# FULL STACK DEVELOPMENT WITH MERN

# 1.Introduction:

# Project Title: Resolve Now

**Team Members:**

* Guravani Prasanna Lakshmi – Team Leader
* K. Kiran Kumar – Team Member
* M. Varshini – Team Member
* M. Swathi – Team Member

**2. Project Overview** – Resolve Now

**Purpose**

Introducing Resolve Now – your digital ally in complaint management and public grievance redressal. Whether you're a citizen seeking swift resolution, an admin aiming for process transparency, or an organization striving for responsive governance, Resolve Now empowers every participant to engage in a seamless, structured, and transparent communication channel.With smart complaint filing, real-time tracking, and feedback loops, the platform ensures grievances don’t get buried—they get solved. It’s more than just a portal; it’s a people-first problem-solving solution. Because every voice deserves to be heard—and resolved.

**Description**

Welcome to a new era of civic and organizational accountability—Resolve Now, a digital complaint management platform meticulously engineered using the robust MERN Stack: MongoDB, Express.js, React.js, and Node.js. Tailored for both citizens and institutional stakeholders, ResolveNow transforms traditional, paper-based grievance processes into streamlined, transparent workflows. The platform balances modern technology with user empathy—giving individuals the confidence to speak up and organizations the tools to act. At the core lies MongoDB, enabling structured and scalable storage of user accounts, complaint records, feedback, and resolution logs. On the server side, Express.js and Node.js ensure resilient, event-driven logic capable of handling dynamic complaints and real-time updates. And with React.js, users experience a fluid, mobile-responsive interface whether they’re raising a concern or resolving one. From administrative dashboards and analytics to SMS/email alerts and role-based permissions—ResolveNow isn't just a system; it’s a full-cycle trust engine designed to turn issues into opportunities for improvement. Say goodbye to long queues, lack of transparency, and follow-up fatigue. Say hello to ResolveNow—where every complaint finds a clear path to resolution.

**Features**

* User Registration and Authentication:  
  Secure sign-up/login with optional OAuth integration for easy onboarding.
* Complaint Submission:  
  Guided form interface with category, description, attachments, and urgency tagging.
* Complaint Tracking:  
  Real-time visibility of complaint status, assigned authority, and resolution timeline.
* Notification System:  
  Email/SMS/in-app notifications for every complaint lifecycle update.
* Feedback Mechanism:  
  Post-resolution satisfaction rating and optional comment input for quality control.
* Admin Dashboard:  
  View, filter, assign, and manage complaints with visual analytics and internal notes.
* Complaint Assignment Workflow:  
  Smart routing and manual assignment options for department-specific handling.
* Reporting and Analytics:  
  Visual insights on categories, resolution times, volume trends, and satisfaction levels.
* Role-Based Access Control:  
  Separate interfaces and permissions for users, admins, and departmental officers.
* Third-Party Integration Support:  
  Optional tie-ins with SMS/email gateways, chatbots, government databases, or civic portals.

**Scenario-Based Case Study**

Ravi is a university student frustrated by a leaking water pipe in his dorm. Despite reporting it verbally to staff multiple times, no one acted. He's looking for an easier way to officially report and track it—without navigating bureaucracy or chasing updates.

Enter Resolve Now:

* User Registration and Authentication:  
  Ravi creates an account and logs in securely.
* Complaint Submission:  
  He fills a short form selecting “Infrastructure > Water Maintenance” and attaches a photo.

# Complaint Tracking: Within 24 hours, he sees the status change to “Assigned” and later to “In Progress.”

# Notification System: He receives SMS/email alerts every time the complaint moves through a new stage.

# Feedback Mechanism: After the issue is resolved, Ravi receives a prompt to rate the handling. He gives it 4/5 and notes that the plumber arrived quickly.

# Ravi no longer feels ignored—and the admin team gains a tracked record of common issues across campus. That’s the power of Resolve Now.

