Full Stack Development with MERN Project Documentation

# Introduction

* + **Project Title:** Resolve now: your platform for online complaints
  + **Team Members:**

**Team ID:** LTVIP2025TMID52552

**Team Leader:** Tippuluri Hari Krishna Reddy

**Team member:** Yalavala Venkata Ravi Teja

**Team member:** Venu Koniki

**Team member:** Pittala Venkatasai

# Project Overview

* + **Purpose:**

The purpose of Resolve Now is to provide a streamlined, user-friendly platform where individuals can file and manage online complaints against businesses, services, or institutions. The primary goals of the project are to:

* Empower consumers to voice their grievances effectively.
* Facilitate quick and transparent resolution of complaints.
* Bridge communication between complainants and relevant authorities or service providers.
* Improve accountability and customer service standards across industries.

* **Features of Resolve Now: Your Platform for Online Complaints**

1. **User-Friendly Complaint Submission**: Simple forms to submit complaints with category selection, description, and evidence upload.
2. **Real-Time Complaint Tracking**: Track the status and progress of complaints through a personalized dashboard.
3. **Automated Notifications**: Get email/SMS updates at every stage of the complaint resolution process.
4. **Two-Way Communication**: Secure messaging between complainants and service providers for faster resolution.
5. **Anonymous Complaints Option**: Submit complaints without revealing personal identity when needed.
6. **Escalation Mechanism**: Automatically escalate unresolved complaints to higher authorities or regulatory bodies.
7. **Ratings and Feedback System**: Rate the resolution experience and provide feedback on service providers.
8. **Data Analytics and Reports**: Visual insights into complaint trends, resolution times, and satisfaction levels.
9. **Multi-Channel Support**: Access the platform via web, mobile app, or social media integrations.
10. **Secure and Confidential**: Robust data protection to ensure the privacy and security of user information.

# Architecture

* + **Backend:**
* **Set Up Project Structure:** Create a new directory for your project and set up a package.json file using npm init command. Install necessary dependencies such as Express.js, Mongoose, and other required packages.
* **Set Up Project Structure:** Create a new directory for your project and set up a package.json file using npm init  command.Install necessary dependencies such as Express.js, Mongoose, and other required packages.
* **Create Express.js Server:** Set up an Express.js server to handle HTTP requests and serve API endpoints.Configure middleware such as body-parser for parsing request bodies and cors for handling cross-origin requests.
* **Define API Routes:** Create separate route files for different API functionalities such as authentication, stock actions, and transactions.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 Bank, users, transactions, deposits and loans.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:** Implement user authentication using strategies like JSON Web Tokens (JWT) or session-based authentication. Create routes and middleware for user registration, login, and logout. Set up authentication middleware to protect routes that require user authentication.

#### **Handle new transactions:** Allow users to make transactions to other users using the user’s account id. Update the transactions and account balance dynamically in real-time.

#### **Admin Functionality:** Implement routes and controllers specific to admin functionalities such as fetching all the data regarding users, transactions, stocks and orders.

#### **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.

#### Frontend:

#### **Setup React Application:** Bringing Customer Care Registry to life involves a three-step development process. First, a solid foundation is built using React.js. This includes creating the initial application structure, installing necessary libraries, and organizing the project files for efficient development. Next, the user interface (UI) comes to life. To start the development process for the frontend, follow the below steps. Install required libraries. Create the structure directories.

#### **Design UI components:** Reusable components will be created for all the interactive elements you'll see on screen, from stock listings and charts to buttons and user profiles. Next, we'll implement a layout and styling scheme to define the overall look and feel of the application. This ensures a visually-appealing and intuitive interface.  Finally, a navigation system will be integrated, allowing you to effortlessly explore different sections of Customer Care Registry, like making specific complaints or managing your Product complaints.

#### **Implement frontend logic:** In the final leg of the frontend development, we'll bridge the gap between the visual interface and the underlying data. It involves the below stages.

