**COMPLAINT-REGISTERY MASTER**

### 1. INTRODUCTION:

PROJECT TITLE: SHOPEZ ONE- STOP SHOP FOR ONLINE PURCHASE

ShopEZ is your one-stop destination for effortless online shopping. With a user-friendly interface and a comprehensive product catalog, finding the perfect items has never been easier. Seamlessly navigate through detailed product descriptions, customer reviews, and available discounts to make informed decisions. Enjoy a secure checkout process and receive instant order confirmation. For sellers, our robust dashboard provides efficient order management and insightful analytics to drive business growth. Experience the future of online shopping with ShopEZ today.

Seamless Checkout Process

Effortless Product Discovery

Personalized Shopping Experience

Efficient Order Management for Sellers

### TEAM MEMBERS:

1.SOWBHAGYA RAMYA DULAM-22MH1A0599

2.DURGARAO TANGILA -22MH1A05F0

3.TEJASWINI PAMPANA -22MH1A05D3

4.DOLAPRIYA KASILINKA -22MH1A0524

### 2. PROJECT OVERVIEW

##### Purpose:

To provide a **secure, centralized platform** for users to easily submit, track, and communicate about complaints. To enable **efficient routing and handling** of issues by automatically assigning them to appropriate agents or departments. To ensure **timely resolution and satisfaction** through real‑time updates, notifications, interaction with agents, and feedback collection.

.

**Features:**

1.Core Complaint Submission & Tracking: **Multi-channel intake**: Allows users to submit complaints via web forms, email, phone, SMS, social media, or mobile apps, all consolidated into a unified system. **Centralized ticketing**: Every complaint lives in a single secure repository with logs of actions, status changes, and attachments. **Attachment support**: Users can attach images, documents, or videos when submitting complaints.

2.User & Agent Interactions: **Customer self‑service portal**: Track complaint status in real-time, view responses, and provide feedback through a secure user interface .**Built‑in messaging/chat**: Enables direct communication between complainants and agents during the resolution process .

**Internal collaboration tools**: Agents and admins can leave private notes, collaborate across teams, and reassign cases when needed .

### 3.ARCHITECT URE:

#### 

##### FRONTEND:

Build using [React.js](http://react.js) the frontend includes user interface components such as User

Admin Authentication and an Admin Dashboard.

* create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

### 

### 

###### COMPONENTS:

* Navbar, Footer: Shared layout elements
* Home, ProductDetails, AllProducts: Page-level components
* ProductCard, CategoryCard: Reusable UI pieces
* Filters, SortOptions: Dynamic control panels

**PAGELAYERS:**

The pages/ folder contains route-level components organized by feature or user role:

* Admin: Admin dashboard pages (manage users, orders, products)
* customer: Customer pages (product browsing, checkout)
* Home.jsx: Landing page combining categories, banners, and product previews
* Authentication.jsx: Login and registration UI.

##### BACKEND:

| **Component** | **Purpose** |
| --- | --- |
| Express.js API routes | Expose complaint & user operations |
| Mongoose models | Structure data, enforce validation & relationships |
| Controllers | Encapsulate business logic (CRUD, routing, notifications) |
| Middleware | Handle authentication, validation, error catching |
| Real-time channels | Enable chat and instant updates via Socket.IO |
| Security layers | Data validation, headers protection, rate limiting, secure secrets |
| Notification subsystem | Email/SMS messaging triggered on key events |

**Implement Data Models:**

• Define Mongoose schemas for the different data entities like products, users,

and models.

• Create corresponding Mongoose models to interact with the MongoDB

database.

• Implement CRUD operations (Create, Read, Update, Delete) for each model to

perform database operations.

#### Database:

The Database section represents the database that stores collections for Users, cart,

Orders and Product.

* Uses MongoDB to store data such as user info, product details, orders, and admin

Credentials.

###### 1. Configure MongoDB

● Install Mongoose

. ● Create database connection.

● Create Schemas & Models.

## ER DIAGRAM:

## 

**Schema:**

• Schema: userSchema

• Model: ‘User’

• The User schema represents the user data and includes fields such as username, email, and password.

• It is used to store user information for registration and authentication purposes.

• The email field is marked as unique to ensure that each user has a unique email address

**2. Product Schema:**

• Schema: productSchema

• Model: ‘Product’

• The Product schema represents the data of all the products in the platform.

• It is used to store information about the product details, which will later be useful

for ordering .

**3. Orders Schema:**

• Schema: ordersSchema

• Model: ‘Orders’

• The Orders schema represents the orders data and includes fields such as userId, product Id, product name, quantity, size, order date, etc.,

• It is used to store information about the orders made by users.

• The user Id field is a reference to the user who made the order.

**4. Cart Schema:**

• Schema: cartSchema

• Model: ‘Cart’

• The Cart schema represents the cart data and includes fields such as userId, product Id, product name, quantity, size, order date, etc.,

• It is used to store information about the products added to the cart by

users.

• The user Id field is a reference to the user who has the product in cart.

**5. Admin Schema:**

• Schema: adminSchema

• Model: ‘Admin’

• The admin schema has essential data such as categories, banner.

