**Freelancing Application MERN**

**Team Members:**

* Jayaprakash & Adhitya - UI/UX Designer

Focuses on designing an intuitive and user-friendly interface for the application.

Responsible for creating wireframes, mockups, and implementing design elements using tools like Sass and React for a polished user experience

* Adhitya & Sharmesh- Frontend Developer

Handles the development of the Vite with React frontend using yarn, including creating UI components, implementing user flows, and ensuring responsiveness.

Collaborates with the backend team to integrate API endpoints and data fetching.

* Lokesh - Project Manager & Backend Developer

Responsible for overseeing the project development, coordinating with team members, and ensuring project milestones are met.

Leads backend development, including setting up the server, creating API endpoints, and integrating with MongoDB, JWT, & Postman

* Kashvi - Backend Developer & Quality Assurance & Tester

Ensures the application meets quality standards by creating and executing test cases for both frontend and backend.

Responsible for identifying bugs, documenting issues, and conducting final testing before deployment.

**Architecture:**

* **Frontend:** Built with Vite and React, it provides the user interface, manages state, and communicates with the backend via authenticated API calls.
* **Backend:** Node.js with Express.js manages API routes, JWT-based authentication, and business logic. Middleware handles token verification.
* **Database:** MongoDB stores application data, with Mongoose simplifying database interaction and schema validation.

**JWT Middleware Architecture:**

* **Middleware:** Middleware in the backend (e.g., Node.js with Express.js) intercepts API requests, verifies JSON Web Tokens (JWTs) in headers, and enforces authentication/authorization rules.
* **JWT:** The backend generates and validates secure JWTs containing user claims (like user ID and roles), signed using a secret or public/private key pair.
* **Testing:** Postman tests API endpoints by including the JWT in headers (e.g., Authorization: Bearer <token>) to verify middleware behavior and backend logic.

**Setup Instructions:**

* Prerequisites: Before setting up the project, make sure you have the following software dependencies installed:
* Node.js (v14 or higher)
* Required for running the backend server (Node and Express).
* Download from [Node.js official website](https://nodejs.org/).
* npm (Node Package Manager)
* Comes with Node.js and is required to install project dependencies.
* To verify, run: node -v and npm -v in your terminal.
* MongoDB (Community Edition or MongoDB Atlas for cloud)
* Used for database management to store complaints, user data, and more.
* For local setup, download from [MongoDB official website](https://www.mongodb.com/try/download/community).
* Alternatively, set up a MongoDB Atlas account for a cloud database [here](https://www.mongodb.com/cloud/atlas).
* Git
* Required to clone the project repository from a version control platform (e.g., GitHub).
* Download from [Git official website](https://git-scm.com/).
* Installation: Follow these steps to set up the project locally:
* Clone the Project Repository
* Open your terminal and navigate to the directory where you want to clone the project. Then run:
* git clone < https://github.com/kashvi456/Freelancing-Application-MERN.git>
* Navigate to Project Directory
* After cloning, navigate to the project directory:
* cd Freelancing-Application-MERN
* Install Dependencies
* There are dependencies for both frontend and backend, so follow these steps:
* Frontend: Navigate to the frontend folder and install dependencies.
* cd frontend
* yarn add
* Backend: Open a new terminal, navigate to the backend folder, and install dependencies.
* cd ../backend
* yarn add
* Set Up Environment Variables
* Create .env files in both the *frontend* and *backend* folders to configure environment-specific variables. Here’s how:
* Backend Environment Variables: In the backend folder, create a .env file and add the following variables:
* PORT=8001 # Port for backend server
* Running the Project
* Once the setup is complete, start the project by running the frontend and backend servers.
* Run the Backend Server:
* In the backend folder, run:
* node server.js
* The backend server should start on http://localhost:8001.
* Run the Frontend Server:
* Open a new terminal, navigate to the frontend folder, and run:
* yarn run dev
* The frontend server should start on http://localhost:1573 or another available port.

