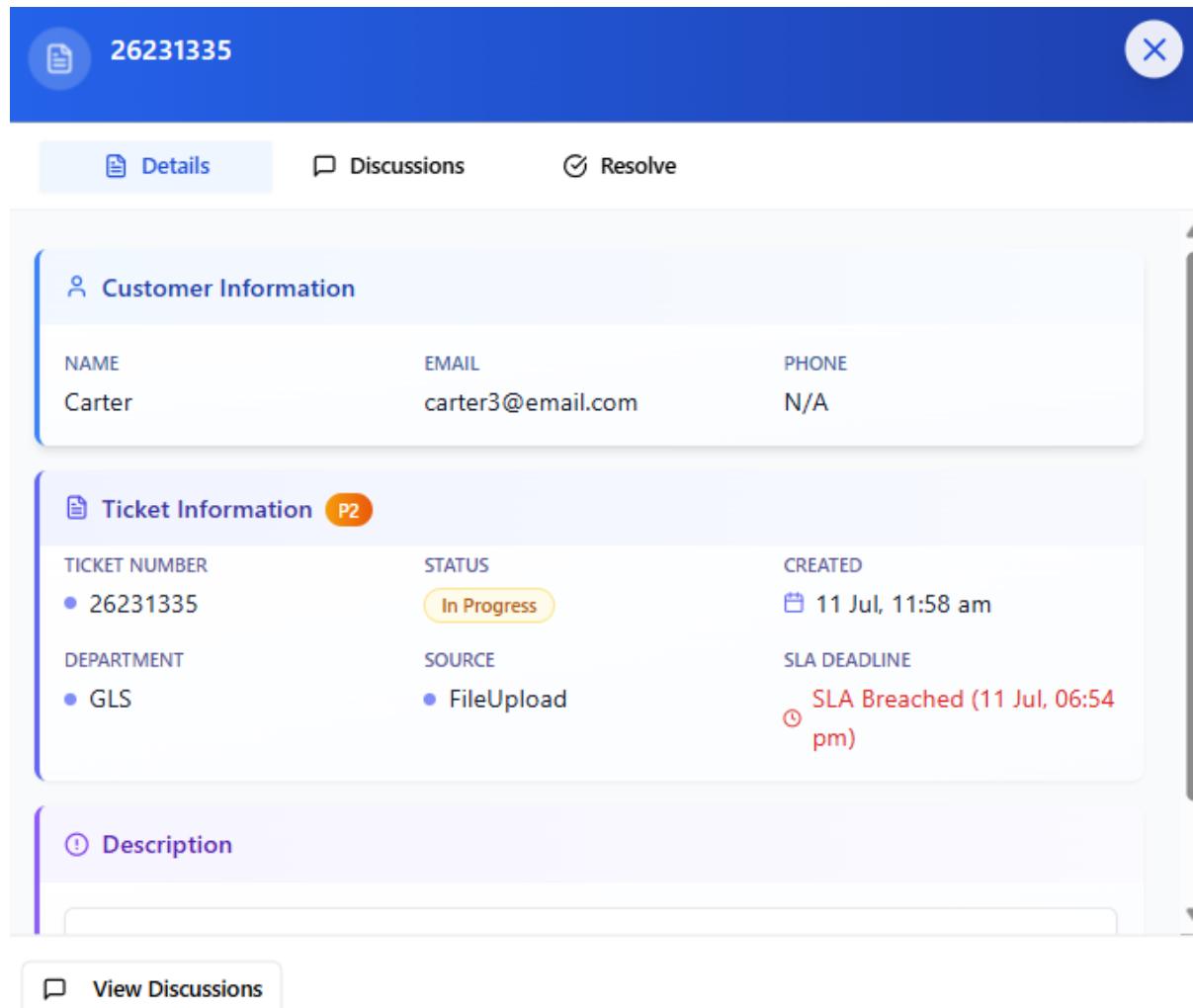


Figure 14: Analyst Dashboard UI

26231335

Details Discussions Resolve

**Resolve Ticket #26231335**

You are about to resolve the ticket for **Carter**. Please provide detailed information about how the issue was resolved. These notes will be sent to the customer.