### 4.SETUP INSTRUCTIONS:

### PREREQUISITES:

1. Node.js and npm: Node.js is required to run JavaScript on the server side.

2. MongoDB : Install MongoDB locally or use a cloud-based MongoDB service .

###### INSTALLATION:

###### 

**Clone the repository:**

• Open your terminal or command prompt.

• Navigate to the directory where you want to store the-commerce

• Execute the following command to clone the repository

Gitclone: https://github.com/Ramyadulam/online-complaint.git

##### Install Dependencies:

• Navigate into the cloned repository directory:

Cd onlincomplaint**—e-commerce-App-MERN**

• Install the required dependencies by running the following command

cd client

npm install

* Axios
* Bootstrap
* React
* React-router-dom
* React-icons
* React-scripts
* web-vitals

**Start the Development Server:**

cd…/server

npm install

* Bcrypt
* Body-parser
* Cors
* Dontev
* Express
* mongoose

###### Environment variables (env.file) :

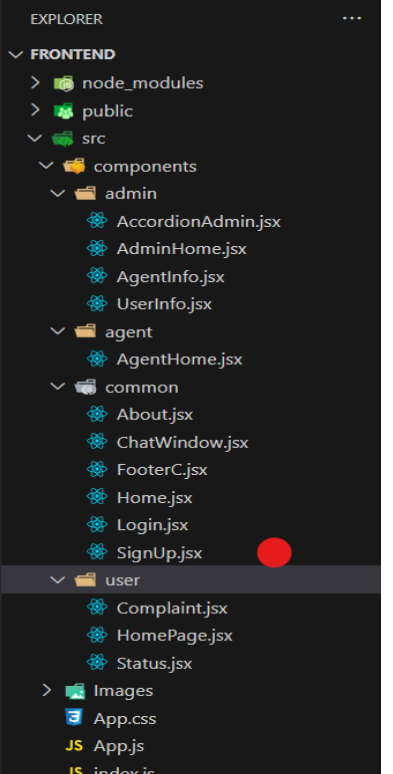
You can change the port configuration in the .env file if needed.

MONGODB\_URI=mongodb://localhost:27017/ecommerce

PORT=6001

**Folder Structure:**

Client Folder:

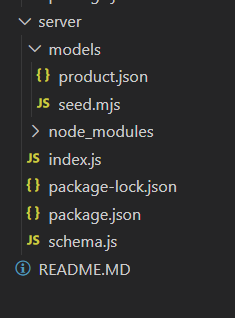


This structure assumes a React app and follows a modular approach. Here's a brief explanation of the main directories and files:

• src/components: Contains components related to the application such as, register, login, home, etc.,

• src/pages has the files for all the pages in the application.

* Server Folder:



###### Setup express server:

[Node.js](http://node.js) is a runtime environment that allows you to run javascript code on the server .

• Create index.js file.

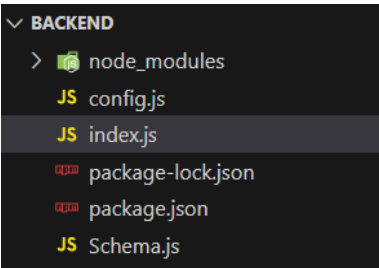
• Create an express server on your desired port number.

• Define API’s

###### 6.RUNNING THE APPLICATION:

Backend: cd server

npm start



Frontend: cd client

npm start

##### 7.API DOCUMENTATION:

##### USER API:

**Register User**

* Endpoint: /register
* Method: POST
* Body:

{

"username": "John",

"email": "john@example.com",

"usertype": "customer",

"password": "123456"

}

#### Login User:

* Endpoint: /login
* Method: POST
* Body:

{

"email": "john@example.com",

"password": "123456"

}

#### Fetch All Users:

* Endpoint: /fetch-users
* Method: GET

### Product APIs

Fetch All Products

* Endpoint: /fetch-products
* Method: GET

#### Fetch Product By ID

* Endpoint: /fetch-product-details/:id
* Method: GET

#### Add New Product

* Endpoint: /add-new-product
* Method: POST

#### Update Product

* Endpoint: /update-product/:id
* Method: PUT

### Order APIs

Place Order (Single Product)