# 3.Architecture:

**Frontend:**

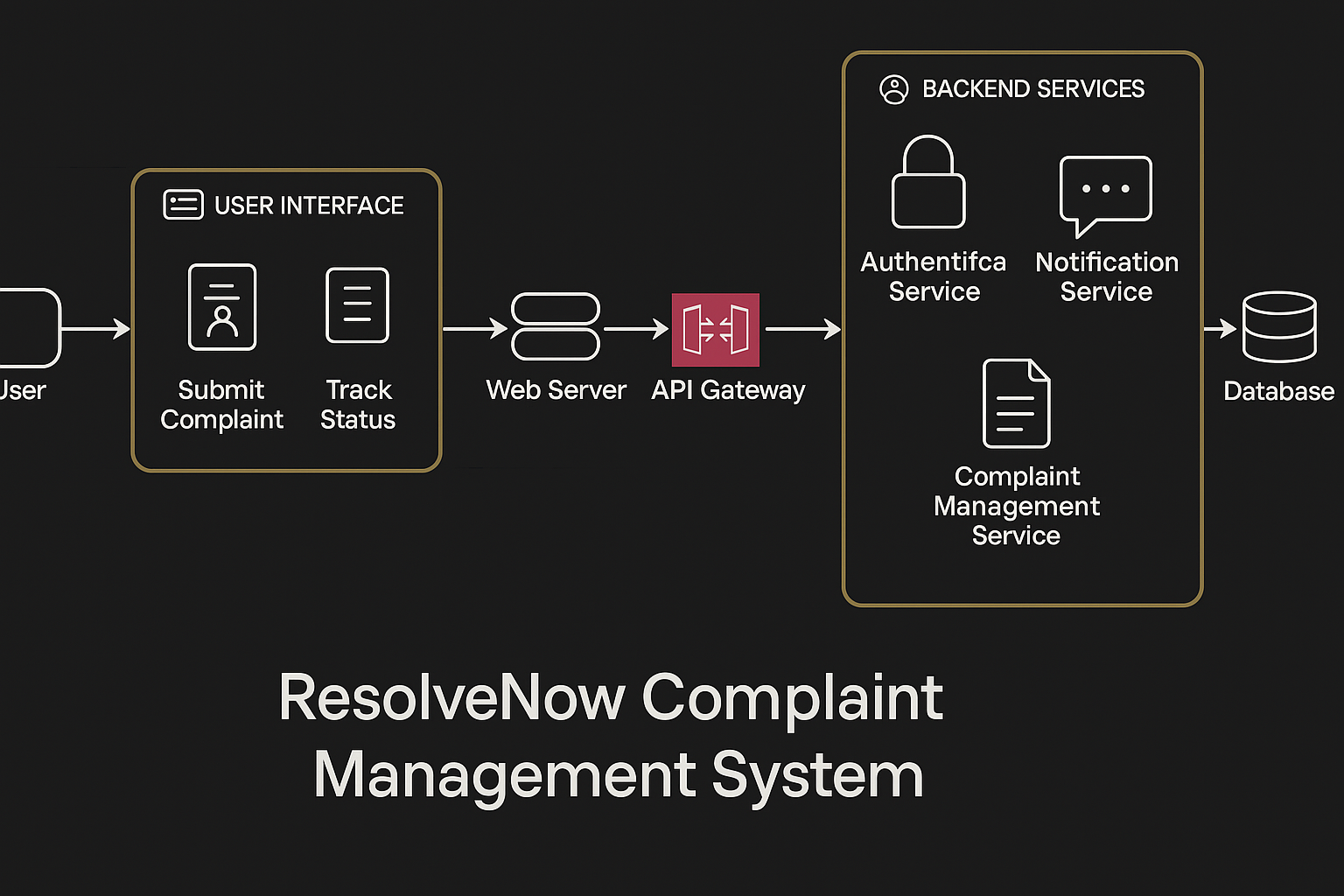
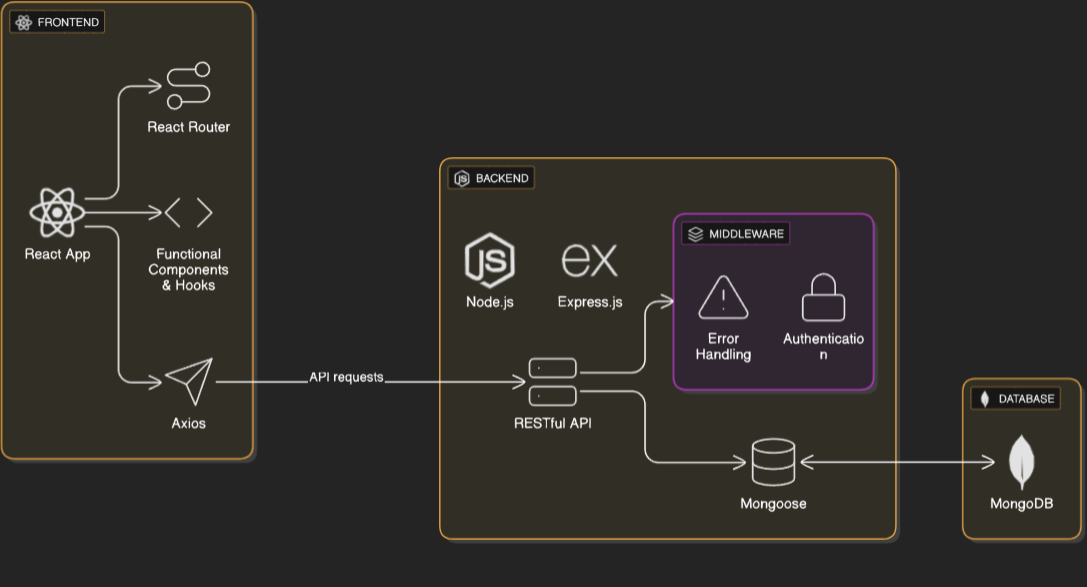
* - Built with React.js
* - Uses functional components and hooks
* - React Router for routing
* - Axios for API requests

**Backend:**

* - Built with Node.js and Express.js
* - RESTful API architecture
* - Middleware for error handling and authentication