**API Documentation:**

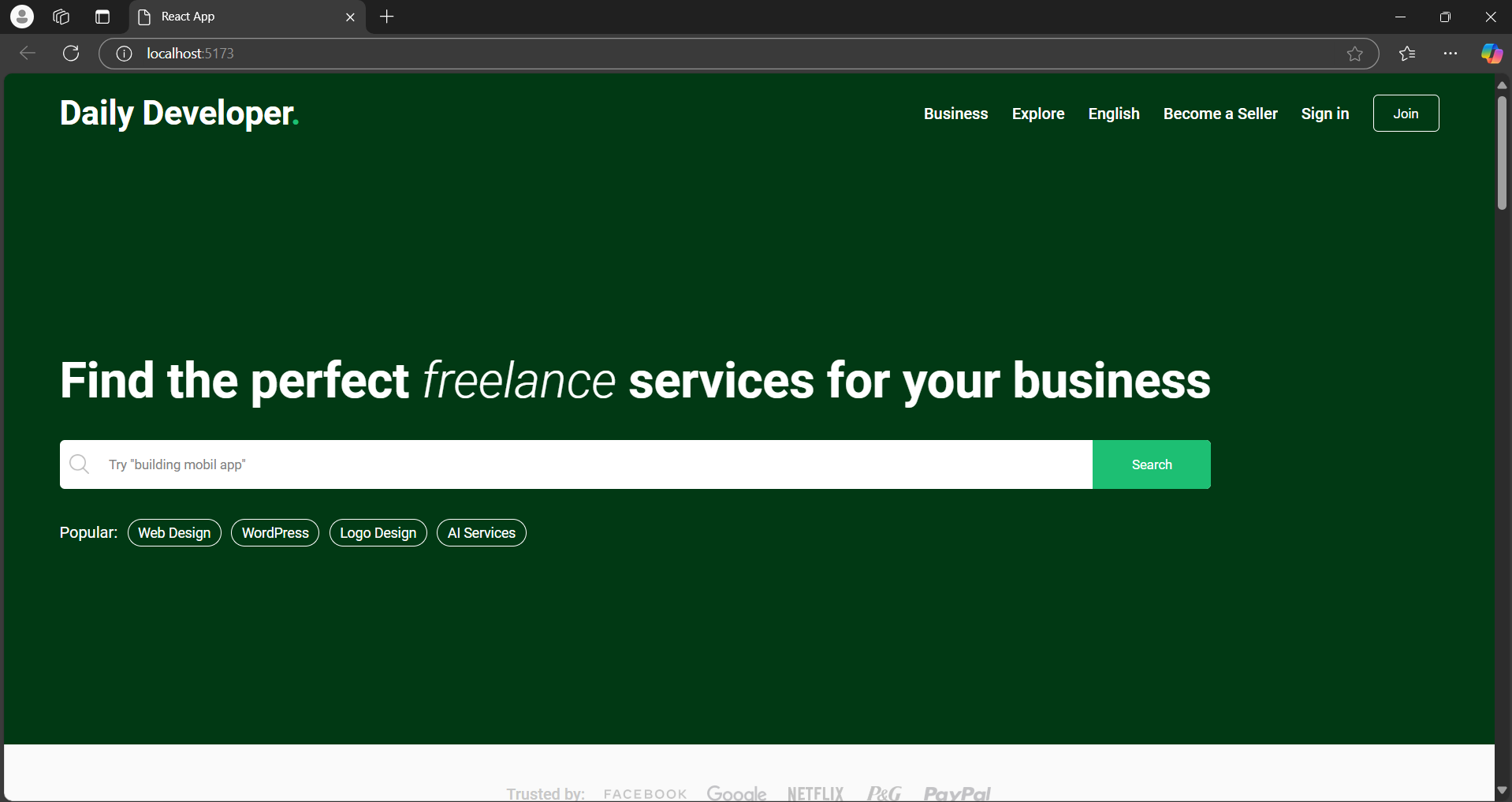
* Authentication: POST /api/auth/register, POST /api/auth/login for user registration and login.
* Complaints: CRUD operations on complaints, including submitting, retrieving, updating, and listing.
* User Profiles: Fetch and update user profile information.Include request methods, parameters, and example responses.