* Integration with API endpoints.
* Implement data binding.
* **Database:**

**User Schema:** The user schema defines the structure of user data stored in the database. It includes fields such as name, email, password, phone, and userType. Each user must provide a name, email, password, phone number, and userType (e.g., customer, agent, admin). User data is stored in the "user\_Schema" collection in the MongoDB database.

**Complaint Schema:** The complaint schema specifies the format of complaint data registered by users. It contains fields like userId, name, address, city, state, pincode, comment, and status. Complaints are associated with users through the userId field, and each complaint must have a name, address, city, state, pincode, comment, and status. Complaint data is stored in the "complaint\_schema" collection in the MongoDB database.

**Assigned Complaint Schema:** The assigned complaint schema defines how complaints are assigned to agents for resolution. It includes fields such as agentId, complaintId, status, and agentName. Each assigned complaint is linked to a specific agent (identified by agentId) and complaint (identified by complaintId). The status field indicates the current status of the assigned complaint. Assigned complaint data is stored in the "assigned\_complaint" collection in the MongoDB database.

**Chat Window Schema:** The chat window schema governs the structure of messages exchanged between users and agents regarding specific complaints. It comprises fields like name, message, and complaintId. Messages are associated with a complaint through the complaintId field, allowing for easy tracking and retrieval of chat history for each complaint. Message data is stored in the "message" collection in the MongoDB database.

# Setup Instructions

* + **Prerequisites:**

Here are the key prerequisites for developing a full-stack application using Node.js, Express.js, MongoDB, React.js:

**Node.js and npm:**

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server-side. It provides a scalable and efficient platform for building network applications.

Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

Download: https://nodejs.org/en/download/

Installation instructions: https://nodejs.org/en/download/package-manager/

**Express.js:**

Express.js is a fast and minimalist web application framework for Node.js. It simplifies the process of creating robust APIs and web applications, offering features like routing, middleware support, and modular architecture.

Install Express.js, a web application framework for Node.js, which handles server-side routing, middleware, and API development.

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

**npm install express**

**MongoDB:**

MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-like format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data.

Set up a MongoDB database to store your application's data.

Download: https://www.mongodb.com/try/download/community

Installation instructions: https://docs.mongodb.com/manual/installation/

**React.js:**

React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

Install React.js, a JavaScript library for building user interfaces.

Follow the installation guide: <https://reactjs.org/docs/create-a-new-react-app.html> .

**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. To Connect the Database with Node JS go through the below provided link:

https://www.section.io/engineering-education/nodejs- mongoosejs-mongodb/

**Front-end Framework**: Utilize Reactjs to build the user-facing part of the application, including entering complaints, status of the complaints, and user interfaces for the admin dashboard.

For making better UI we have also used some libraries like material UI and boostrap.

**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>

To run the existing Video Conference App project downloaded from GitHub:

Follow below steps:

Clone the Repository:

* Open your terminal or command prompt.
* Navigate to the directory where you want to store the e-commerce app.
* Execute the following command to clone the repository:

**git clone**: <https://github.com/awdhesh-student/complaint-registery.git>.

**Install Dependencies**:

• Navigate into the cloned repository directory:

cd complaint-registery

• Install the required dependencies by running the following commands:

cd frontend

npm install

cd ../backend

npm install

Start the Development Server:

• To start the development server, execute the following command:

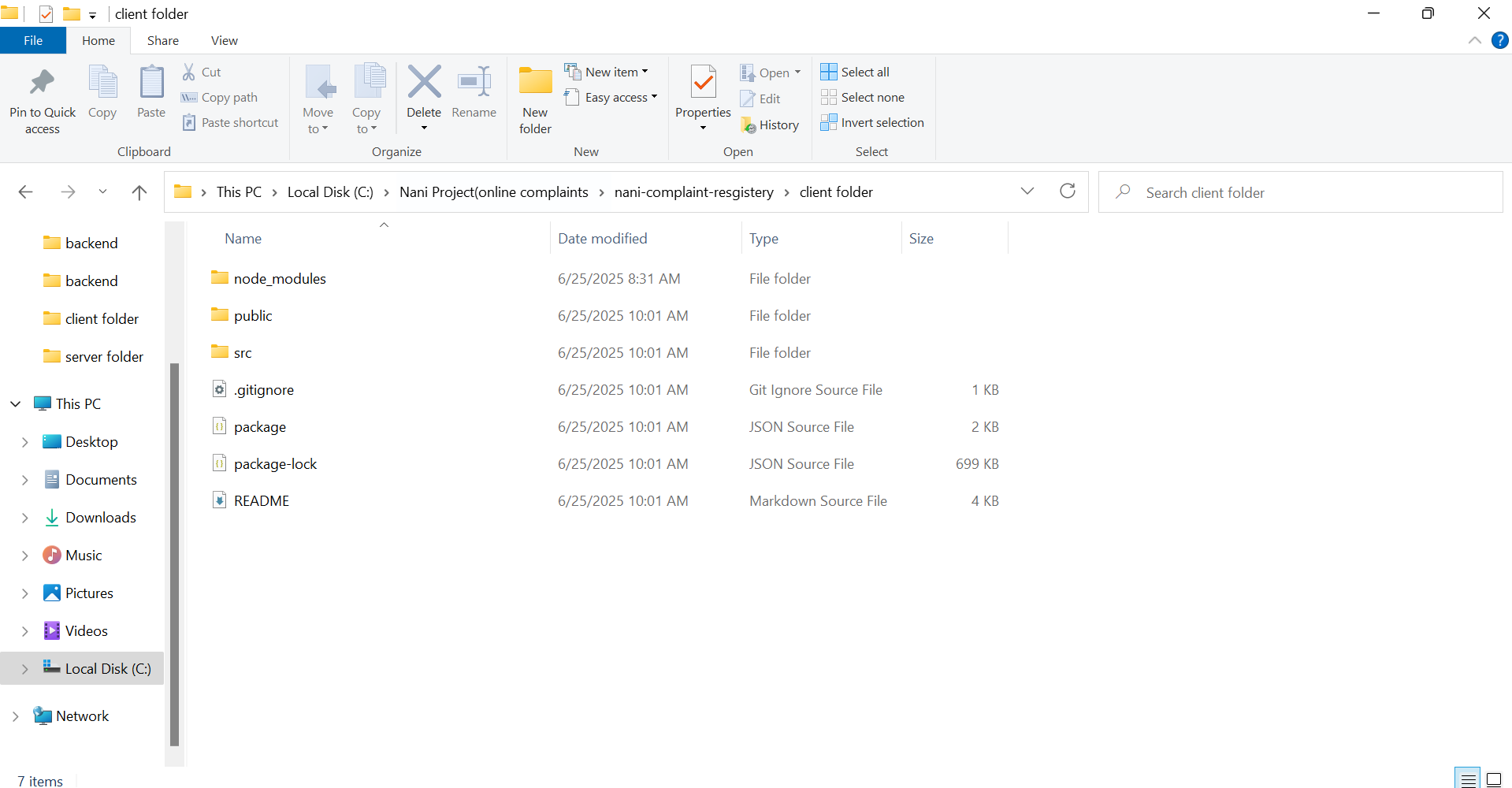
npm start

• The online complaint registration and management app will be accessible at [http://localhost:3000](http://localhost:3000/)