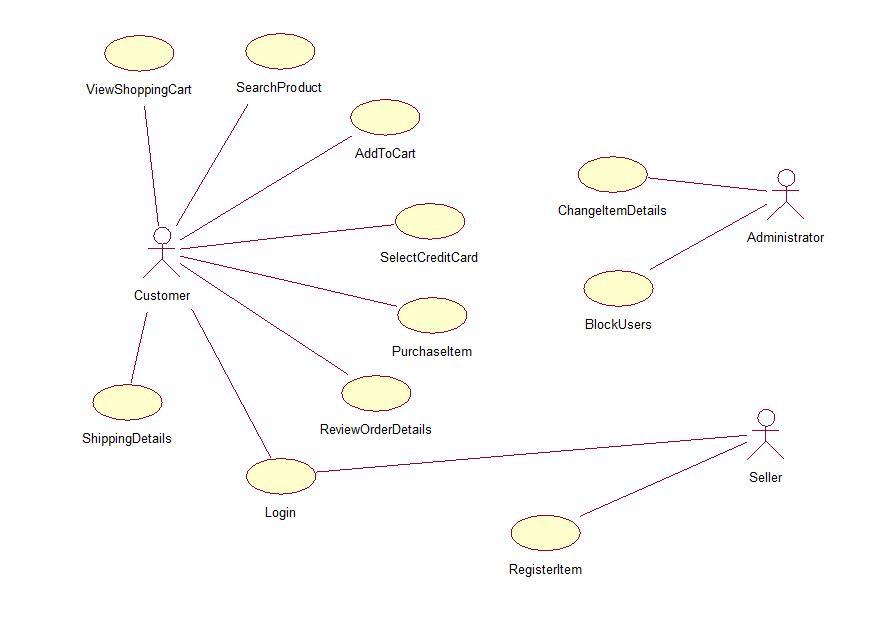
**Database:**

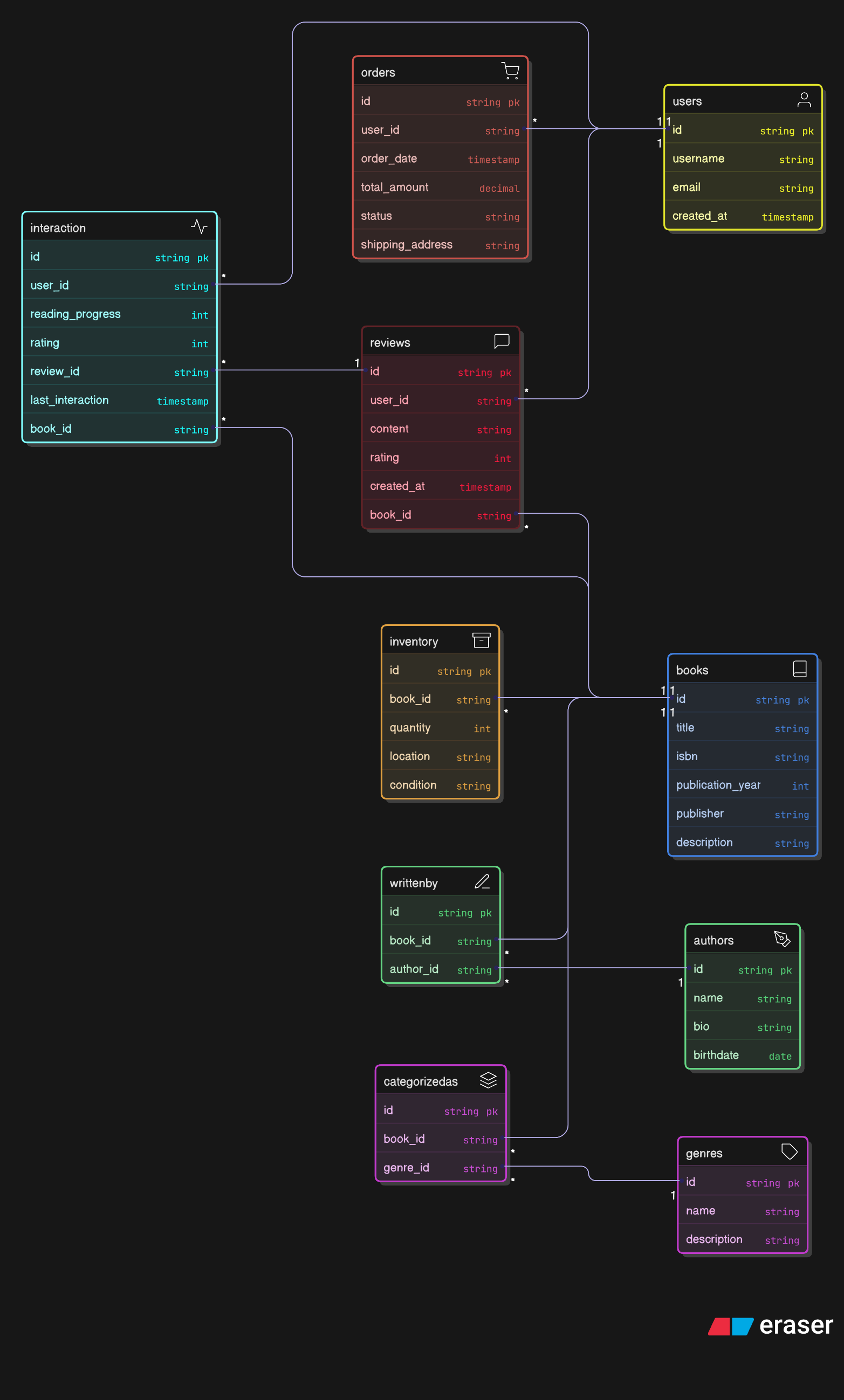
* - MongoDB for data persistence
* - Mongoose for schema definition and ODM



* **User Interface:** The user interface serves as the main platform for complainants and administrators to interact with the system. It allows users to file complaints, view status updates, submit feedback, and navigate through dashboards with ease. The UI is designed to be intuitive and mobile-responsive for accessibility on any device.
* **Web Server:** The web server hosts the frontend interface and manages server-side rendering of complaint forms, dashboards, timelines, and analytics. It ensures smooth communication between users and backend services, delivering a seamless and responsive grievance-handling experience.
* **API Gateway:** The API gateway functions as the centralized point for routing incoming client requests. It manages complaint submissions, authentication calls, user account operations, and communication with backend services such as complaint tracking and feedback processing.
* **Authentication Service:** This component handles user verification, secure sign-up/login, and access control based on roles (user/admin/resolver). It uses techniques like JWT-based session management and optional OAuth integration to protect system access and user data.
* **Database:** The database stores structured records such as user profiles, complaints, timelines, assignment logs, admin notes, and user-submitted feedback. MongoDB is used for its scalability, flexibility in schema design, and support for real-time updates.
* **View Complaints:**  
  This feature enables users to view all their submitted complaints along with their statuses, progress history, and any resolution updates. Each complaint entry displays relevant metadata for transparency and traceability.
* **Category Selection:**  
  Users can categorize complaints by predefined sectors (e.g., maintenance, administration, services) for better routing. Admins can filter by these categories to prioritize issue resolution more efficiently.
* **Complaint Management Service:**  
  This service oversees the life cycle of each complaint—from registration to assignment and resolution. It ensures that complaints are routed to the appropriate admin teams and keeps logs of all interactions and status updates.
* **Resolution Tracking Service:**  
  This component enables real-time monitoring of complaint progress. It synchronizes updates from resolvers/admins to the user interface and triggers notifications accordingly. It also stores timestamped status transitions for auditing.

**ER-Diagram:**

****



**User–Complaint Relationship**

* **Type:**One-to-Many(1:M)  
  A single user can submit multiple complaints, but each complaint belongs to one user.
* **Implementation:**  
  Store a UserID foreign key in the Complaint table.  
  This allows filtering complaints submitted by a specific user and linking status history.

**Complaint–Status History Relationship**

* **Type:**One-to-Many(1:M)  
  A complaint can go through multiple status updates (e.g., Submitted → In Progress → Resolved).
* **Implementation:**  
  Create a Complaint Status table containing fields like:
  + ComplaintID (FK)
  + Status (e.g., New, Assigned, Resolved)
  + Timestamp
  + UpdatedBy (admin ID)

**Admin–Complaint Assignment Relationship**

* **Type:** Many-to-Many (M:M)  
  An admin may handle multiple complaints, and a complaint may be reassigned to different admins.
* **Implementation:**  
  Use an intermediate Assignment table:
  + AdminID (FK)
  + ComplaintID (FK)
  + AssignmentDate
  + Notes or reassign reason

**Complaint–Feedback Relationship**

* **Type:** One-to-One (1:1)  
  Each resolved complaint may have one feedback submission from the user.
* **Implementation:**  
  Link a Feedback table with a Complaint ID (unique FK constraint) containing:
  + Rating
  + Comments
  + Submission Date

**User–Feedback Relationship**

* **Type:** One-to-Many (1:M)  
  A user may give feedback for different complaints over time.
* **Implementation:**  
  Include UserID as a foreign key in the Feedback table for tracking user sentiment over time.