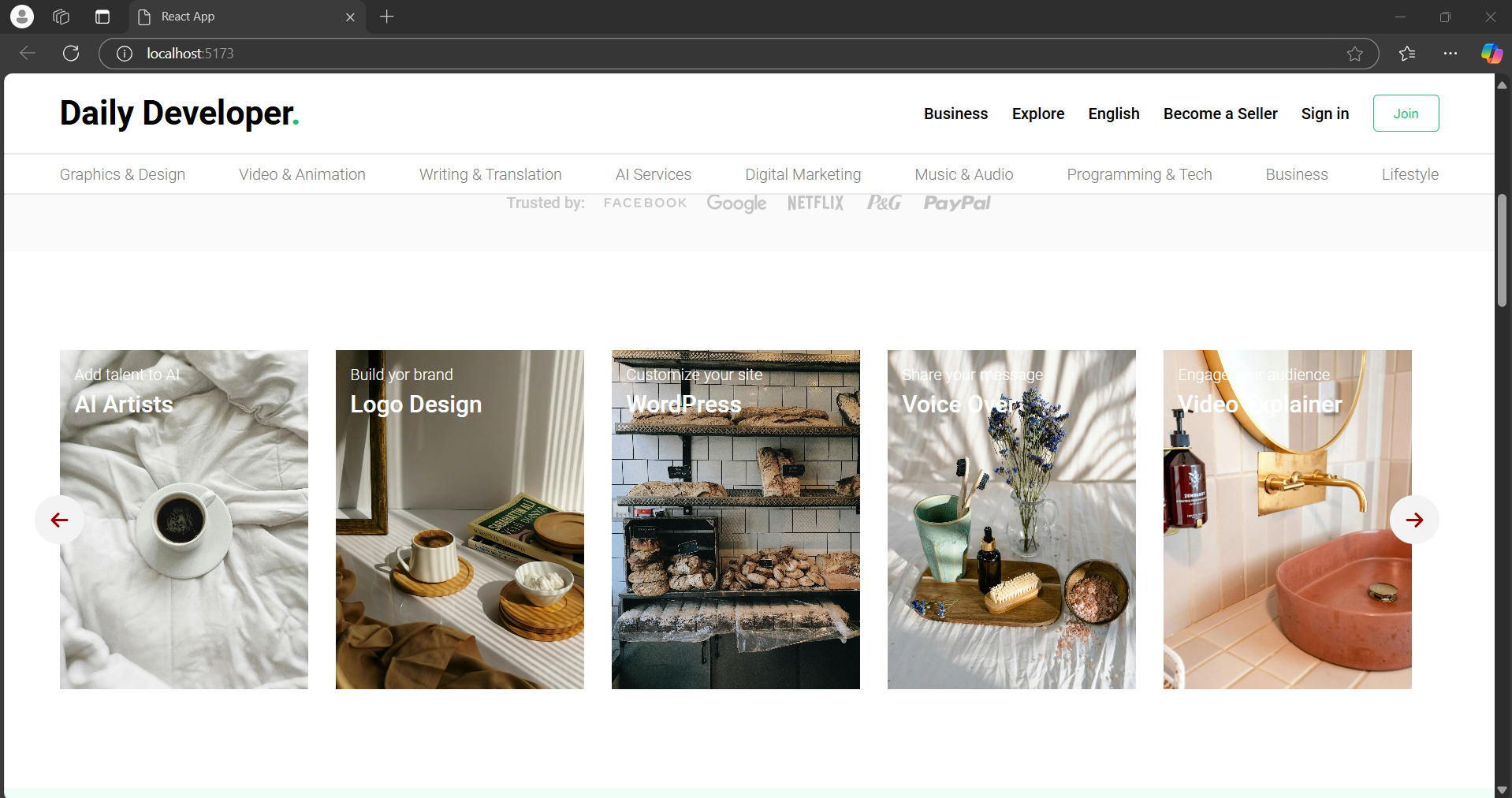
**Authentication:**

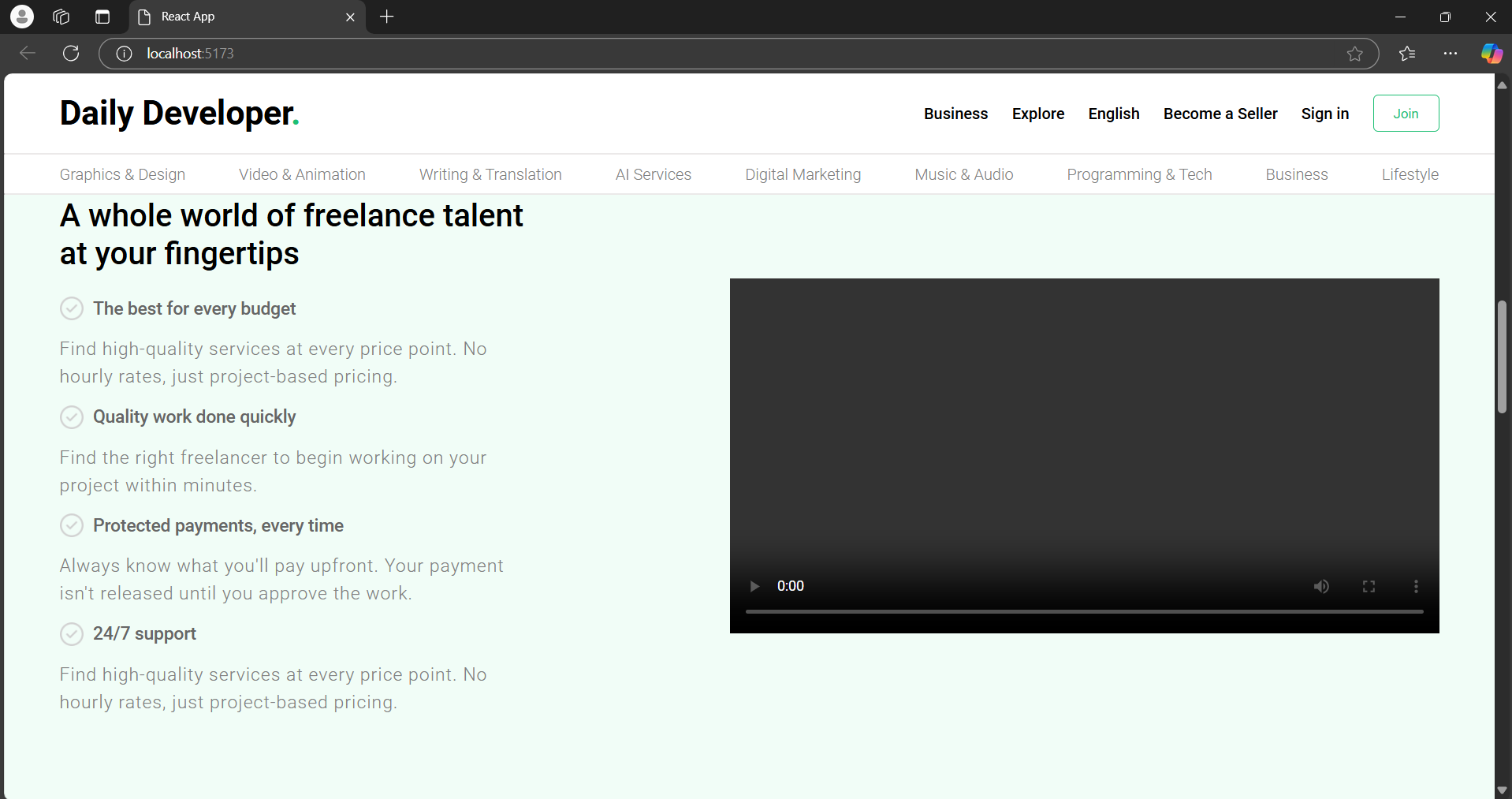
* Authentication: Handled using JWTs, where each authenticated user receives a token after logging in, which they include in the headers of subsequent requests.
* Authorization: Controlled by role-based access, enforced through middleware to restrict access to certain routes and actions based on the user's role.