Resolution Notes \* 27/300 characters

resolving your ticket buddy

These notes will be visible to the customer

ORIGINAL ISSUE  
UPI PIN not set

**Cancel** **Resolve Ticket**

Figure 15: Analyst Dashboard UI

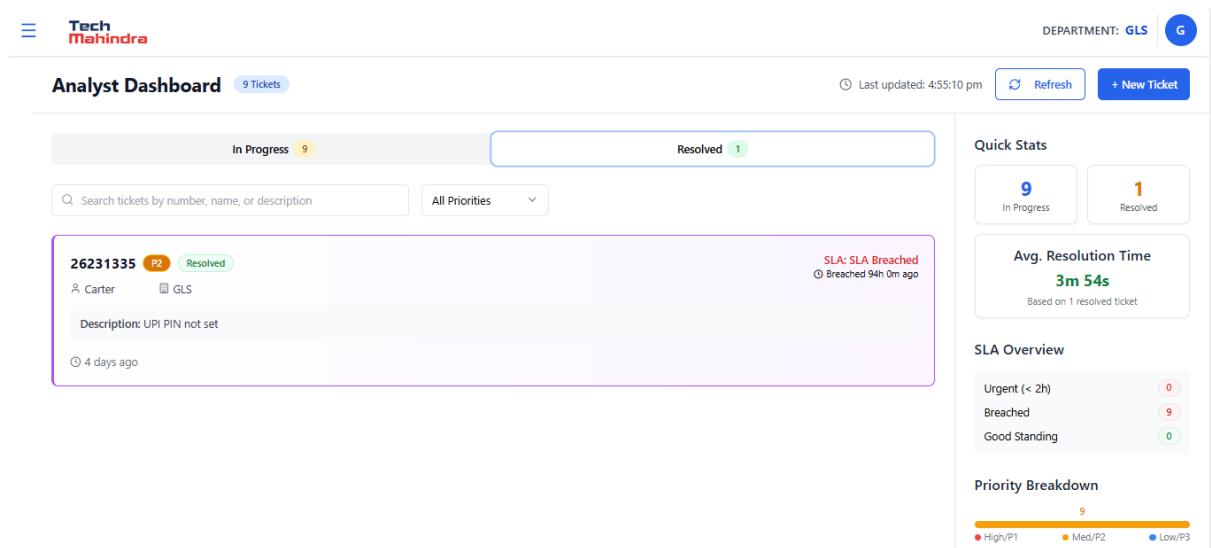


Figure 16: Analyst Dashboard UI

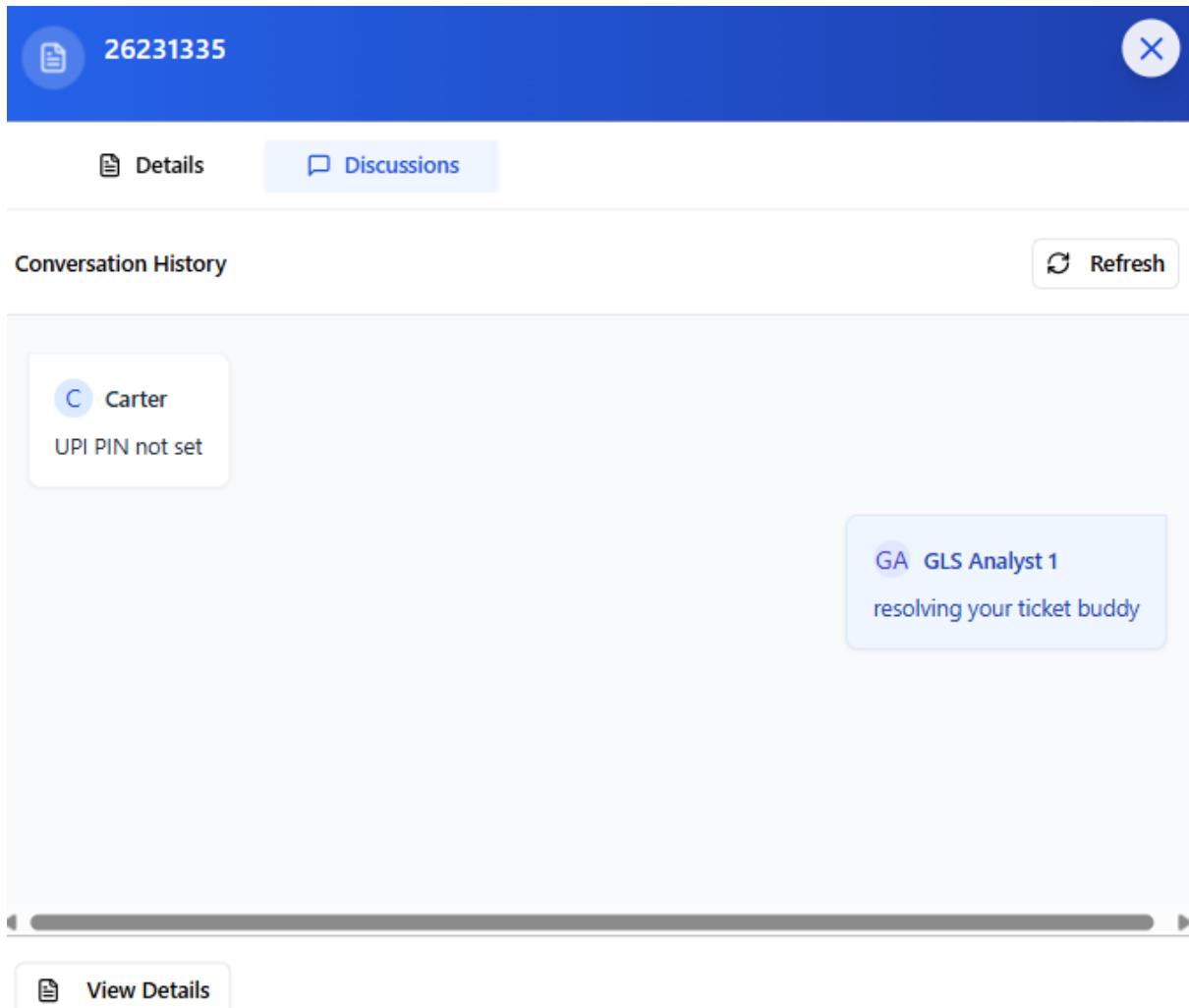


Figure 17: Analyst Dashboard UI

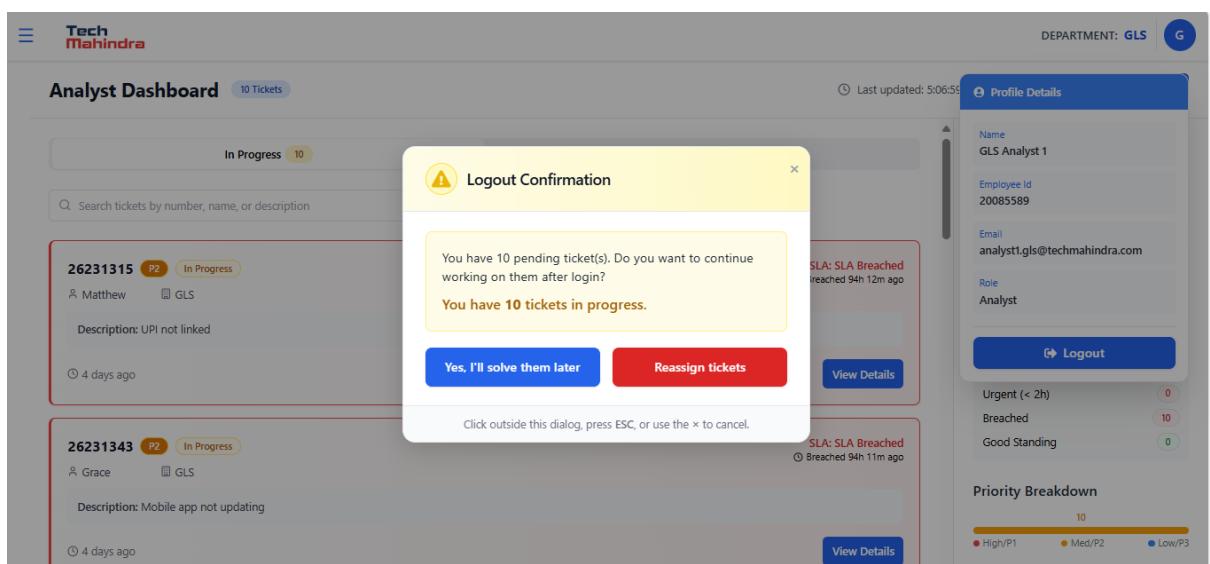


Figure 18: Analyst Dashboard UI

## 6 Admin and Superadmin

### 6.1 Overview

The Admin is responsible for managing ticket operations, overseeing analysts, and maintaining system data integrity. Admins have access to dashboards, ticket management, user permissions, and reporting tools.

### 6.2 Features

- Access to dashboard with ticket statistics and trends
- View, filter, and export tickets
- Upload ticket files and validate data
- Manage analyst information and permissions
- Monitor SLA compliance and ticket status
- Generate and download reports

