# Full Stack Development with MERN

## Project Documentation

### 1. Introduction

\*\*Project Title:\*\*E-Commerce Website Development

#### \*\*Team Members:\*\*

List all team members and their specific roles in the project. For instance:

Danush R

Vijay E

Ramprabakar J

Sarathy A

Ranjith S

### 2. Project Overview

#### \*\*Purpose:\*\*

The goal of ShopEZ is to create a comprehensive e-commerce platform that serves both buyers and sellers efficiently. It simplifies the shopping experience by providing intuitive navigation, real-time product updates, and secure transactions. For sellers, the application offers a feature-rich dashboard to manage products, orders, and gain insights into business performance.

#### \*\*Features:\*\*

- \*\*Effortless product discovery:\*\* Advanced search and filtering capabilities ensure users find exactly what they need.

- \*\*Detailed product information:\*\* Includes product specifications, images, and customer reviews.

- \*\*Secure checkout:\*\* Implements encryption and secure payment gateways for safe transactions.

- \*\*Seller dashboard:\*\* Offers sales analytics, order tracking, and inventory management tools.

### 3. Architecture

#### \*\*Frontend:\*\*

Developed using React.js for dynamic and responsive UI/UX. The structure includes reusable components, state management with Redux, and integration with backend APIs.

#### \*\*Backend:\*\*

Built with Node.js and Express.js to handle API requests, authentication, and business logic. The backend ensures scalability and supports RESTful endpoints.

#### \*\*Database:\*\*

MongoDB is employed for storing user profiles, product details, order histories, and other application data. The schema is optimized for fast query execution and data relationships.

### 4. Setup Instructions

#### \*\*Prerequisites:\*\*

- Install Node.js (v14 or later).

- Ensure MongoDB is installed and running on your machine.

- Install a package manager like npm or Yarn.

#### \*\*Installation Steps:\*\*

1. Clone the repository using the command: `git clone [https://github.com/ranjith933/NM-MERN-STACK-DEVELOPMENT-E-COMMERCE-WEBSITE.git]`.

2. Navigate to the project folder: `cd ShopEZ`.

3. Install dependencies for the frontend and backend:

- Frontend: `cd client && npm install`

- Backend: `cd server && npm install`

4. Create `.env` files for both frontend and backend, adding the required environment variables (e.g., database URIs, API keys).

5. Start the application using the provided commands in the 'Running the Application' section.

### 5. Folder Structure

#### \*\*Client Folder:\*\*

The React-based frontend includes folders for components, pages, services, and assets. Example:

- `components/`: Contains reusable UI elements like headers, footers, and buttons.

- `pages/`: Holds pages like Home, Product Details, and Cart.

#### \*\*Server Folder:\*\*

The Node.js backend has well-defined directories for maintaining modularity and readability. Example:

- `routes/`: Defines API routes like `/products`, `/orders`, and `/users`.

- `models/`: Contains MongoDB schemas for products, users, and orders.

- `controllers/`: Implements business logic for handling API requests.

### 6. Running the Application

Follow these commands to run the application locally:

- \*\*Frontend:\*\*

```bash

cd client

npm start

```

- \*\*Backend:\*\*

```bash

cd server

npm start

```

- Open `http://localhost:3000` in your browser to view the application.

### 7. API Documentation

ShopEZ offers a range of RESTful APIs to facilitate communication between the frontend and backend. Examples include:

- \*\*GET /products:\*\* Retrieves a list of all products.

- Parameters: Optional filters like category or price range.

- Response: JSON array of product objects.

- \*\*POST /orders:\*\* Creates a new order.

- Body: Includes user ID, product IDs, and payment details.

- Response: Confirmation message with order ID.

- \*\*PUT /users/:id:\*\* Updates user information.

- Response: Updated user object.

### 8. Authentication

Authentication is implemented using JSON Web Tokens (JWT). Key features include:

- \*\*User roles:\*\* Separate access controls for buyers and sellers.

- \*\*Session management:\*\* Tokens are securely stored in HTTP-only cookies.

- \*\*Authorization:\*\* Ensures secure access to sensitive endpoints like `/orders` and `/dashboard`.

### 9. User Interface

The UI prioritizes ease of use and accessibility. Features include:

- A clean and modern homepage with product highlights.

- Responsive design ensuring optimal performance on mobile and desktop devices.

- Interactive elements like carousels, modals, and form validations.