* Endpoint: /buy-product
* Method: POST

###### Place Order (From Cart)

* Endpoint: /place-cart-order
* Method: POST

#### Fetch All Orders

* Endpoint: /fetch-orders
* Method: GET

#### Cancel Order

* Endpoint: /cancel-order
* Method: PUT

#### Update Order Status

* Endpoint: /update-order-status
* Method: PUT

### Cart APIs

#### Fetch Cart Items

* Endpoint: /fetch-cart
* Method: GET

#### Add Item to Cart

* Endpoint: /add-to-cart
* Method: POST

#### Increase Quantity

* Endpoint: /increase-cart-quantity
* Method: PUT

#### Decrease Quantity

* Endpoint: /decrease-cart-quantity
* Method: PUT

#### Remove Item

* Endpoint: /remove-item
* Method: PUT

### Admin APIs

#### Fetch Banner

* Endpoint: /fetch-banner
* Method: GET

#### Update Banner

* Endpoint: /update-banner
* Method: POST

#### Fetch Categories

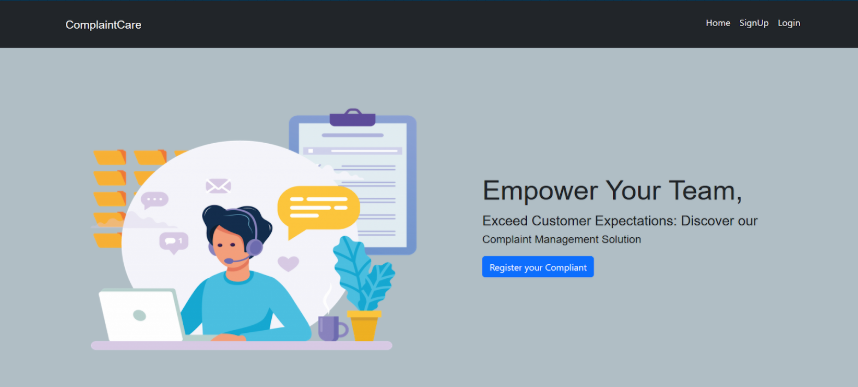
* Endpoint: /fetch-categories
* Method: GET

###### 8.AUTHENTICATION:

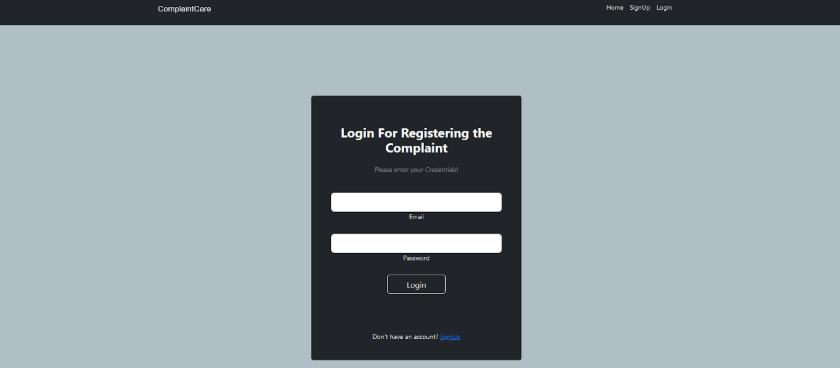
* Authentication is handled via hashed passwords using bcrypt.
* No JWT is currently used; sessions or token handling can be added in future.

**9. UI images:**

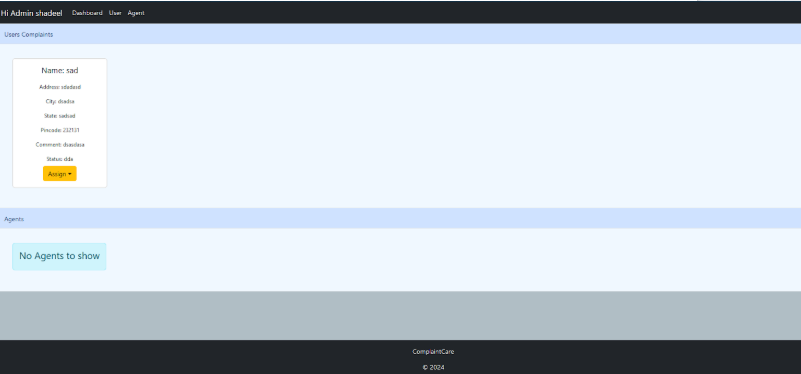
· **Landing page**



**Authentication**



·**UserProfile**



**CONCLUSION:**

The Online Complaint Registration & Management System is more than just a ticketing platform—it empowers organizations to proactively transform customer dissatisfaction into valuable insights and improvements. By offering a secure, centralized, and standardized framework for submitting, routing, communicating, and analyzing complaints, the system helps:

* Reduce reputational and regulatory risk, while increasing customer retention through attentive, timely service
* Drive operational efficiency with automated workflows, intelligent routing, and escalation mechanisms that ensure consistent resolution practices .
* Uncover root causes and system gaps, feeding analytics backinto product improvements, QA processes, and compliance management .

###### 10.KNOWN ISSUES:

| **Issue** | **Impact** |
| --- | --- |
| Lack of standardized procedure | Inconsistent handling and resolution delays |
| Incomplete/inaccurate data | Poor investigations and user frustration |
| Delayed resolutions | Missed SLAs and lowered customer trust |
| Integration limitations | Data silos and inefficient workflows |
| Training/adoption gaps | Underutilization and misuse of the system |

##### 

**11.FUTURE ENHANCEMENT**:

| **Future Enhancement** | **Benefit** |
| --- | --- |
| AI-driven classification & sentiment | Faster, accurate sorting and prioritization |
| Predictive analytics | Proactive resolution, earlier intervention |
| Omni‑channel self‑service | Consistent support across touchpoints |
| Modular Agentic AI | Scalable, flexible AI ecosystem |
| Multimodal intake (voice, image) | Richer complaint context, faster workflows |
| Blockchain auditability | Regulatory compliance and trust |
| Personalized handling | Enhanced user satisfaction and fairness |