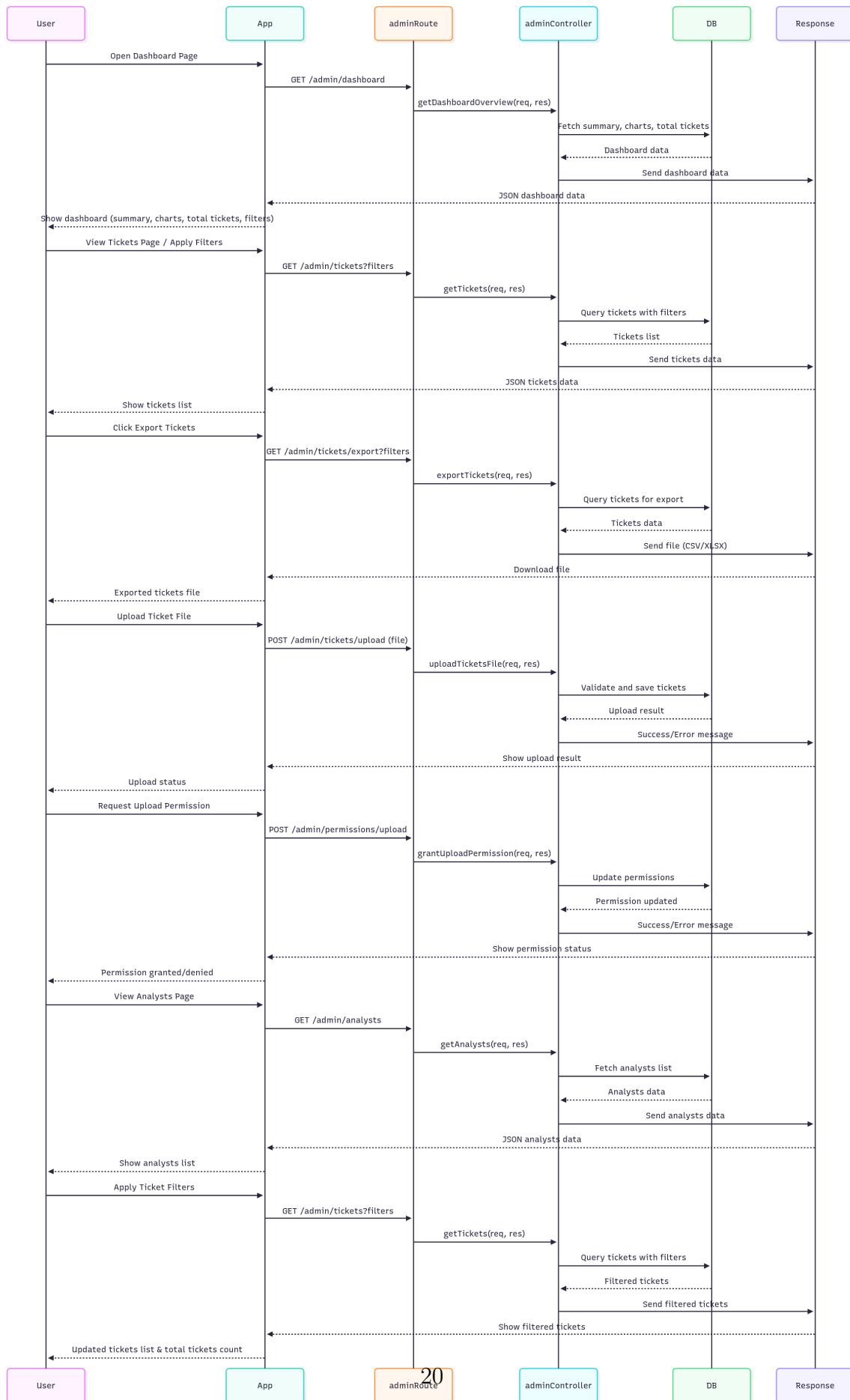
### 6.3 Responsibilities

- Oversee ticket assignment and resolution
- Ensure data accuracy through file uploads and validations
- Manage analyst roles and permissions
- Monitor system performance and SLA breaches
- Provide reports and insights to management

### 6.4 System Integration and Real-Time Updates

- **MongoDB:** Primary database for tickets, user data, and analytics, supporting fast queries and scalability.
- **xlsx Library:** Used for reading, validating, and processing Excel files during ticket uploads.
- **Socket.io:** Enables real-time communication for instant updates and notifications.
- **RESTful APIs:** Facilitate secure and efficient communication between frontend and backend.
- **Redis:** Used for caching and queue management.

## 6.5 Admin Sequence Diagram



## 6.6 Admin Workflow

The Admin workflow streamlines ticket management, analyst oversight, and system maintenance. Typical activities include:

1. **Login and Dashboard Review:** Upon logging in, the Admin is presented with a dashboard displaying key metrics such as ticket statistics, trends, SLA compliance, and recent activities.
2. **Ticket Management:** View all tickets, apply filters (status, priority, department), and sort results to monitor and prioritize resolution efforts.
3. **Exporting Ticket Data:** Export ticket data in CSV or Excel format for reporting and analysis.
4. **Uploading and Validating Ticket Files:** Upload bulk ticket files using the integrated xlsx library. The system validates data before processing to ensure accuracy.
5. **Managing Analysts and Permissions:** Access analyst profiles, update roles, and grant upload permissions to maintain data security and compliance.
6. **Monitoring SLA Compliance:** Receive real-time alerts for tickets approaching SLA deadlines and take proactive measures to minimize breaches.
7. **Generating Reports:** Generate and download reports on ticket summaries, analyst performance, and SLA compliance for informed decision-making.

## 6.7 Admin Actions

- GET /admin/dashboard: Retrieves dashboard summary and ticket statistics.
- GET /admin/tickets: View and filter tickets based on various criteria.
- GET /admin/tickets/export: Export ticket data as CSV or Excel.
- POST /admin/tickets/upload: Upload and validate ticket data files.
- GET /admin/tickets/source: Aggregated ticket data grouped by source.
- GET /admin/tickets/status: Ticket counts grouped by status.
- GET /admin/tickets/trends: Ticket trends over time for analysis.
- GET /admin/analysts: View details of all analysts.
- POST /admin/permissions/upload: Grant upload permissions to users.

## 6.8 API Endpoints

- GET /admin/dashboard: Dashboard metrics.
- POST /admin/tickets/upload: Upload tickets in bulk.
- GET /admin/tickets/export: Export tickets.
- GET /admin/analysts: View analyst details.
- POST /admin/permissions/upload: Grant upload access.

## 6.9 Superadmin

The Superadmin possesses all features available to the Admin. Additionally, the Superadmin can view and manage activities across multiple departments, providing broader oversight and control within the system.

## 6.10 Admin UI

The screenshot shows the 'Export Tickets' section of the Admin Dashboard. At the top, there is a blue header bar with the title 'Export Tickets' and the sub-instruction 'Filter and download ticket data'. Below the header, there is a 'Quick select' dropdown menu with three options: 'Last 7 days', 'Last 30 days', and 'Today'. Underneath this, there is a 'Filter Options' section. It includes a 'Department' section with five buttons: 'All Departments' (which is highlighted in blue), 'CMD', 'GLS', 'SCFU', and 'YONO'. There is also a 'Status' section with five buttons: 'All' (highlighted in blue), 'Open', 'InProgress', 'Closed', and 'Resolved'. Below these, there is an 'SLA Breached' section with a dropdown menu set to 'All'. At the bottom, there is a 'Date Range' section with two date input fields labeled 'Start Date' and 'End Date', each with a calendar icon.