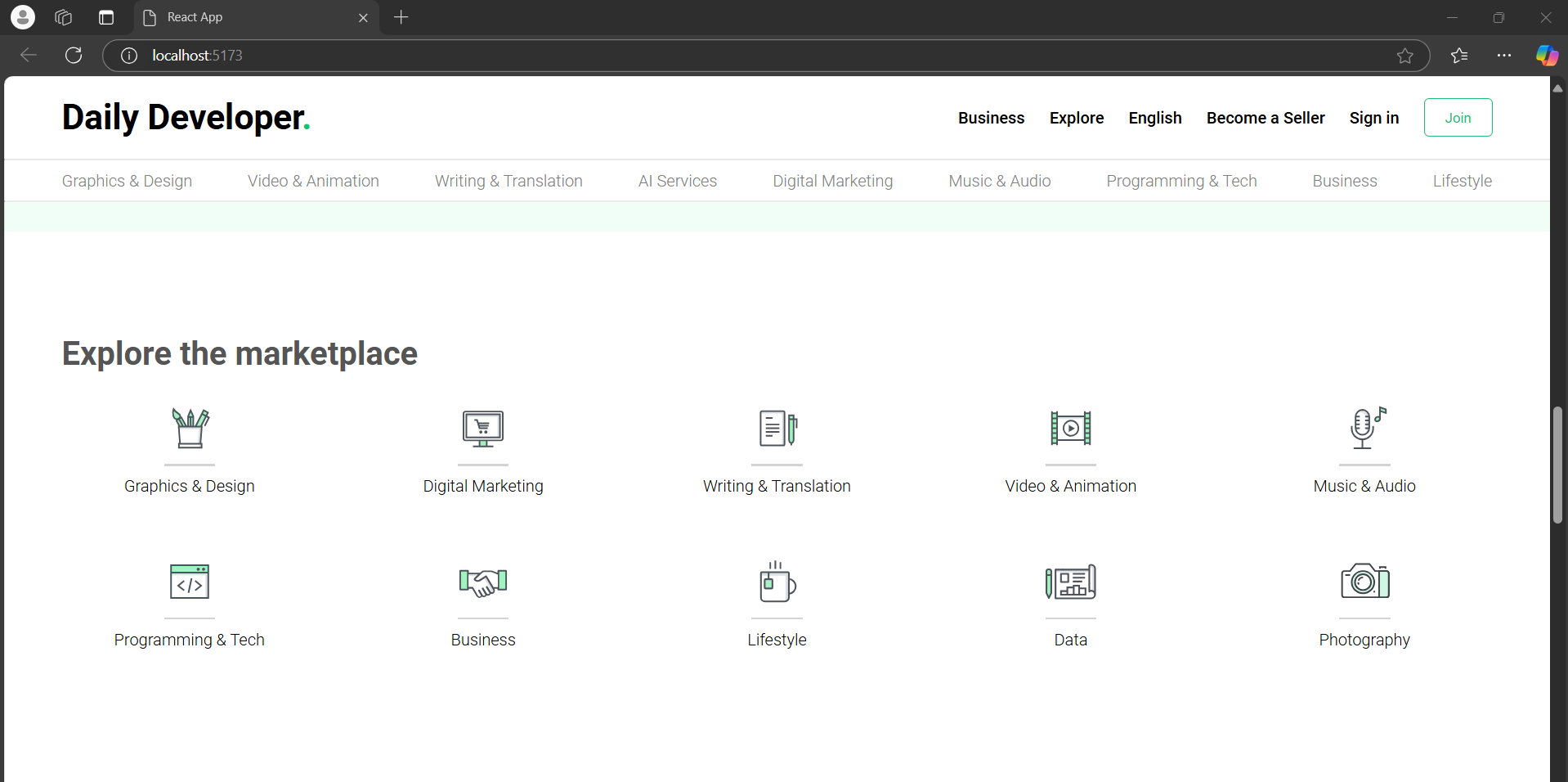
This combination of JWT-based authentication and role-based authorization ensures that only verified users have access to the application’s resources, and their permissions are aligned with their roles. This setup improves security and keeps sensitive data protected within the application.