**Admin–Department Relationship**

* **Type:** One-to-Many (1:M)  
  Each department has multiple admins, but each admin belongs to one department.
* **Implementation:**  
  Store a DepartmentID FK in the Admin table. Useful for department-based complaint routing.

**Complaint–Category Relationship**

* Type: Many-to-One (M:1)  
  Many complaints fall under the same category (e.g., Infrastructure, Water, IT Support).
* Implementation:  
  Store a CategoryID FK in the Complaint table and define categories in a separate lookup table.

## 4.Setup Instructions:

## Pre requisites:

* - Node.js (vX.X.X)
* - MongoDB
* - npm or yarn

**Installation:**

```bash  
git clone https://github.com/your-repo/project-name.git  
cd project-name  
  
cd client  
npm install  
  
cd./server  
npm install  
  
# Create a .env file and set up environment variables

To develop a full-stack Book Store App using React js, Node.js, Express js and MongoDB, there are several prerequisites you should consider. Here are the key prerequisites for developing such an application:

Node.js and npm: Install Node.js, which includes npm (Node Package Manager), on your development machine. Node.js is required to run JavaScript on the server side.

• **Download:** <https://nodejs.org/en/download/>

• **Installation instructions:**<https://nodejs.org/en/download/package-manager/>

**MongoDB:** Set up a MongoDB database to store hotel and booking information. Install MongoDB locally or use a cloud-based MongoDB service.

• **Download:**<https://www.mongodb.com/try/download/community>

• **Installation instructions:**<https://docs.mongodb.com/manual/installation/>

**Express.js:** Express.js is a web application framework for Node.js. Install Express.js to handle server-side routing, middleware, and API development.

**• Installation:** Open your command prompt or terminal and run the following

**command:** npm install express

**React js:** React is a JavaScript library for building client-side applications.

And Creating Single Page Web-Application

**Getting Started**

Create React App is an officially supported way to create single-page React applications. It offers a modern build setup with no configuration.

Quik Start

npm create vite@latest

cd my-app

npm install

npm run dev

If you've previously installed create-react-app globally via npm install -g create-react-app, we recommend you uninstall the package using npm uninstall -g create-react-app or yarn global remove create-react-app to ensure that npx always uses the latest version.

**Create a new React project:**

• Choose or create a directory where you want to set up your React project.

• Open your terminal or command prompt.

• Navigate to the selected directory using the cd command.