Figure 20: Admin Dashboard UI

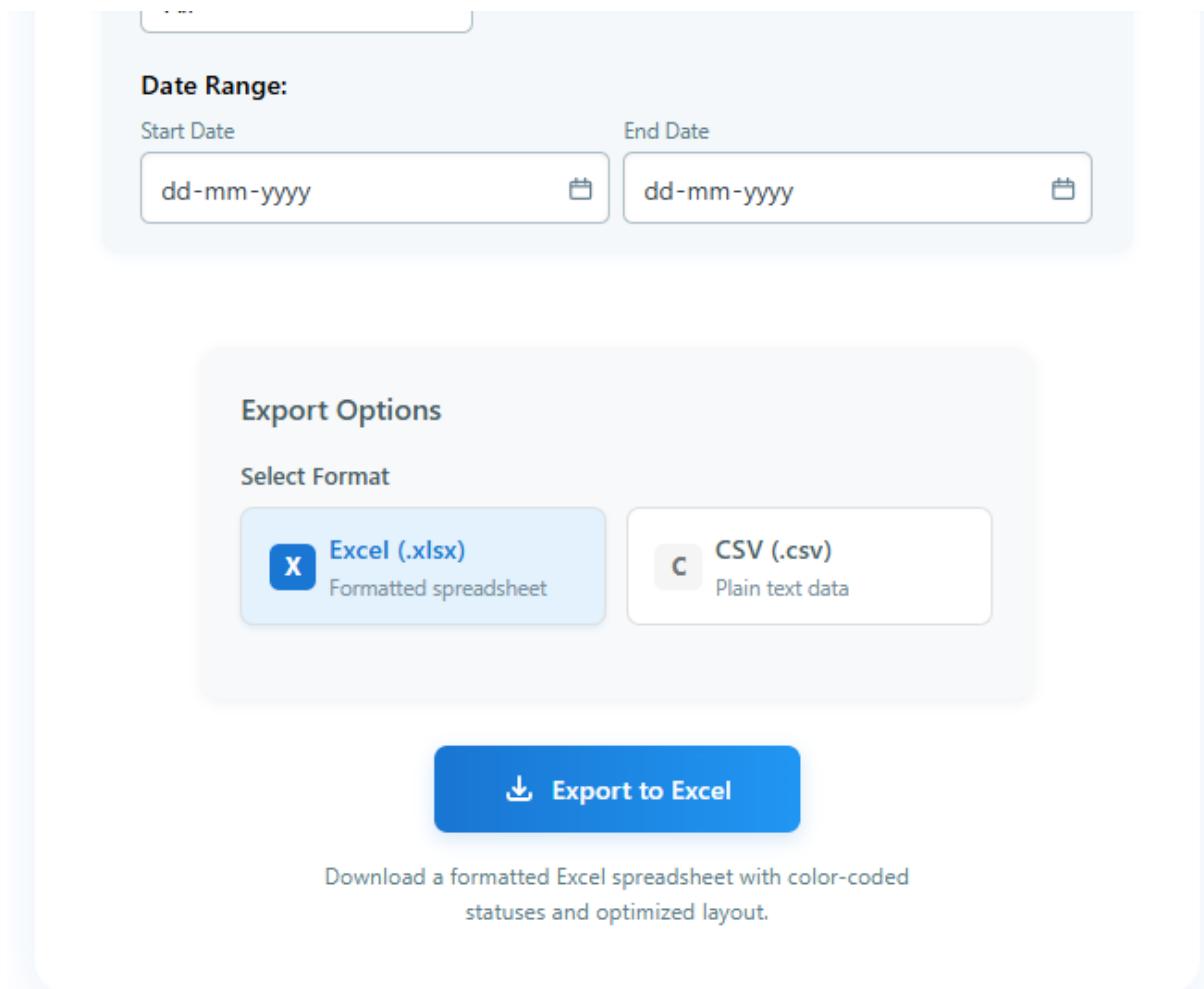


Figure 21: Admin Dashboard UI

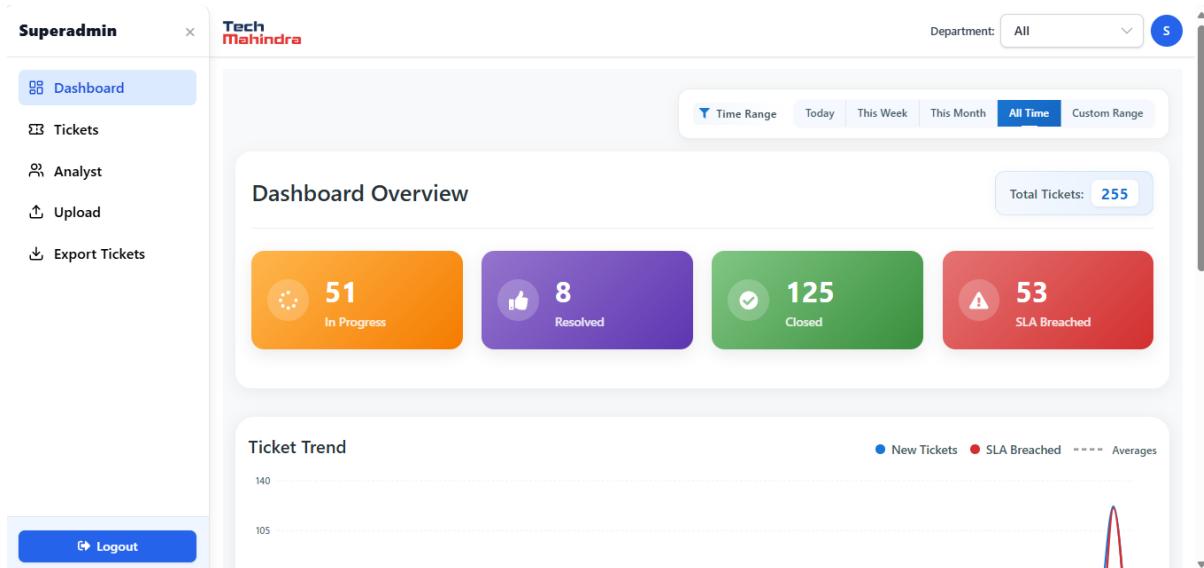


Figure 22: Admin Dashboard UI

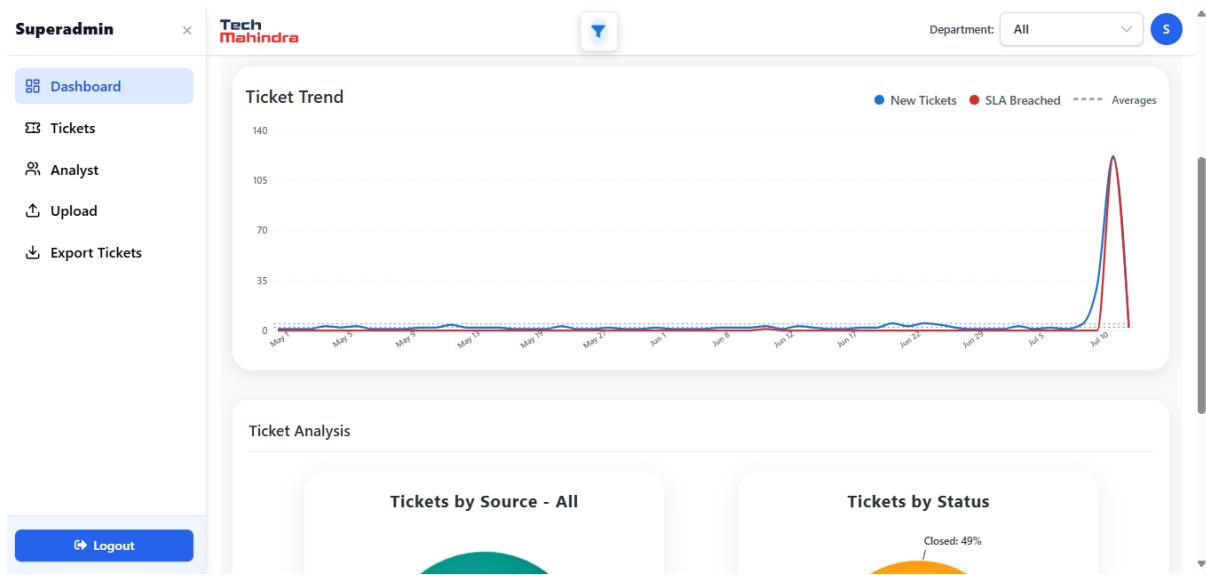


Figure 23: Admin Dashboard UI

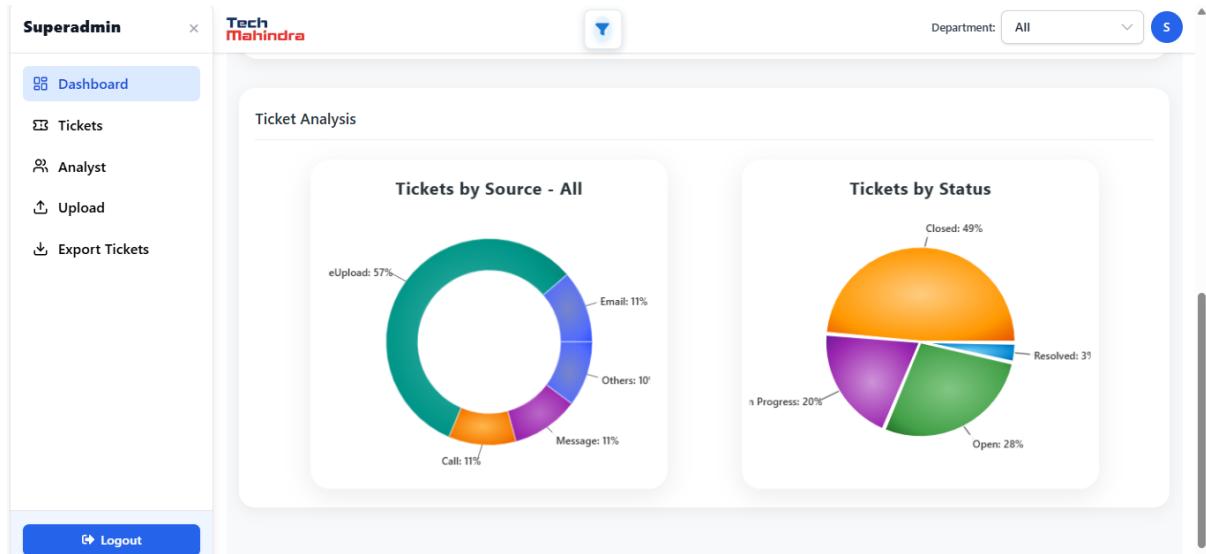


Figure 24: Admin Dashboard UI

The screenshot shows the Admin Dashboard for Tech Mahindra. On the left, a sidebar menu includes options like Dashboard, Tickets (selected), Analyst, Upload, and Export Tickets. The main area displays ticket statistics: New (0), Open (71), In Progress (51), Resolved (2), and Closed (0). A search bar at the top allows filtering by Ticket No. or Analyst. Below the stats, two ticket cards are shown:

- Ticket 26231265**: SLA Breached. Customer: Chloe, Department: SCFU, Subject: Unable to update KYC. Status: Open. SLA Deadline: 11 Jul 2025, 04:54 pm. Created at: 11 Jul 2025, 11:57 am.
- Ticket 26231287**: SLA Breached. Customer: Aurora, Department: GLS, Subject: Incorrect PAN details. Status: Open. SLA Deadline: 11 Jul 2025, 04:54 pm. Created at: 11 Jul 2025, 11:57 am.

At the bottom, a message indicates Showing 1-20 of 71 tickets, with a navigation bar for previous, next, and specific page numbers (1, 2, 3, 4, Go to: 1, of 4).

Figure 25: Admin Dashboard UI

The screenshot shows the Admin Dashboard for Tech Mahindra. The sidebar menu includes Dashboard, Tickets, Analyst (selected), Upload, and Export Tickets. The main area displays analyst statistics: Total Analysts: 21. A search bar and filter/sort button are available. Four analyst profiles are listed:

- CMD Analyst 3**: Joined Jun 26, 2025. Email: analyst3.cmd@techmahindra.com, Employee ID: 11969270, Department: CMD. Today's Ticket: Assigned: 12, In Progress: 11, Resolved: 1. View Details.
- GLS Analyst 1**: Joined Jun 26, 2025. Email: analyst1.gls@techmahindra.com, Employee ID: 20085589, Department: GLS. Today's Ticket: Assigned: 11, In Progress: 10, Resolved: 1. View Details.
- GLS Analyst 2**: Joined Jun 26, 2025. Email: analyst2.gls@techmahindra.com, Employee ID: 12905405, Department: GLS. Today's Ticket: Assigned: 0, In Progress: 0, Resolved: 0. View Details.
- GLS Analyst 3**: Joined Jun 26, 2025. Email: analyst3.gls@techmahindra.com, Employee ID: 11953703, Department: GLS. Today's Ticket: Assigned: 0, In Progress: 0, Resolved: 0. View Details.

At the bottom, a message indicates Showing 1-20 of 21 analysts, with a navigation bar for previous, next, and specific page numbers (1, 2, Go to: 1, of 2).

Figure 26: Admin Dashboard UI

## Analyst Profile



Personal Details

Ticket Statistics

Permissions

### Personal Information

Full Name

CMD Analyst 3

Email Address

analyst3.cmd@techmahindra.com

Employee ID

11969270

Department

CMD

### Account Information

Role

Analyst

Upload Permission

● Enabled

Account Created

Jun 26, 2025, 08:39 PM

Last Login

Jul 15, 2025, 12:19 PM

Figure 27: Admin Dashboard UI

## Analyst Profile



Personal Details

Ticket Statistics

Permissions

Time period:

Today

This Week

This Month

This Year

From

dd-mm-yyyy



to

dd-mm-yyyy



### Ticket Overview



**12**  
Assigned



**11**  
In Progress



**1**  
Resolved



**12**  
SLA Breached

### Performance Metrics

#### Resolution Rate

**8%**

Tickets resolved vs. total assigned

#### SLA Compliance

**0%**

Tickets within SLA vs. total tickets

Figure 28: Admin Dashboard UI

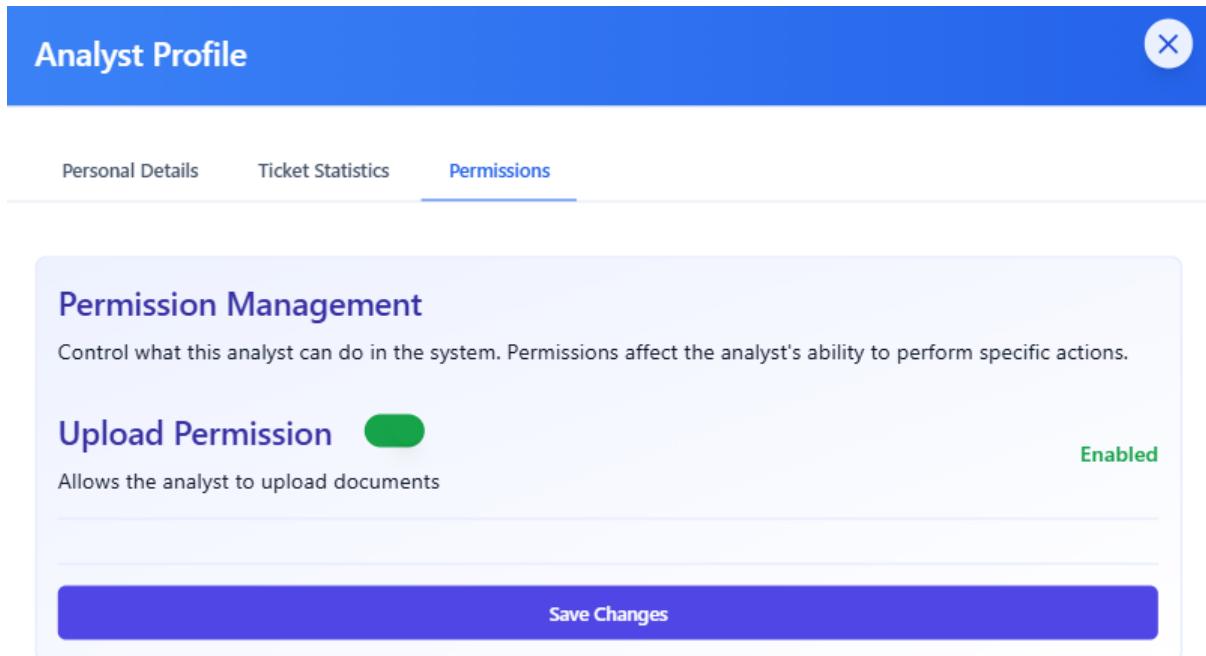


Figure 29: Admin Dashboard UI

## 7 Maker Controller (Manual Entry)

### 7.1 Overview

The Maker Controller manages the creation of new tickets via manual entry in the Ticket Alerting System. It handles validation, ticket creation, discussion initialization, and integration with the auto-assignment queue. This controller is invoked by the route `POST /manualentry` with the middlewares: `authMiddleware_cookie`, `maker_entry`.

### 7.2 Key Responsibilities

- Validates all required ticket fields from the request.
- Generates a unique ticket number based on department, phone, and timestamp.
- Fetches master data (category, module, SLA, priority, severity) for the ticket.
- Creates and saves the ticket in the MongoDB `tickets` collection.
- Initializes a `TicketDiscussion` entry for conversation history.
- Pushes unassigned tickets into the AutoAssignQueue for automated analyst assignment.
- Returns the created ticket as a JSON response.