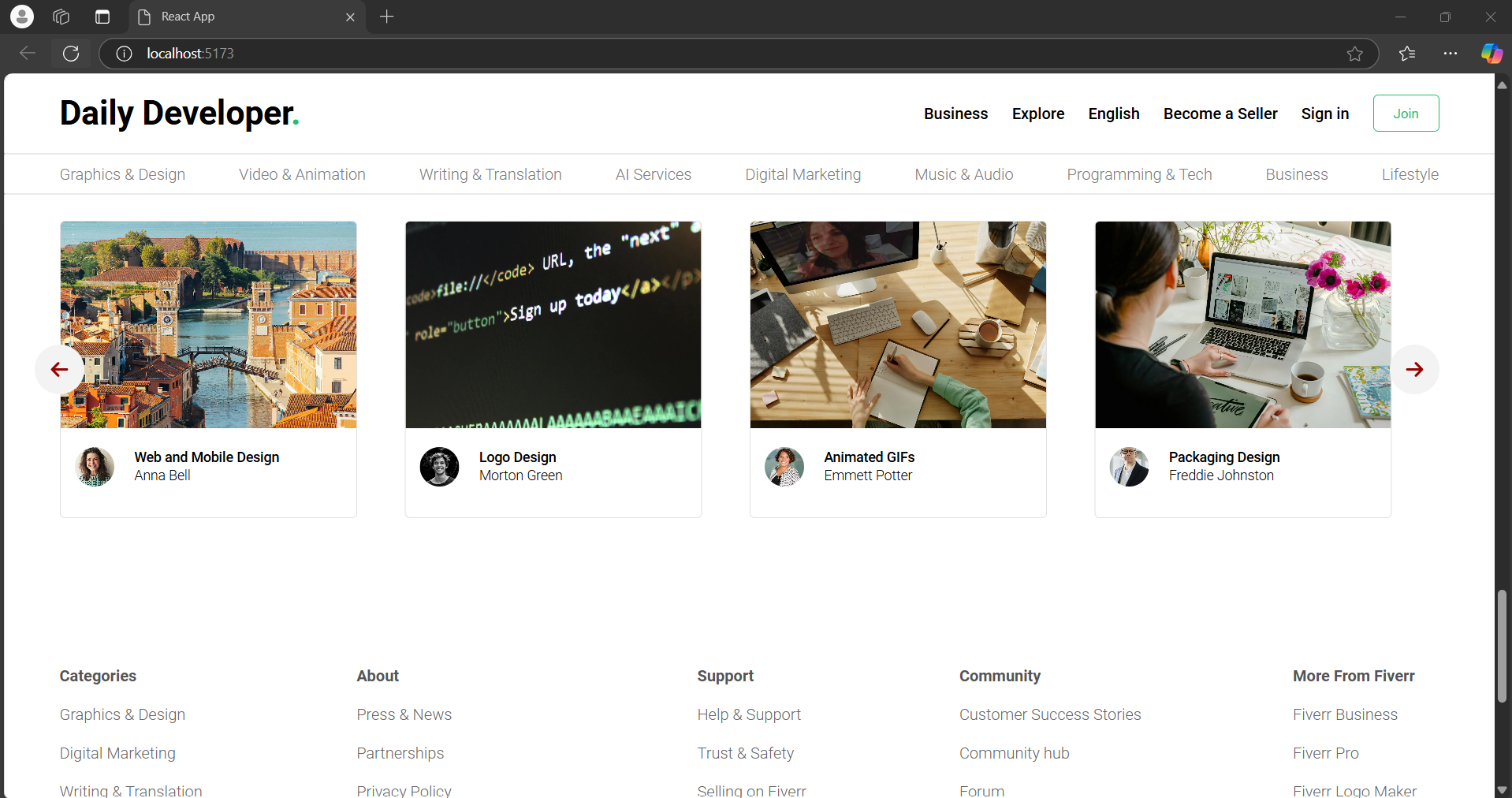
**User Interface