### 10. Testing

The testing process includes:

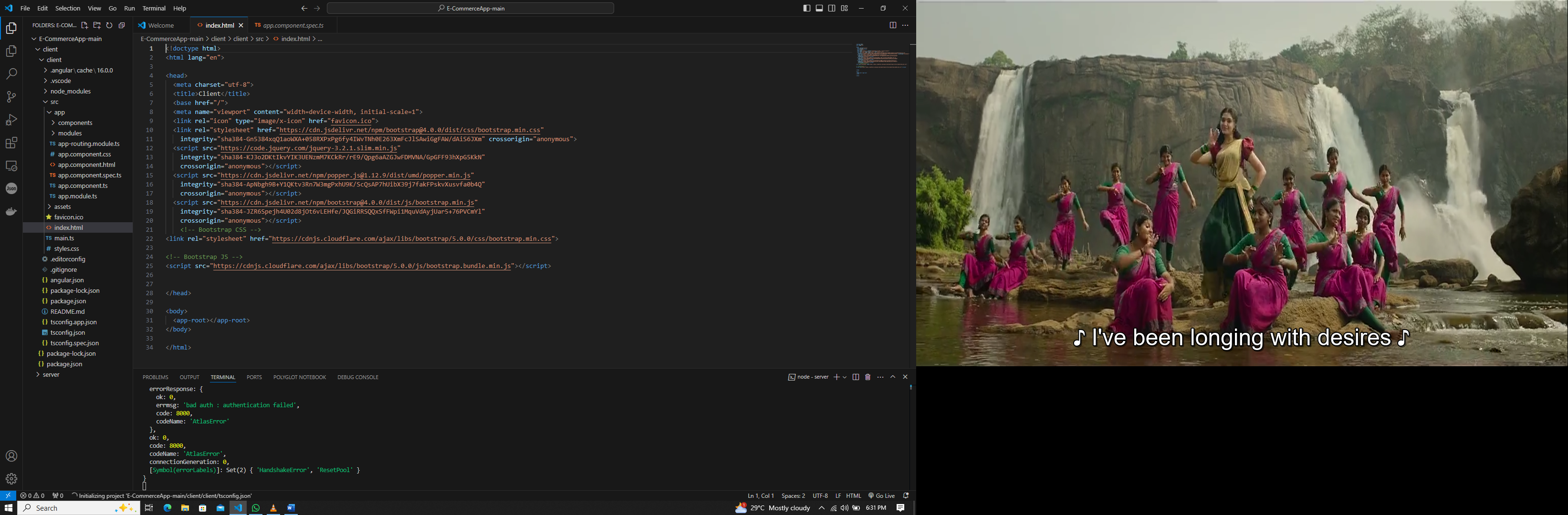
- \*\*Unit testing:\*\* Using Jest to test individual components and backend functions.

- \*\*Integration testing:\*\* Ensuring seamless interaction between frontend and backend.

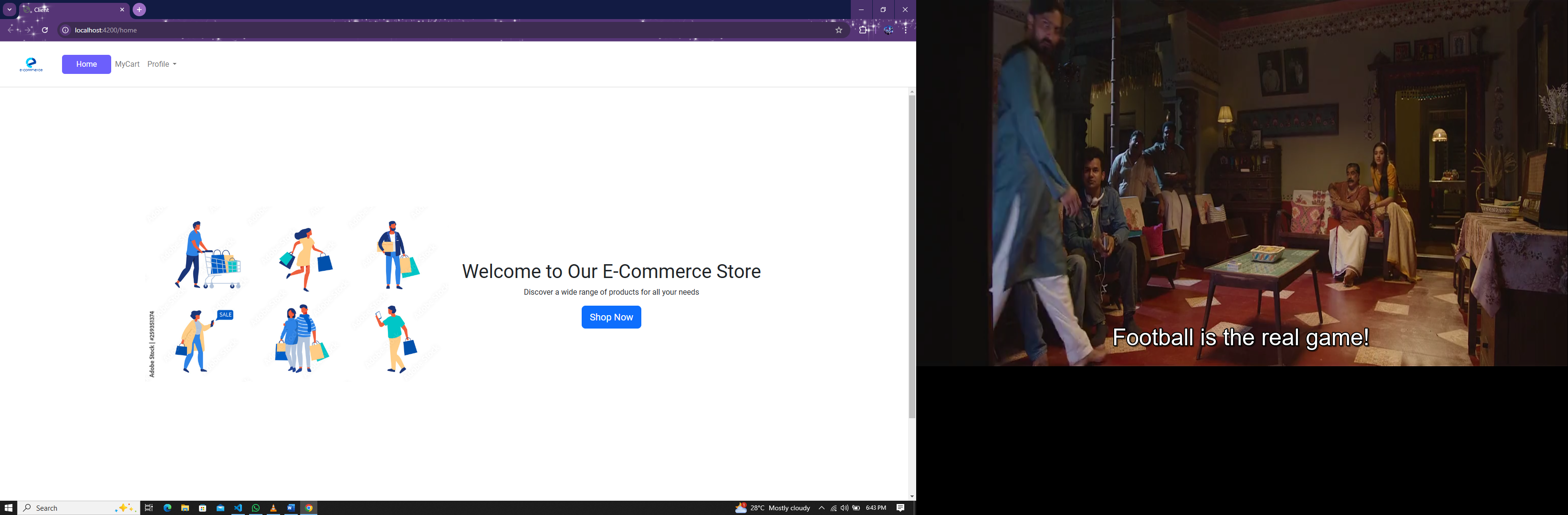
- \*\*Manual testing:\*\* Verifying core functionalities like checkout, login, and search.