### 7.3 Manual Entry Sequence Diagram

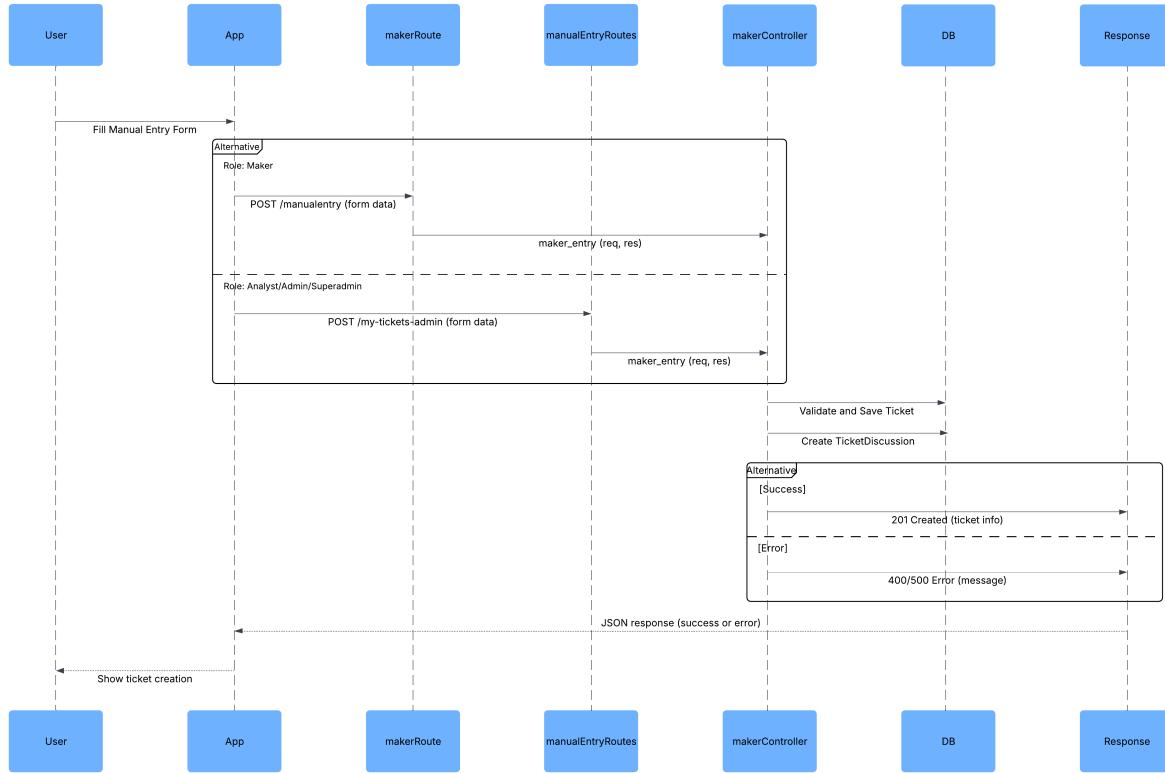


Figure 30: Manual Entry Sequence Diagram

### 7.4 Workflow

- Input Validation:** Checks for required fields: customer name, country code, phone number, mode, category, module, description, and department. Validates formats for country code, phone number, and email. Ensures the query mode is one of the allowed values (**Call**, **Email**, **Message**, **Others**).
- Ticket Number Generation:** Constructs a unique ticket number using:
  - Department code
  - Last 3 digits of phone number
  - Timestamp
  - Random 2-letter code
- Master Data Lookup:** Retrieves ticket master data (category, module) to determine priority, severity, and expected SLA. If master data is invalid, returns an error.
- Ticket Creation:** Combines country code and phone number. Sets ticket fields including status (**Open**), description, priority, severity, SLA, and department. Saves the ticket to the database.

5. **Ticket Discussion Initialization:** Creates a new `TicketDiscussion` entry with the ticket number and initial remarks (from the maker).
6. **Auto-Assignment Queue Integration:** If the ticket is not directly assigned (`assignedTo` is null):
  - Pauses the auto-assign queue.
  - Adds the ticket to the queue with priority ( $P1 > P2 > P3$ ).
  - Sets retry attempts and exponential backoff.
  - Resumes the queue.
7. **Response:** Returns a success response with the created ticket details. Handles and logs errors, returning a server error response if needed.

## 7.5 API Endpoints

<code>POST /manualentry</code>	Create a new ticket via manual form
--------------------------------	-------------------------------------

## 7.6 Example Request

```
POST /manualentry
{
  "customerName": "John Doe",
  "customerCountryCode": "+91",
  "customerPhoneNumber": "9876543210",
  "customerEmail": "john@example.com",
  "mode": "Email",
  "categoryName": "Technical",
  "moduleName": "Login",
  "description": "Cannot log in",
  "department": "IT"
}
```

## 7.7 Summary

The Maker Controller ensures that all manually entered tickets are validated, properly initialized, and either assigned directly or queued for auto-assignment. It maintains data integrity and integrates with the ticket discussion and assignment subsystems for seamless ticket lifecycle management.

## 7.8 Manual Entry UI

The screenshot shows the 'Tech Mahindra' manual entry interface. On the left, there's a sidebar with 'Maker12' at the top, followed by 'Manual Entry' and 'My Tickets'. A blue circular icon with a white 'M' is in the top right corner. The main area is titled 'Create New Ticket' and contains fields for Customer ID, Customer Name, Query Mode, Country Code, Customer Phone Number, Customer Email, Department, Category, Module, and a large Description field with a character limit of 0/300. A 'Logout' button is at the bottom left, and a 'Create Ticket' button is at the bottom right.

Figure 31: Manual Entry UI

The screenshot shows the 'Tech Mahindra' ticket dashboard. At the top, it says 'My Tickets' with '2 Total' and a close button. Below is a 'Filter & Search' section with date pickers for 'dd-mm-yyyy', 'Filter This Week', 'All Months', 'All Years', and an 'Apply Filters' button. Two ticket cards are listed: 'CMD89002491NF' (Date: 11/7/2025 01:26 pm) and 'GLS78919601OO' (Date: 11/7/2025 12:30 pm). Each card displays Customer Information (Name, Phone, Email, Source) and Ticket Details (Department, Category, Module, Description).

Figure 32: Manual Entry UI

## 8 BullMQ Queue System

BullMQ is used in our ticketing tool to manage background job queues, enabling efficient and reliable ticket assignment. It helps prioritize, schedule, and process ticket-related tasks asynchronously, ensuring tickets are assigned to analysts based on business rules without blocking the main application flow. This improves scalability, performance, and real-time responsiveness in handling ticket workloads.