• Create a new React project by running the following command: npx create-react-app your-app-name. Wait for the project to be created:

• This command will generate the basic project structure and install the necessary dependencies

**Navigate into the project directory:**

• After the project creation is complete, navigate into the project directory by running the following command: cd your-app-name

**Start the development server:**

• To launch the development server and see your React app in the browser, run the following command: npm run dev

• The npm start will compile your app and start the development server.

• Open your web browser and navigate to <https://localhost:5173> to see your React app.

You have successfully set up React on your machine and created a new React project. You can now start building your app by modifying the generated project files in the src directory.

Please note that these instructions provide a basic setup for React. You can explore more ad- vanced configurations and features by referring to the official React documentation: <https://react.dev/>

HTML, CSS, and JavaScript: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

**Database Connectivity:** Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations.

**Front-end Library:** Utilize React to build the user-facing part of the application, including products listings, booking forms, and user interfaces for the admin dashboard.

**Version Control:** Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

• **Git:** Download and installation instructions can be found at:<https://git-scm.com/downloads>

**Development Environment:** Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

• **Visual Studio Code:** Download from <https://code.visualstudio.com/download>

• **Sublime Text:** Download from <https://www.sublimetext.com/download>

• **WebStorm:** Download from <https://www.jetbrains.com/webstorm/download>

**Roles and Responsibility**

**User:**

**• Registration:** Users can create an account on Resolve Now by providing necessary information such as name, email, and password. This allows them secure access to complaint-related features.

**• Profile Management:** Users can manage and update their profile details including name, contact information, and password for a personalized and secure experience.

* **Complaint Submission:**  
  Users have the ability to file a complaint by selecting relevant categories, describing the issue, and optionally attaching images or supporting documents.
* **Complaint Tracking:**  
  Users can view the real-time status of submitted complaints, including assignment updates, ongoing progress, and final resolution notes.
* **Feedback:**  