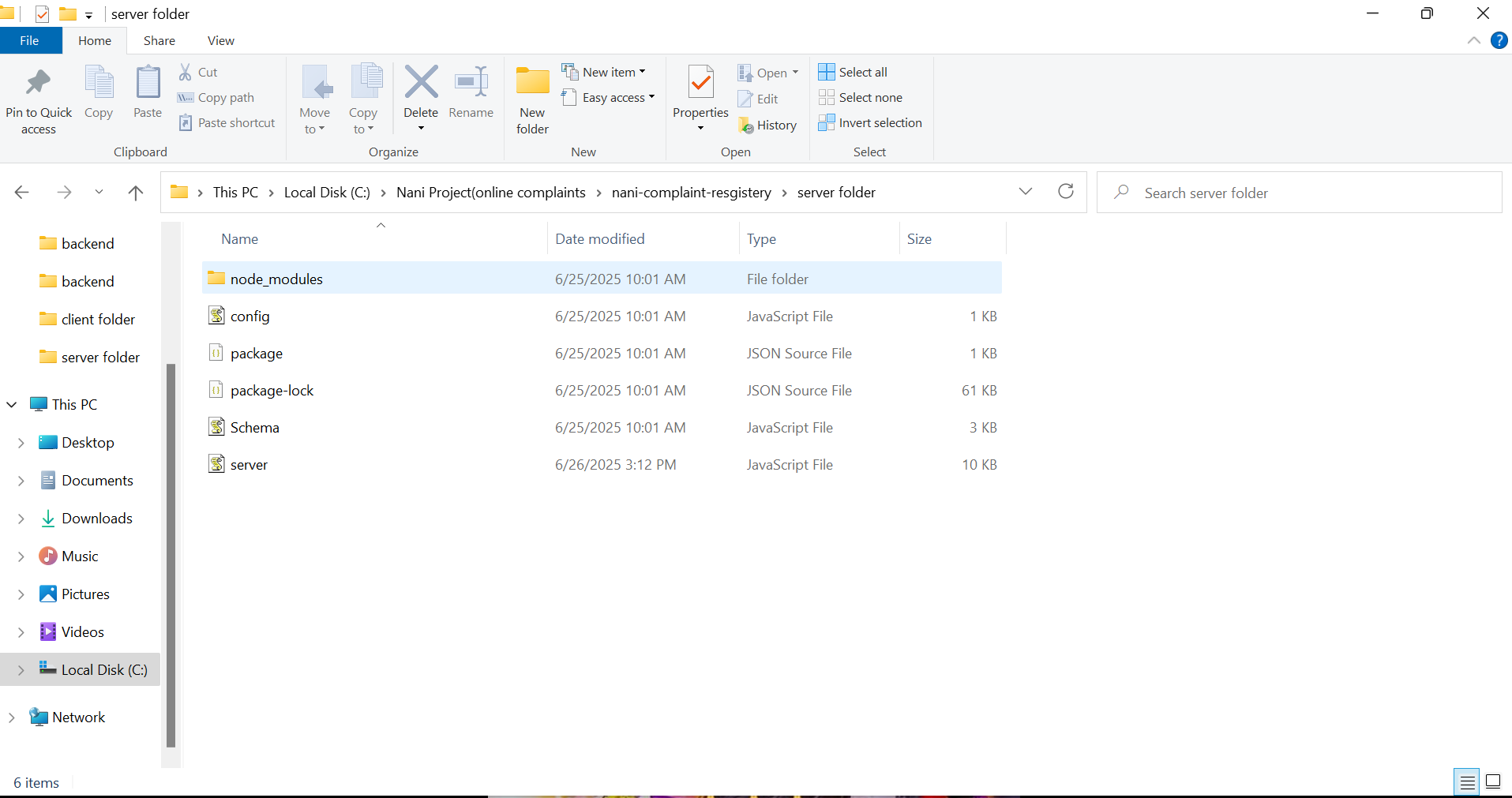
You have successfully installed and set up the online complaint registration and management app on your local machine. You can now proceed with further customization, development, and testing as needed.