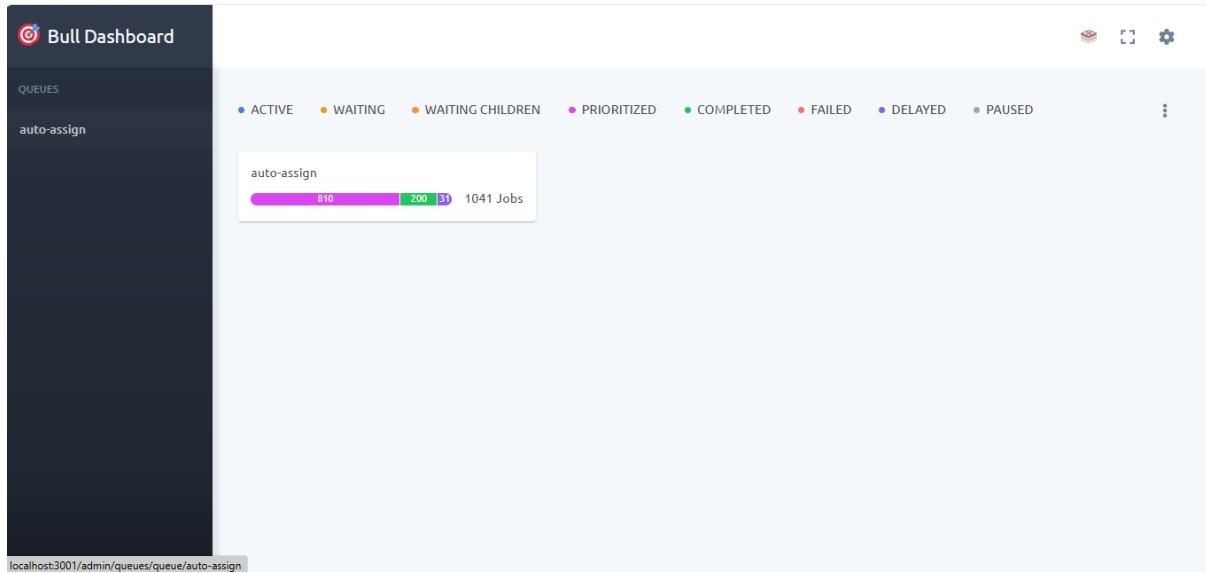


Figure 33: BullMQ Management UI

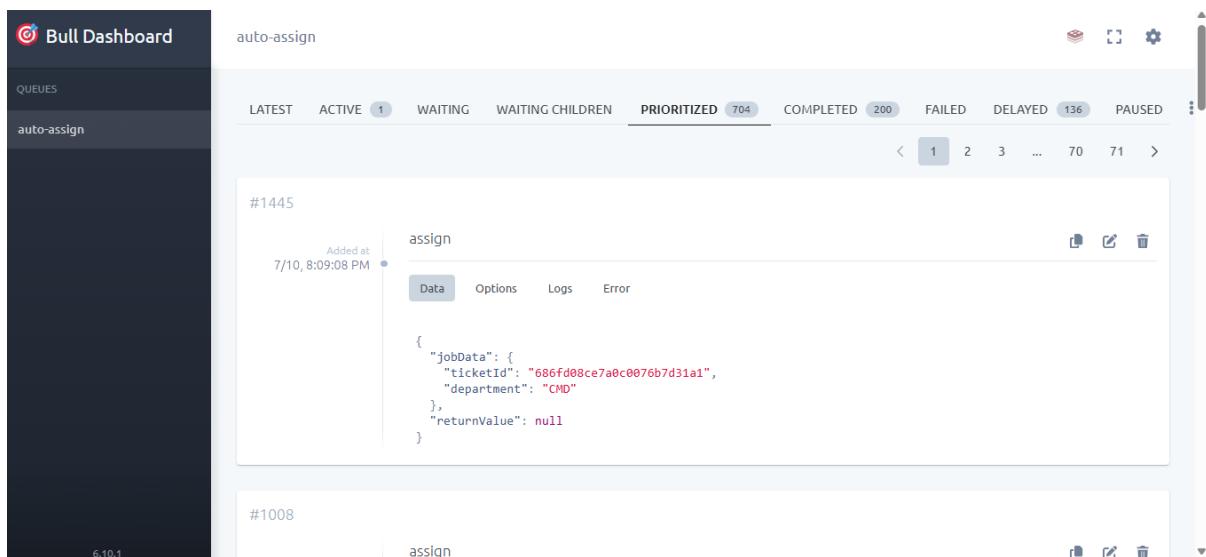


Figure 34: BullMQ Management UI

## 9 File Upload

### 9.1 Overview

The **File Upload** feature allows admin users to upload ticket data in bulk using CSV or Excel files. This streamlines the process of adding or updating multiple tickets at once, improving efficiency and reducing manual entry errors.

### 9.2 API Endpoints

- POST /admin/tickets/upload: Upload file to insert/update ticket records.

### 9.3 User Flow

1. **Access Upload Page:** Admin navigates to the upload page in the admin dashboard.
2. **Select File:** Admin selects a `.csv`, `.xlsx`, or `.xls` file containing ticket data.
3. **Upload:** Admin clicks the upload button. The file is sent to the backend for processing.
4. **Feedback:** The system displays a status message indicating success or failure, and may show a preview of the uploaded data.

### 9.4 File Upload Sequence Diagram

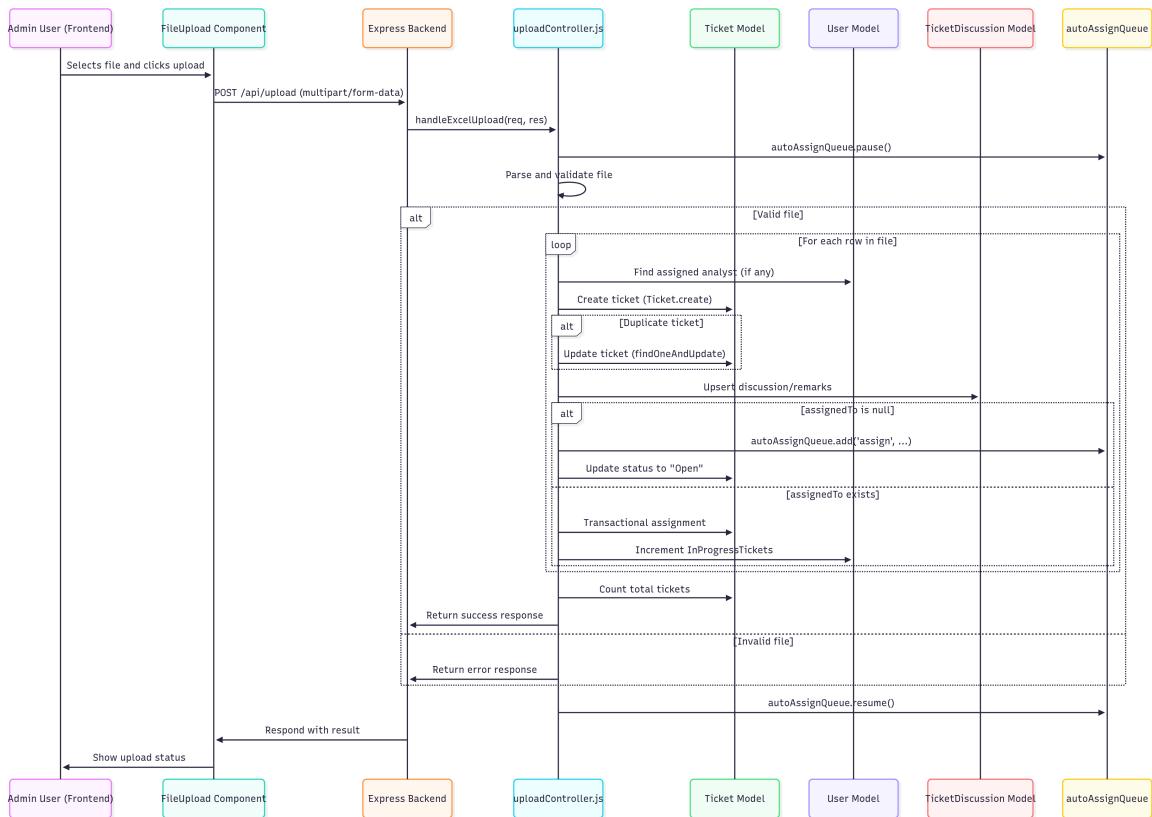


Figure 35: File Upload Sequence Diagram

### 9.5 Supported File Types

- CSV (`.csv`)
- Excel (`.xlsx`, `.xls`)

### 9.6 Required Columns

The uploaded file must contain at least the following columns:

- `ticketNumber`
- `customerName`
- `description`

Additional columns (e.g., `status`, `priority`, `assignedAnalystId`, etc.) are optional but recommended for richer ticket data.