  After resolution, users can rate the handling of their complaint and provide feedback to help improve service quality.
* **Notifications:**  
  Users receive timely alerts and updates via email or in-app notifications whenever there is a change in complaint status or resolution message.

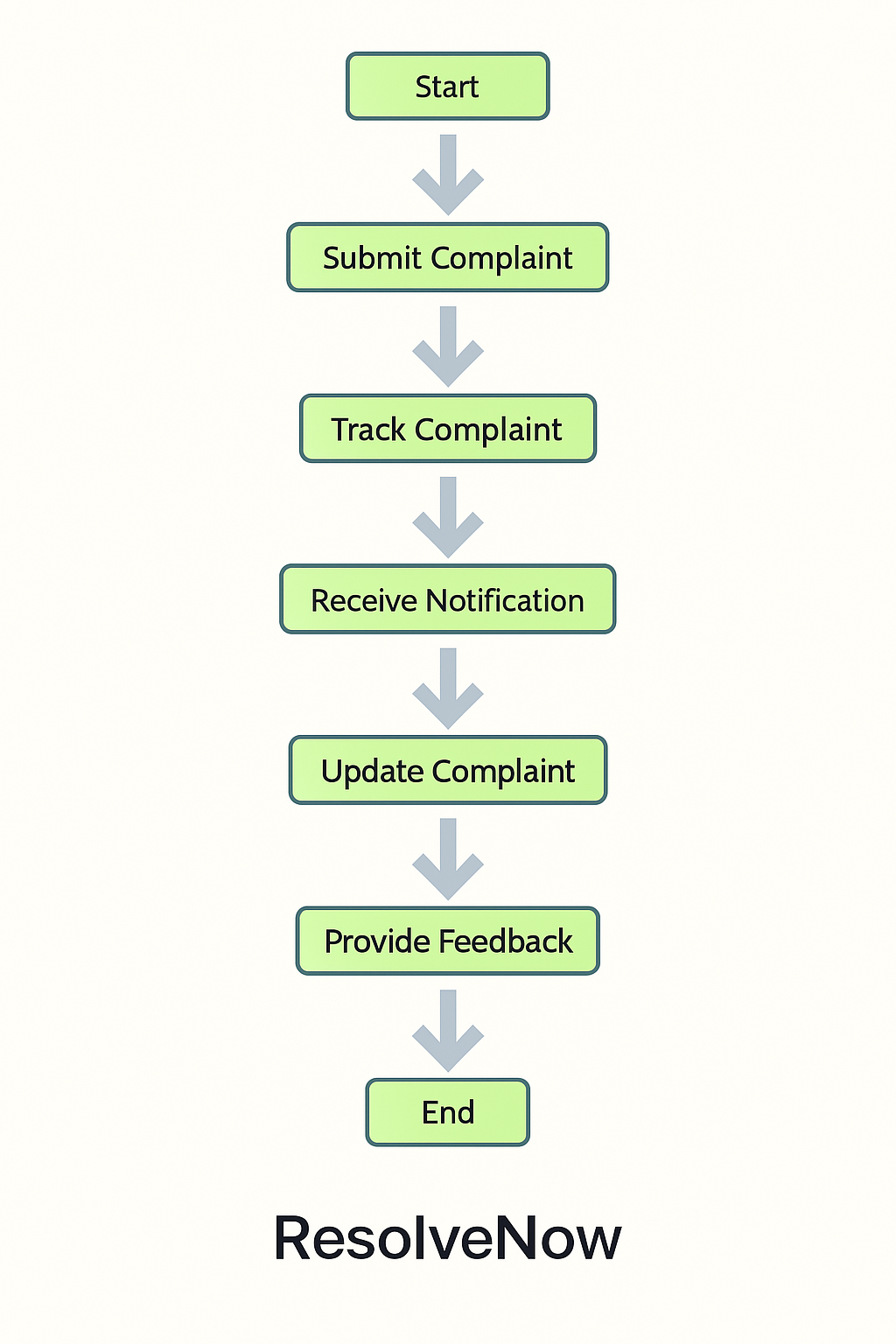
**Seller:**

* **Registration:**  
  Admins and complaint resolvers register an account on the ResolveNow platform by providing required details such as name, official email, department, and role-based authorization credentials. Access is typically granted by a super admin or institution lead.
* **Profile Management:**  
  Admin users can manage and update their profile information including email, department assignment, phone number, and password to ensure secure access.
* **Complaint Management:**  
  Admins and resolvers can view newly submitted complaints, categorize them, assign priorities, add resolution notes, and update their status at every stage of the lifecycle.
* **Assignment & Tracking:**  
  Admins can reassign complaints to specific departments or staff based on category, location, or urgency. They can also track who is currently handling each case with timestamped updates.
* **Feedback Review & Response:**  
  Once a complaint is resolved, admins can view submitted user feedback, assess complaint handling quality, and take further action if low ratings are flagged.
* **Analytics Dashboard:**  
  Admins have access to reports on complaint volume, resolution time, department performance, and common issue trends to help improve processes and services.
* **Logout:**  
  To ensure data security, admins and resolvers can securely log out of the Resolve Now platform after completing their tasks.

**Admin:**

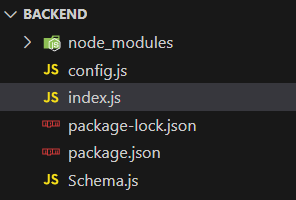
* **System Management:**  
  Admins hold full control over the Resolve Now complaint resolution platform. They oversee core functionalities, ensure configuration integrity, enforce access permissions, and maintain system-wide security protocols for all roles.
* **User Management:**  
  Admins can create, update, or deactivate user accounts across departments and roles. They manage user access levels and respond to user-related escalations or misconduct reports.
* **Complaint Oversight:**  
  Admins monitor all complaint submissions on the platform. They assign complaints to appropriate resolvers, modify statuses, and audit histories to ensure resolution workflows adhere to SLAs.
* **Resolver Team Management:**  
  Admins manage resolver profiles, approve or update departmental assignments, monitor individual performance, and reassign or escalate cases when necessary.
* **Feedback & Quality Control:**  