### 11. Screenshots or Demo

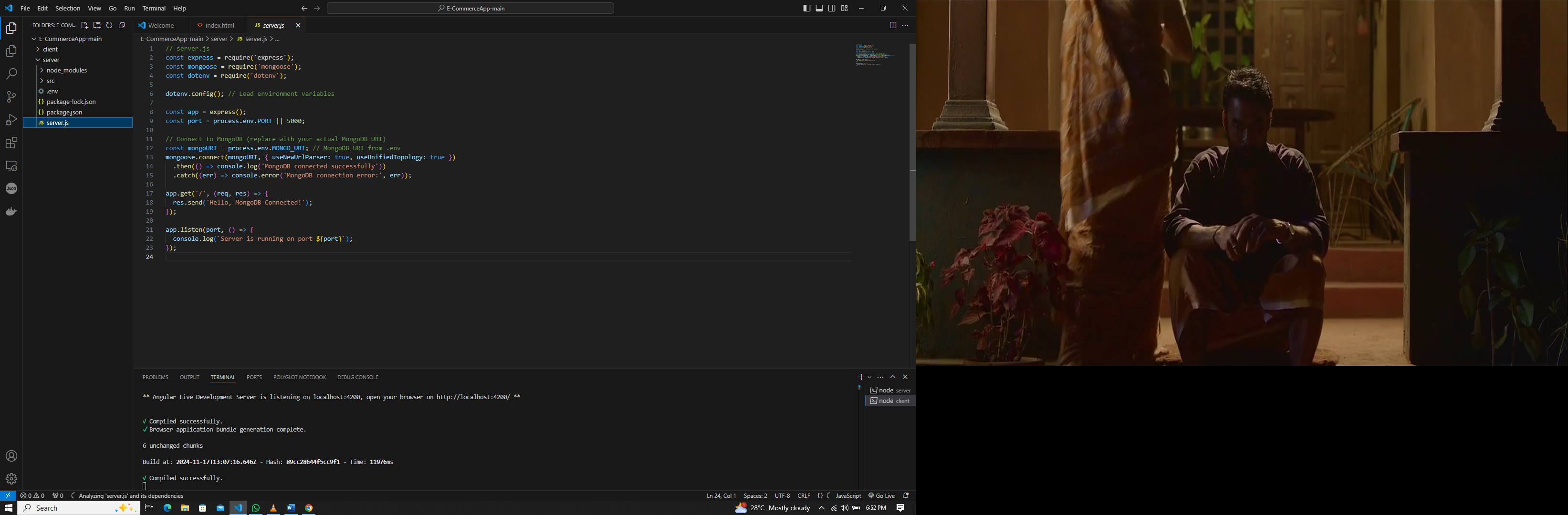
Frontend(client)