Screenshots or Demo:**

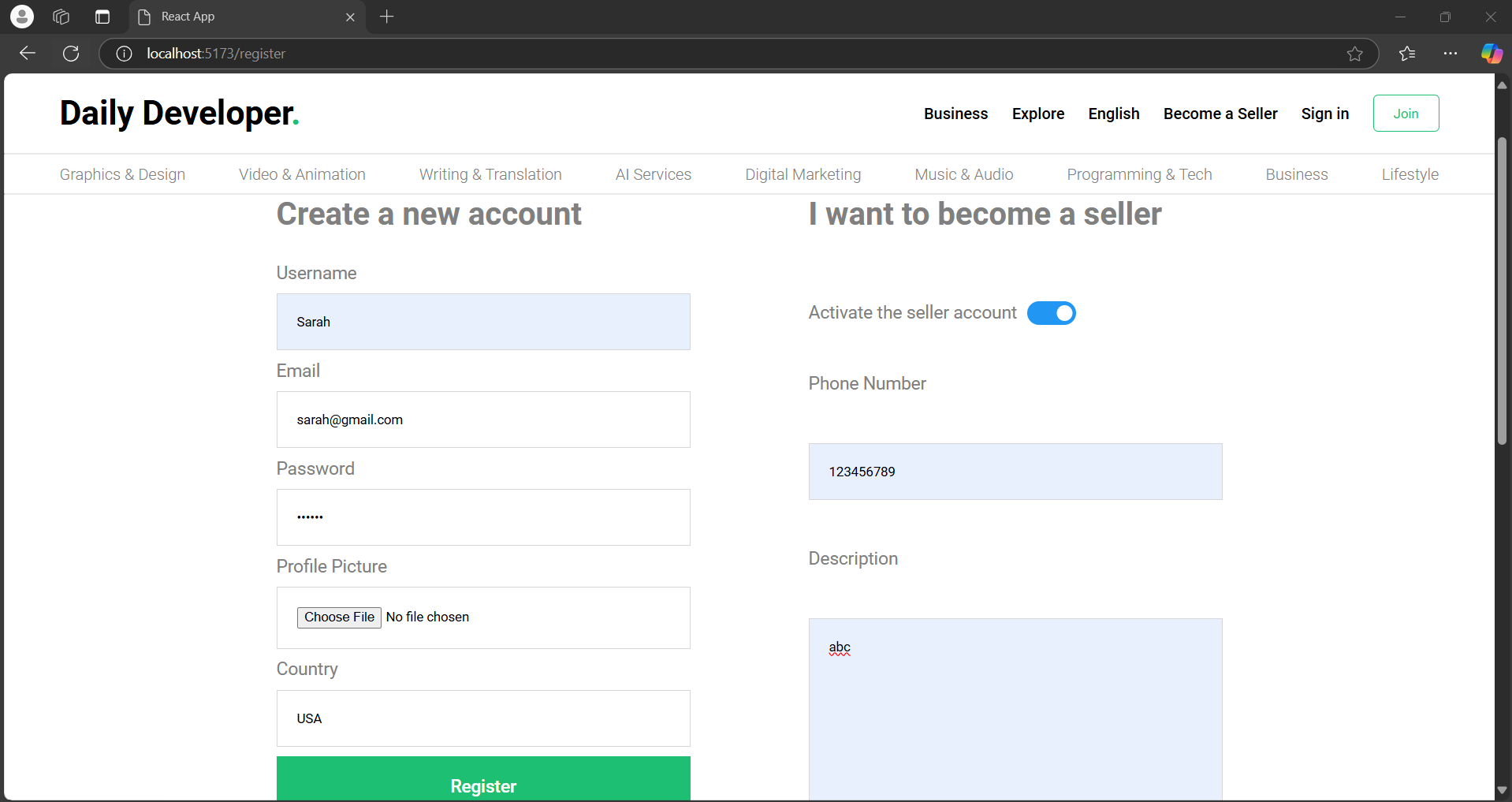