# Folder Structure

# Client: Describe the structure of the React frontend.

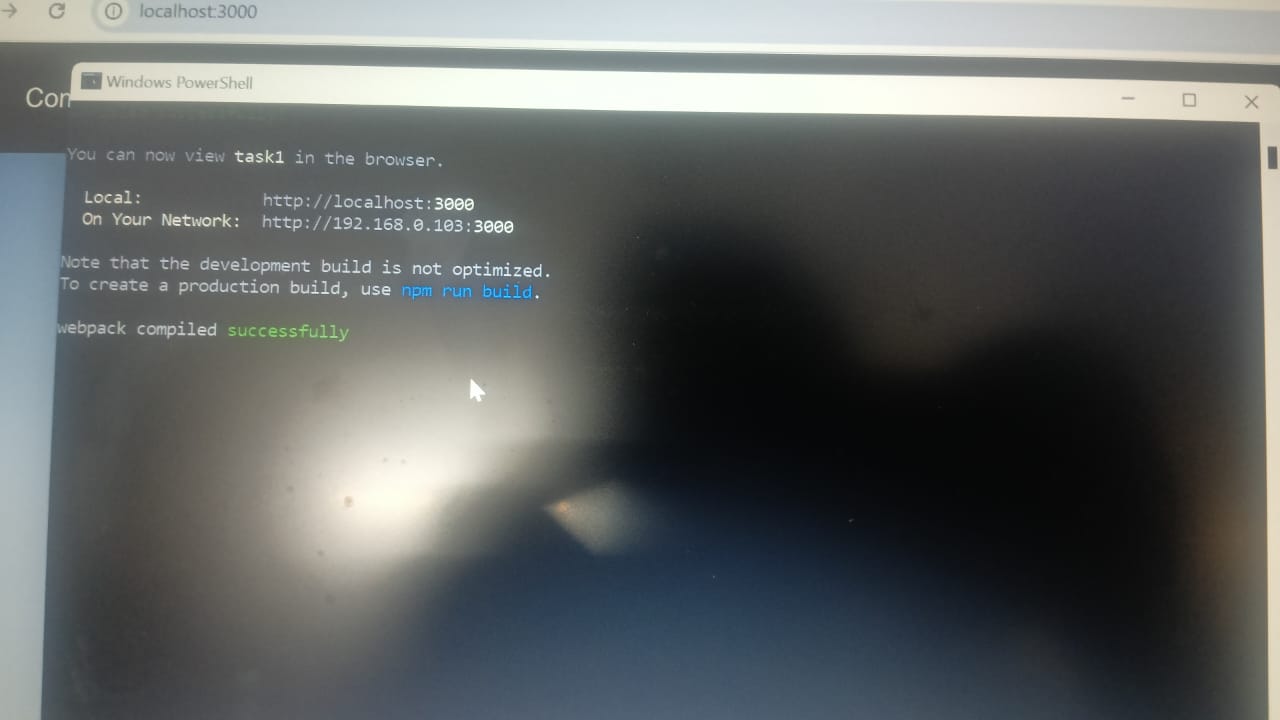


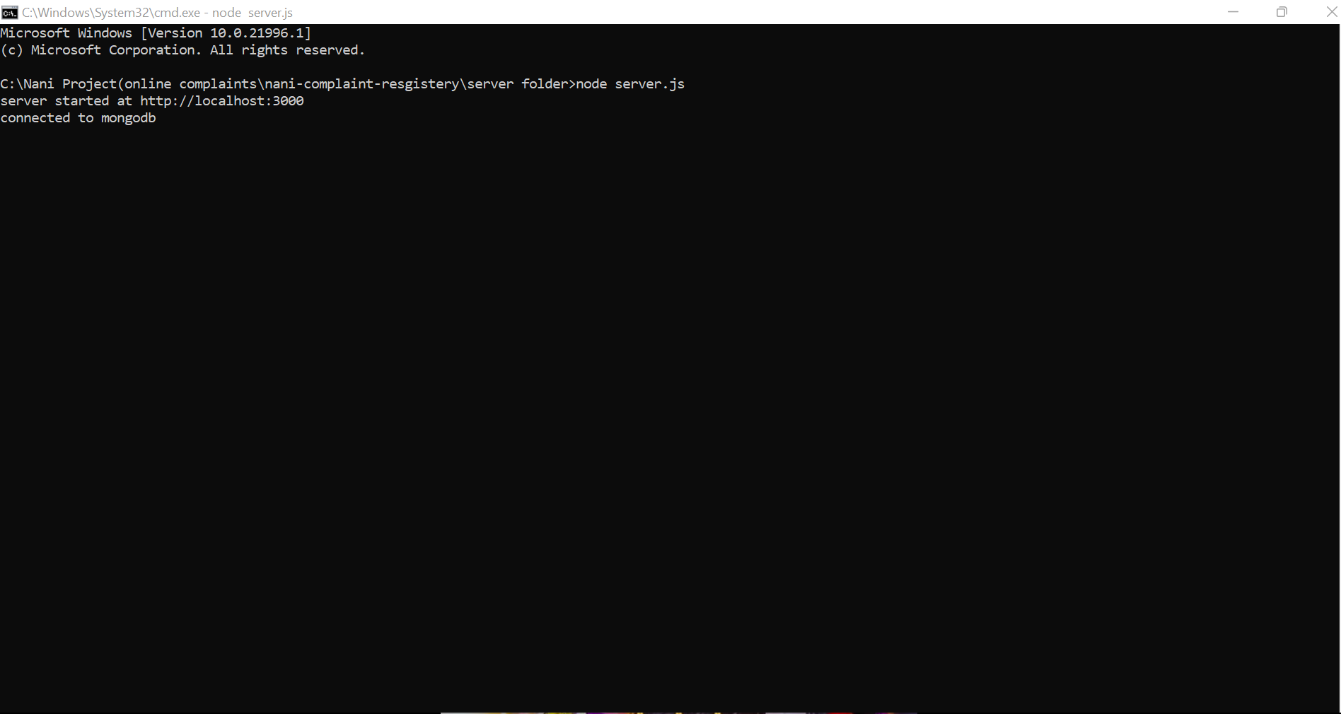
* + **Server:** Explain the organization of the Node.js backend.



# Running the Application

* + Provide commands to start the frontend and backend servers locally.
  + **Frontend:** npm start in the client directory.



**Backend:** npm start in the server directory.

# API Documentation

|  |  |  |  |
| --- | --- | --- | --- |
| Path | Method | Request Body | Parameters |
| /messages | POST | {  name,  message,  complaintId,  } |  |
| /messages/:complaintId | GET |  | complaintId="-createdAt" |
| /SignUp | POST | {  const resultUser = await user.save();  res.send(resultUser);  } |  |
| /Login | POST | const user = await;  if (!user) {  return res.status(401).json({ message: "User doesn`t exists" });  } |  |
| /AgentUsers | GET |  | const { userType } = req.params; |
| /OrdinaryUsers | GET |  | const users = await UserSchema.find({ userType: "Ordinary" }); |
| /OrdinaryUsers/:id | DELETE |  | const { id } = req.params; |

# Screenshots or Demo

# 

**Known Issues:** Known issues in your online complaints project may include limited user engagement due to a lack of real-time feedback or complaint status updates. The system might face challenges in categorizing and prioritizing complaints effectively, especially without automation or AI support. Security vulnerabilities such as weak authentication, lack of CAPTCHA, or unsecured data transmission can expose sensitive user information. Scalability issues may arise if the backend infrastructure is not optimized for handling high complaint volumes. Additionally, the absence of mobile responsiveness or a dedicated app can hinder accessibility. Poor admin tools, such as lack of role-based access or inefficient complaint filtering, can also slow down resolution processes and reduce system efficiency.

**Future Enhancements:** Future enhancements to your online complaints project could include advanced user features like a personalized dashboard for tracking complaint status, multi-language support, and the option for anonymous submissions. Integrating real-time notifications via email or SMS, along with a detailed complaint history, would improve user experience. Automation can be introduced through AI-driven categorization of complaints and chatbot assistance. Implementing sentiment analysis and generating data-driven reports would help identify trends and prioritize critical issues. Strengthening security with two-factor authentication, CAPTCHA, and secure document uploads will ensure user trust. Scalability can be achieved through API integration, mobile app development, and CRM connectivity. Additionally, admin-focused tools such as role-based access, audit trails, and a multi-level escalation system will improve internal efficiency, while features like a public complaint board and resolution ratings can foster transparency and community engagement.