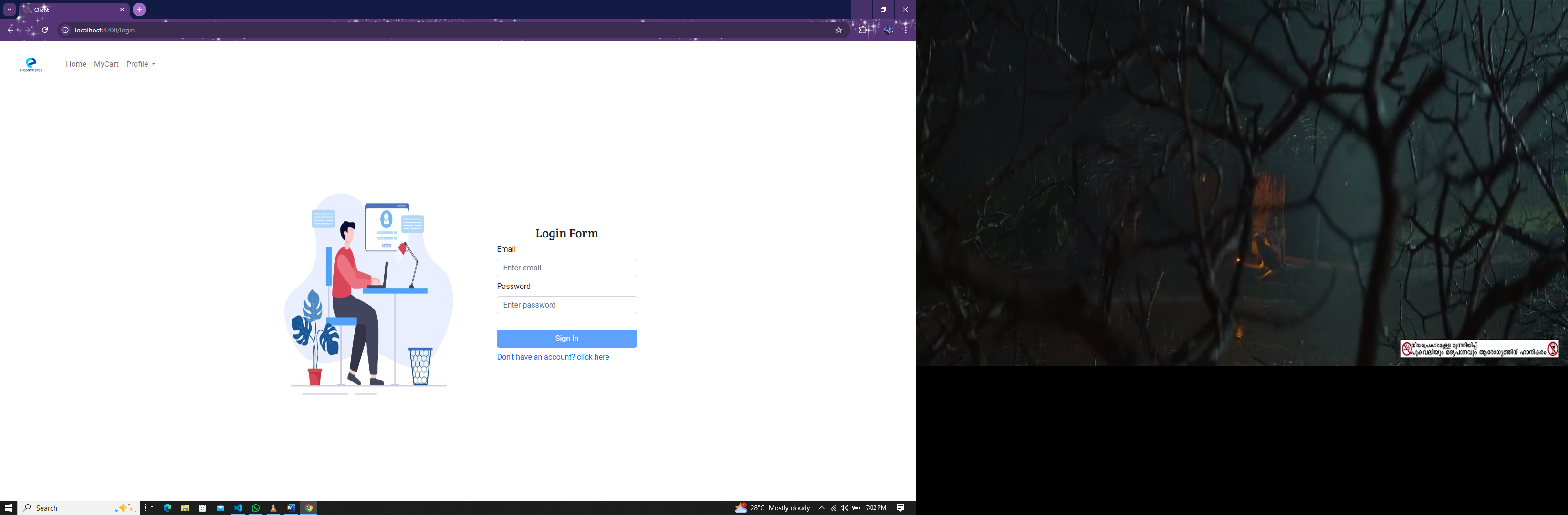
IN WEB

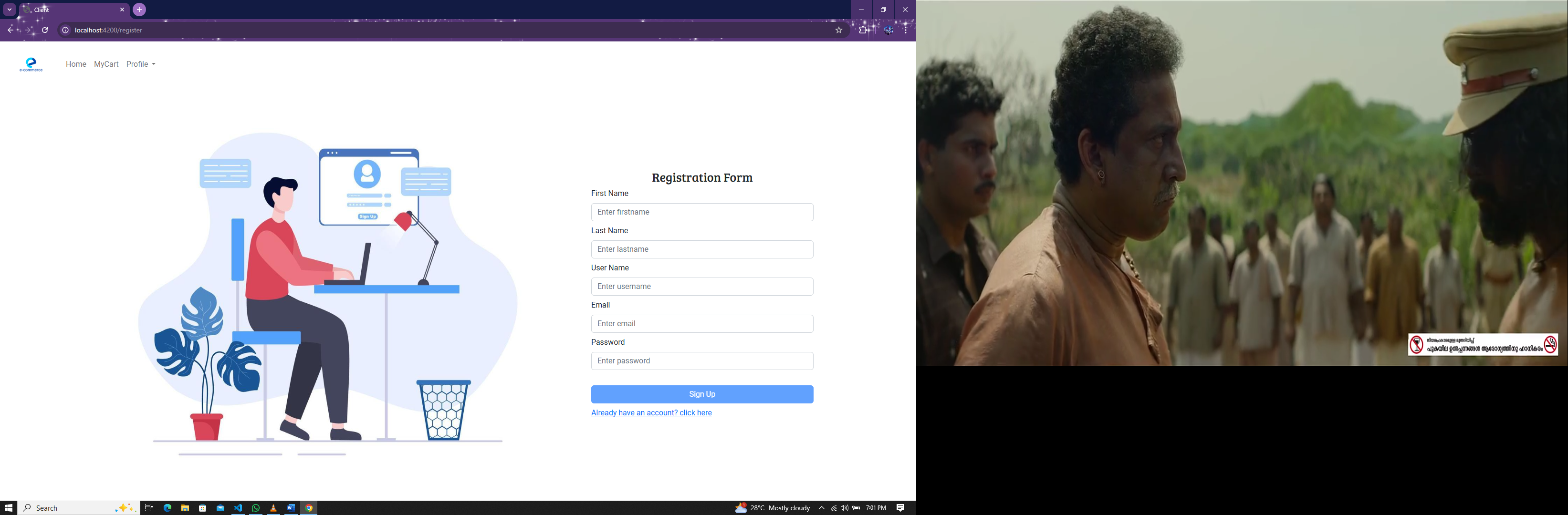


Backend(server)



LOGIN PAGE





### 12. Known Issues

- Occasional latency in API responses under heavy load.

- Limited error handling for certain edge cases.

### 13. Future Enhancements

- Integration with AI-based product recommendations.

- Real-time inventory updates using WebSocket.

- Multi-language support for a global user base.