  Admins can view all user-submitted feedback tied to complaint cases. They analyse trends, identify low-rated resolutions, and initiate internal quality checks or follow-ups.

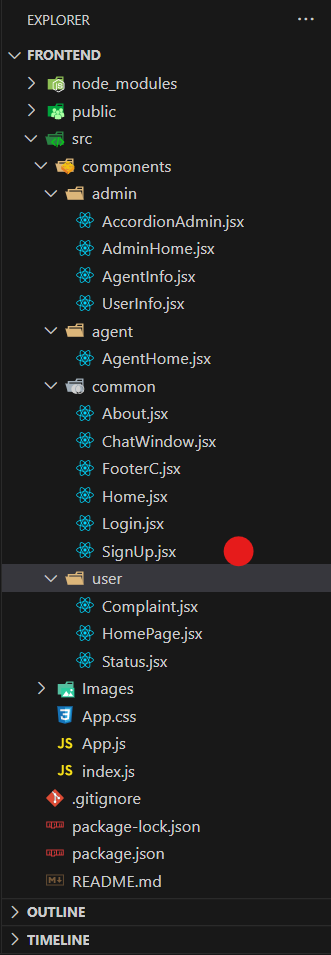
**User Flow:**



* **Start:**  
  Users open the **Resolve Now** application to register a new complaint or check updates on previously submitted issues.
* **Home Page:**  
  Upon logging in, users land on the dashboard, which presents a quick summary of complaint categories, status updates, and navigation options to file a new complaint, track progress, or view feedback.
* **Access Profile:**  
  Users can navigate to their profile to update personal details like name, email, contact number, or view previous submissions and notification preferences.
* **Complaint Submission:**  
  After verifying or updating their profile, users proceed to file a complaint by selecting a category (e.g., Infrastructure, IT, Maintenance), describing the issue, and optionally attaching evidence (e.g., photos or documents).
* **Complaint Tracking:**  
  Once submitted, users can track their complaint in real-time, seeing progress through stages like **Received → Assigned → In Progress → Resolved**, including timestamps and assigned handler info.

**5.Folder Structure:**

****

****

**6.Running the Application**

Use the code in:

Link: [Code](file:///D:\BOOK%20ORDER\Code)

**Milestone 1: Project Setup and Configuration:**

**1. Install required tools and software:**

* Node.js.
* MongoDB.
* Create-react-app.

**2. Create project folders and files:**

* Client folders.
* Server folders.

**3. Install Packages:**

**Frontend npm Packages**

* Axios.
* React-Router –dom.
* Bootstrap.
* React-Bootstrap.

**Backend npm Packages**

* Express.js
* Mongo DB.
* Cors.

Reference Link:

<https://drive.google.com/file/d/1Acv3Lx3PtJcOYkUjREWAzIoC-i6w96Tl/view?usp=drive_link>

**Milestone 2: Backend Development:**

* **Setup express server**

1. Create index.js file in the server (backend folder).
2. Create a .env file and define port number to access it globally.
3. Configure the server by adding cors, body-parser.

* **User Authentication:**
  + Create routes and middleware for user registration, login, and logout.
  + Set up authentication middleware to protect routes that require user authentication.
* **Define API Routes:**
  + Create separate route files for different API functionalities such as users orders, and authentication.
  + Define the necessary routes for listing products, handling user registration and login, managing orders, etc.
  + Implement route handlers using Express.js to handle requests and interact with the database.
* **Implement Data Models:**
  + Define Mongoose schemas for the different data entities like products, users,  and orders.