## 9.7 Backend Processing

1. **File Validation:** The backend checks the file type and ensures required columns are present.
2. **Parsing:** The file is parsed into JSON objects, with column headers normalized for consistency.
3. **Data Validation:** Each row is checked for required fields and valid data formats (e.g., dates, enums).
4. **Ticket Handling:**
  - **Insert:** New tickets are created if the `ticketNumber` does not exist.
  - **Update:** Existing tickets are updated if the `ticketNumber` matches an existing record.
5. **Assignment:** If an `assignedAnalystId` is provided, the ticket is assigned accordingly. Otherwise, it may be queued for auto-assignment.
6. **Response:** The backend returns a summary including the number of tickets inserted, updated, and the total ticket count.

## 9.8 Error Handling

- **Missing File:** Returns an error if no file is uploaded.
- **Invalid File Type:** Returns an error if the file is not a supported format.
- **Missing Required Columns:** Returns an error listing missing columns.
- **Server Errors:** Returns a generic error message if processing fails.

## 9.9 Security & Permissions

- Only authenticated admin users can access the file upload feature.
- Uploaded files are stored securely on the server and processed immediately.

## 9.10 Benefits

- **Bulk Operations:** Quickly add or update large numbers of tickets.
- **Consistency:** Enforces required fields and data formats.
- **Efficiency:** Reduces manual data entry and potential for errors.

## 9.11 Example Status Messages

- File uploaded successfully!
- ✗ Upload failed. Please try again.
- ✗ File is missing required columns: ticketNumber, customerName

## 9.12 Extensibility

- **Preview Data:** Optionally show a preview of parsed data before final upload.
- **Template Download:** Provide a downloadable template to guide users in formatting their files.
- **Row-level Error Reporting:** Highlight specific rows with errors for easier correction.

## 9.13 File Upload UI

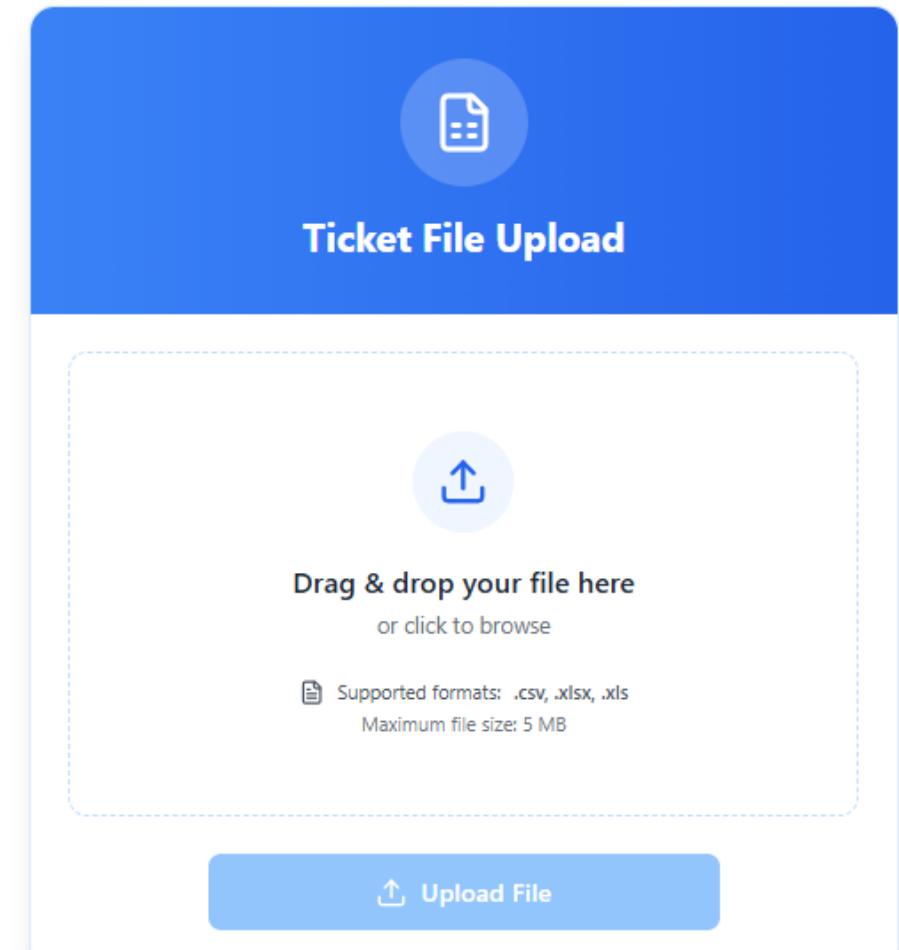


Figure 36: Ticket File Upload UI

## 9.14 Running the Ticket Alerting System

The Ticket Alerting System consists of both frontend and backend components. Follow these steps to run the project:

### 1. Quick Setup (Recommended):

```
# Install all dependencies for both frontend  
# and backend at once  
npm run install-all  
  
# Start both frontend and backend servers with  
# a single command  
npm run dev
```

### 2. Alternative: Manual Setup

If the quick setup fails, follow these standard steps instead:

#### (a) Backend Setup:

```
# Navigate to the backend directory  
cd backend  
  
# Install all required dependencies  
npm install  
  
# Start the backend server  
npm run dev
```

#### (b) Frontend Setup:

```
# Navigate to the frontend directory  
cd frontend  
  
# Install all required React  
# dependencies  
npm install  
  
# Start the React development server  
npm run dev
```

#### (c) Redis Setup (WSL):

```
# Install Redis server on WSL (Windows  
# Subsystem for Linux)  
sudo apt update  
sudo apt install redis-server  
  
# Start the Redis server service  
sudo service redis-server start
```

```
# Test if Redis server is running  
# correctly  
# Should return "PONG" if successful  
redis-cli ping  
  
# Optional: Check Redis server status  
sudo service redis-server status
```

## 9.15 Sample Credentials

All users share the same password: TechMahindra@123

### Superadmin

- superadmin@techmahindra.com

## 9.16 Sample Credentials

All users share the same password: TechMahindra@123

### Superadmin

- superadmin@techmahindra.com

### CMP Department

#### Admins:

- admin1.cmp@techmahindra.com
- admin2.cmp@techmahindra.com

#### Analysts:

- analyst1.cmp@techmahindra.com
- analyst2.cmp@techmahindra.com
- analyst3.cmp@techmahindra.com
- analyst4.cmp@techmahindra.com
- analyst5.cmp@techmahindra.com

### GLS Department

#### Admins:

- admin1.gls@techmahindra.com
- admin2.gls@techmahindra.com

#### Analysts:

- analyst1.gls@techmahindra.com
- analyst2.gls@techmahindra.com
- analyst3.gls@techmahindra.com
- analyst4.gls@techmahindra.com
- analyst5.gls@techmahindra.com

## SCFU Department

### Admins:

- admin1.scfu@techmahindra.com
- admin2.scfu@techmahindra.com

### Analysts:

- analyst1.scfu@techmahindra.com
- analyst2.scfu@techmahindra.com
- analyst3.scfu@techmahindra.com
- analyst4.scfu@techmahindra.com
- analyst5.scfu@techmahindra.com

## YONO Department

### Admins:

- admin1.yono@techmahindra.com
- admin2.yono@techmahindra.com

### Analysts:

- analyst1.yono@techmahindra.com
- analyst2.yono@techmahindra.com
- analyst3.yono@techmahindra.com
- analyst4.yono@techmahindra.com
- analyst5.yono@techmahindra.com

## 10 Conclusion

This LLD document outlines all critical workflows, API endpoints, UI interfaces, database schema, and backend integration involved in the Ticket Alerting System. It ensures clarity for future development, testing, and deployment.