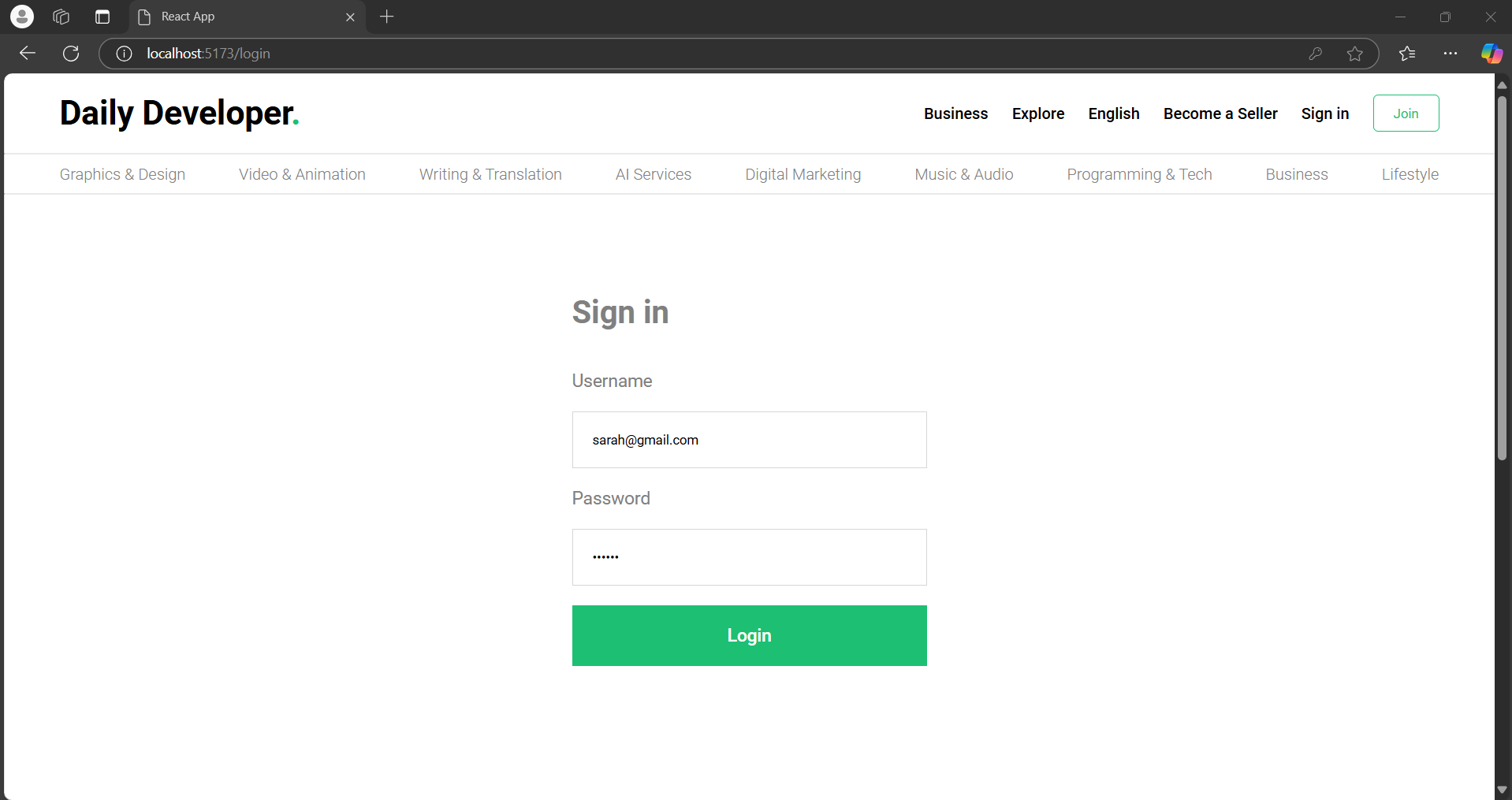


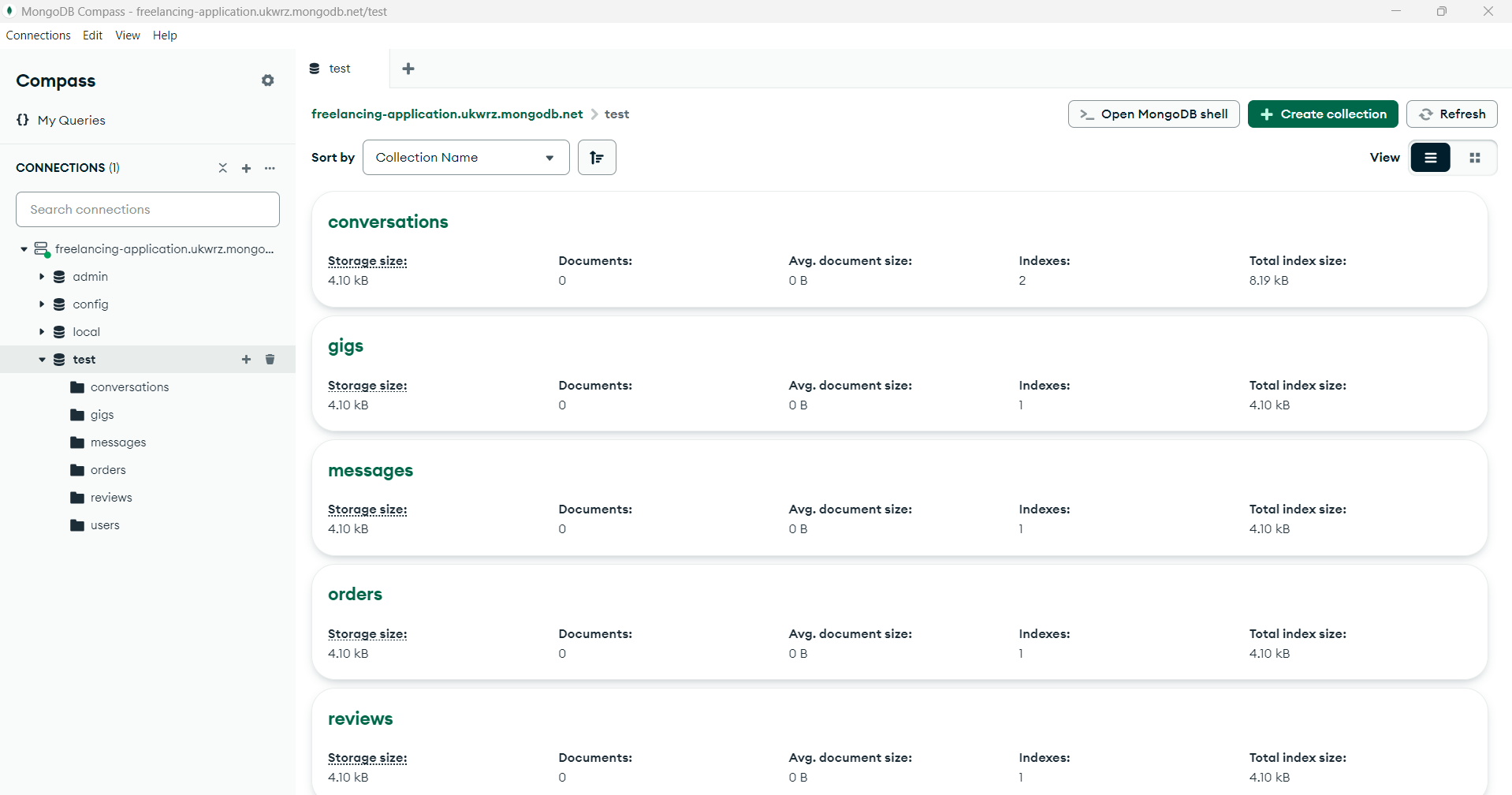