  + Create corresponding Mongoose models to interact with the MongoDB database.
  + Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.
* **User Authentication:**
  + Create routes and middleware for user registration, login, and logout.
  + Set up authentication middleware to protect routes that require user authentication.
* **Error Handling:**

• Implement error handling middleware to catch and handle any errors that occur during the API requests.

• Return appropriate error responses with relevant error messages and HTTP status codes.

Reference Link: <https://drive.google.com/file/d/14Vut4GVqofFnPO-z1DjWKgQsVq3R4-r1/view?usp=drive_link>

**Milestone 3: Database:**

**1. Configure MongoDB:**

* Install Mongoose.
* Create database connection.
* Create Schemas & Models.

**2. Connect database to backend:**

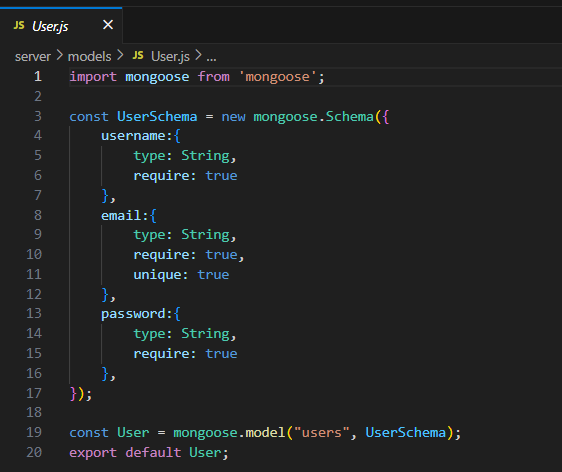
Now, make sure the database is connected before performing any of the actions through the backend. The connection code looks similar to the one provided below.



**3. Configure Schema:**

Firstly, configure the Schemas for MongoDB database, to store the data in such pattern. Use the data from the ER diagrams to create the schemas.

The Schemas for this application look alike to the one provided below.



**Milestone 4: Frontend Development:**

**1. Setup React Application:**

• Create React application.

• Configure Routing.

• Install required libraries.

**2. Design UI components:**

• Create Components.

• Implement layout and styling.

• Add navigation.

**3. Implement frontend logic:**

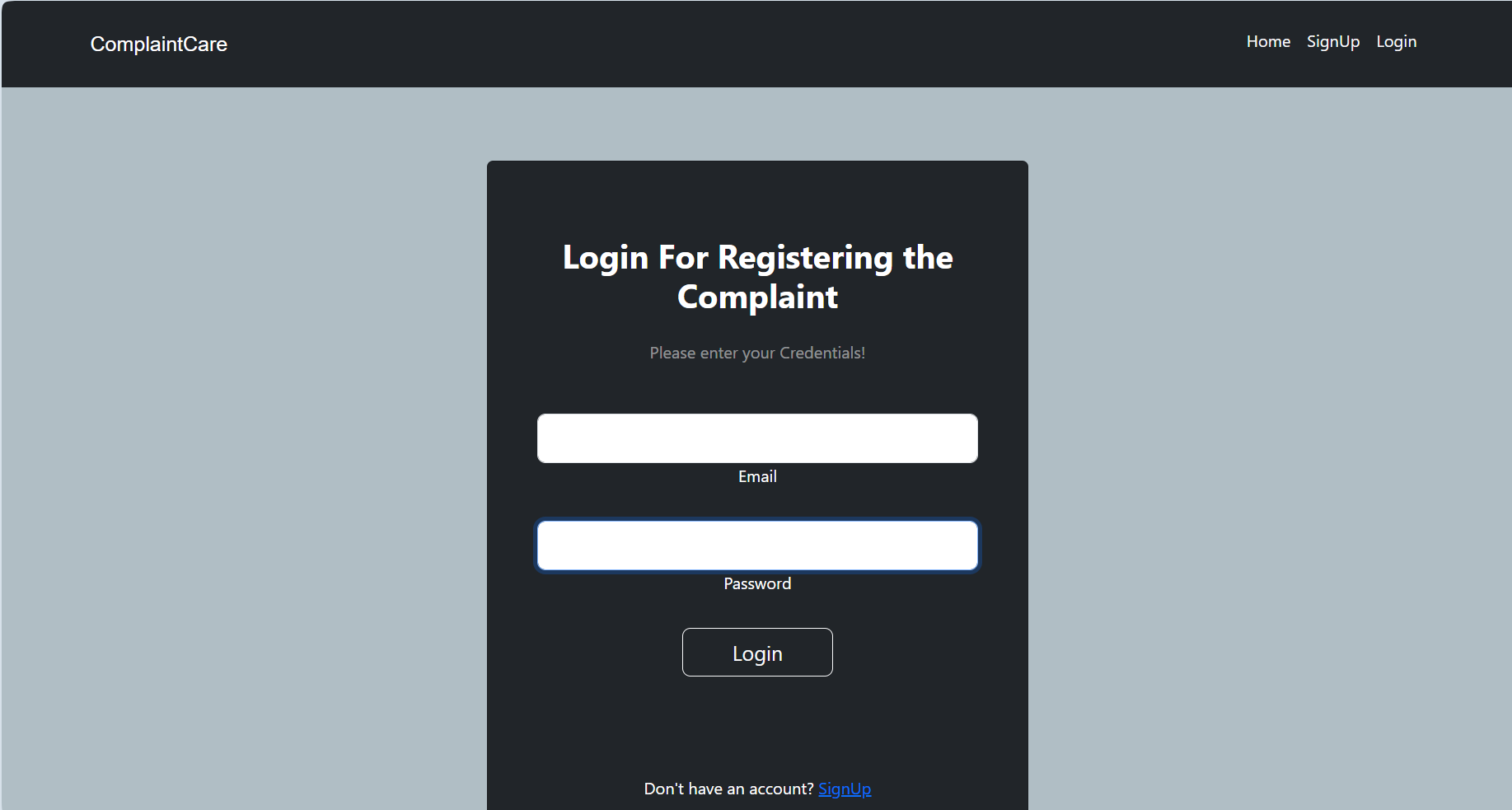
• Integration with API endpoints.

• Implement data binding.

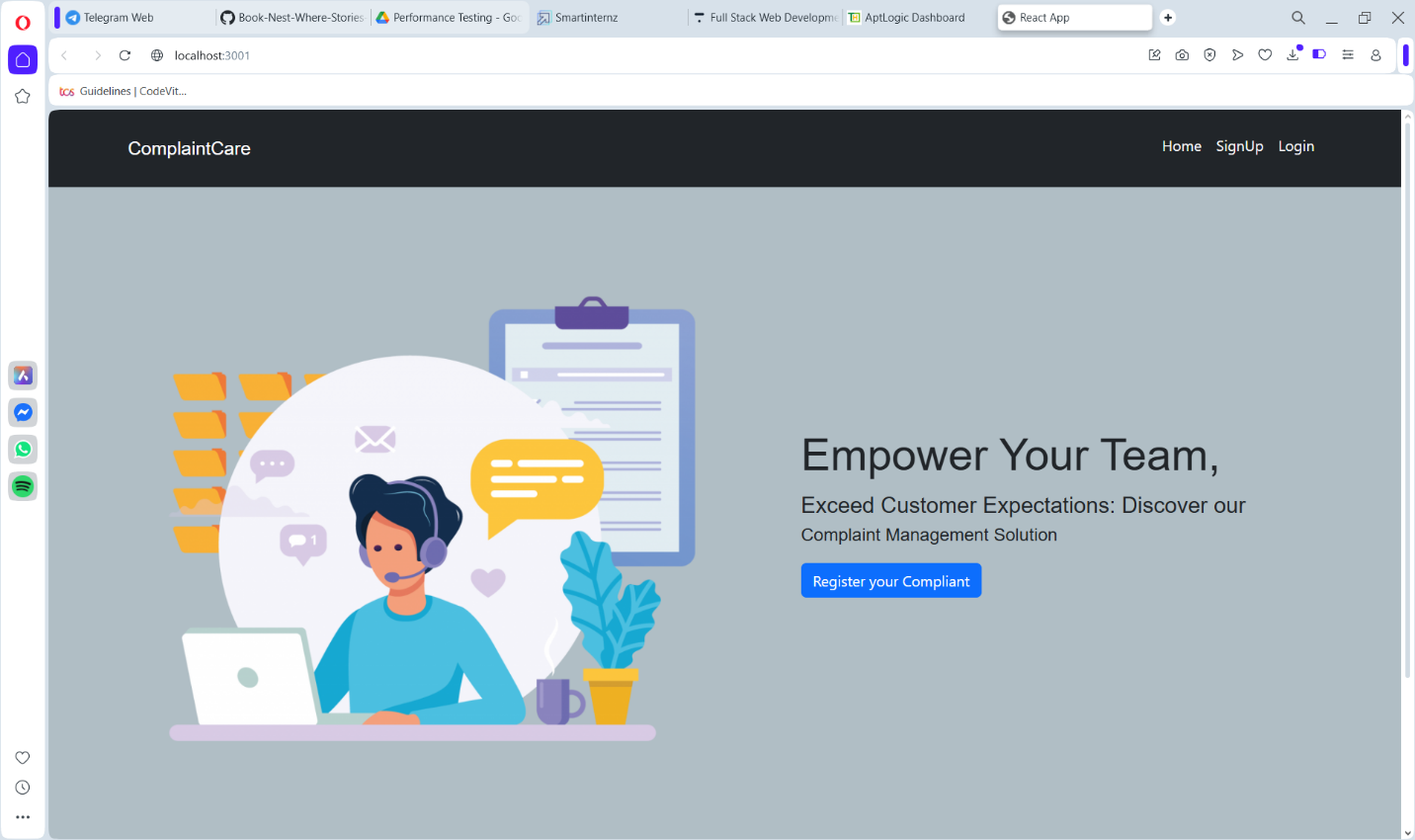
Reference Link:<https://drive.google.com/file/d/1L4aHUedhBJSAmnY2iGsoVhxwXpo5Fz4t/view?usp=drive_link>

**11. Screenshots or Demo:**

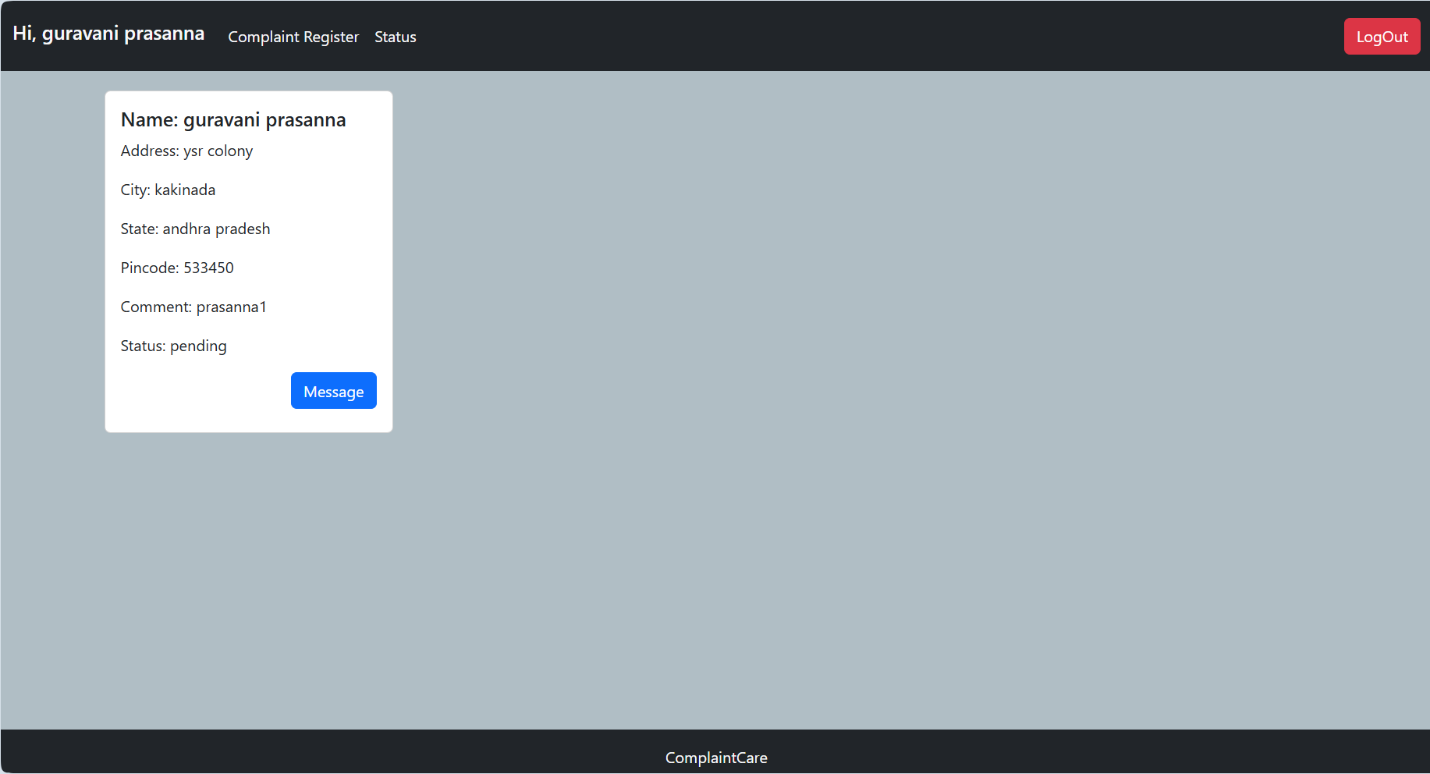
**Login Page:**



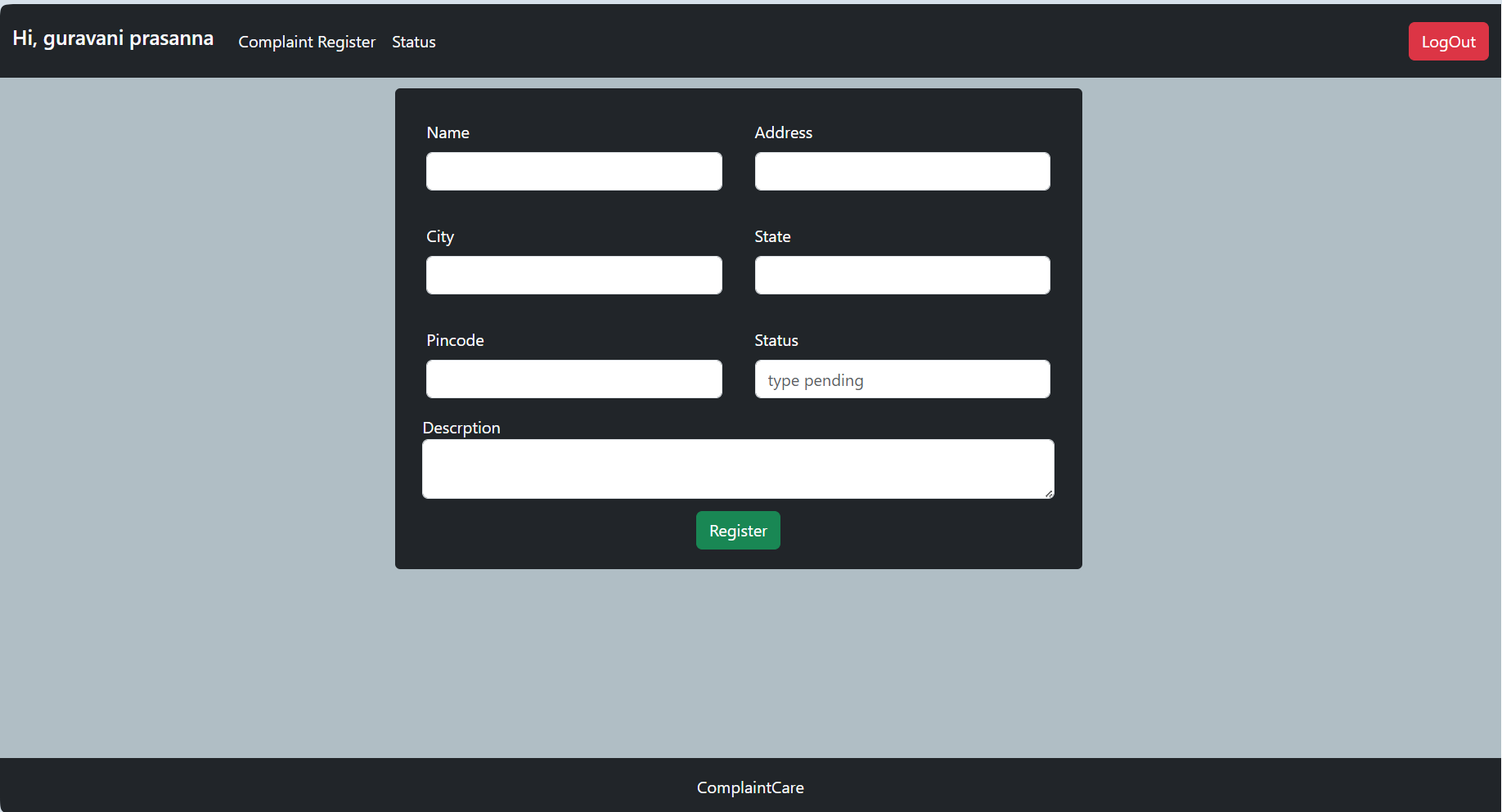
**Home Page:**



**Complaints registered & status**



Complaint registration



**12. Known Issues**

* Intermittent status update delays when server load spikes
* Session timeout issues occasionally force unintended logout
* Limited optimization for mobile browsers on older devices
* Complaint listings do not yet support pagination for large datasets

**13. Future Enhancements**

* Implement detailed user roles and department-based permissions
* Add performance analytics dashboard with export capabilities
* Enable social login options (Google, GitHub) for faster access
* Introduce real-time in-app notifications alongside email alerts
* Add dark mode and accessibility support for inclusive UX
* Integrate AI chatbot to assist users during complaint submission
* API hooks for integration with government or third-party system