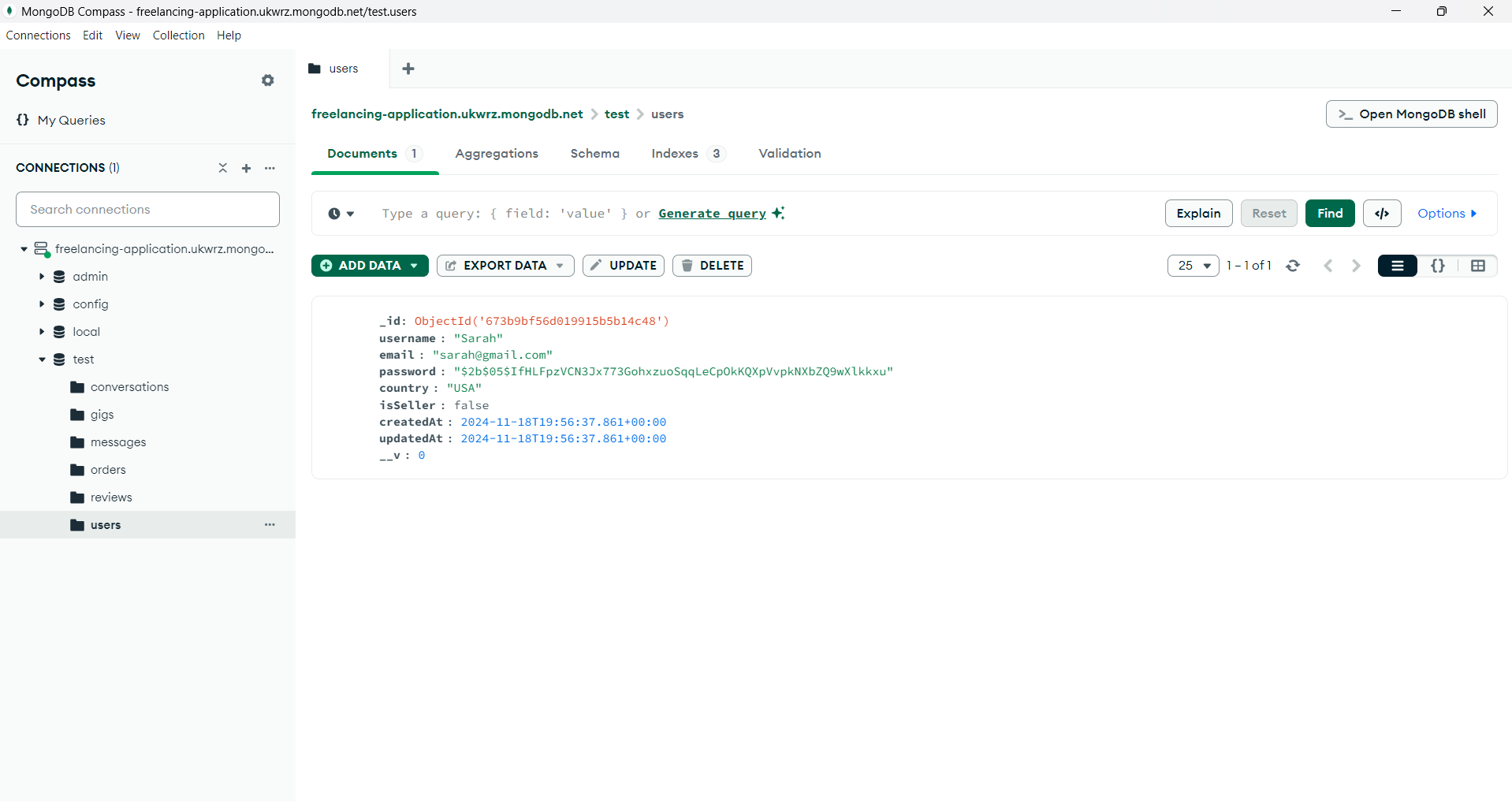


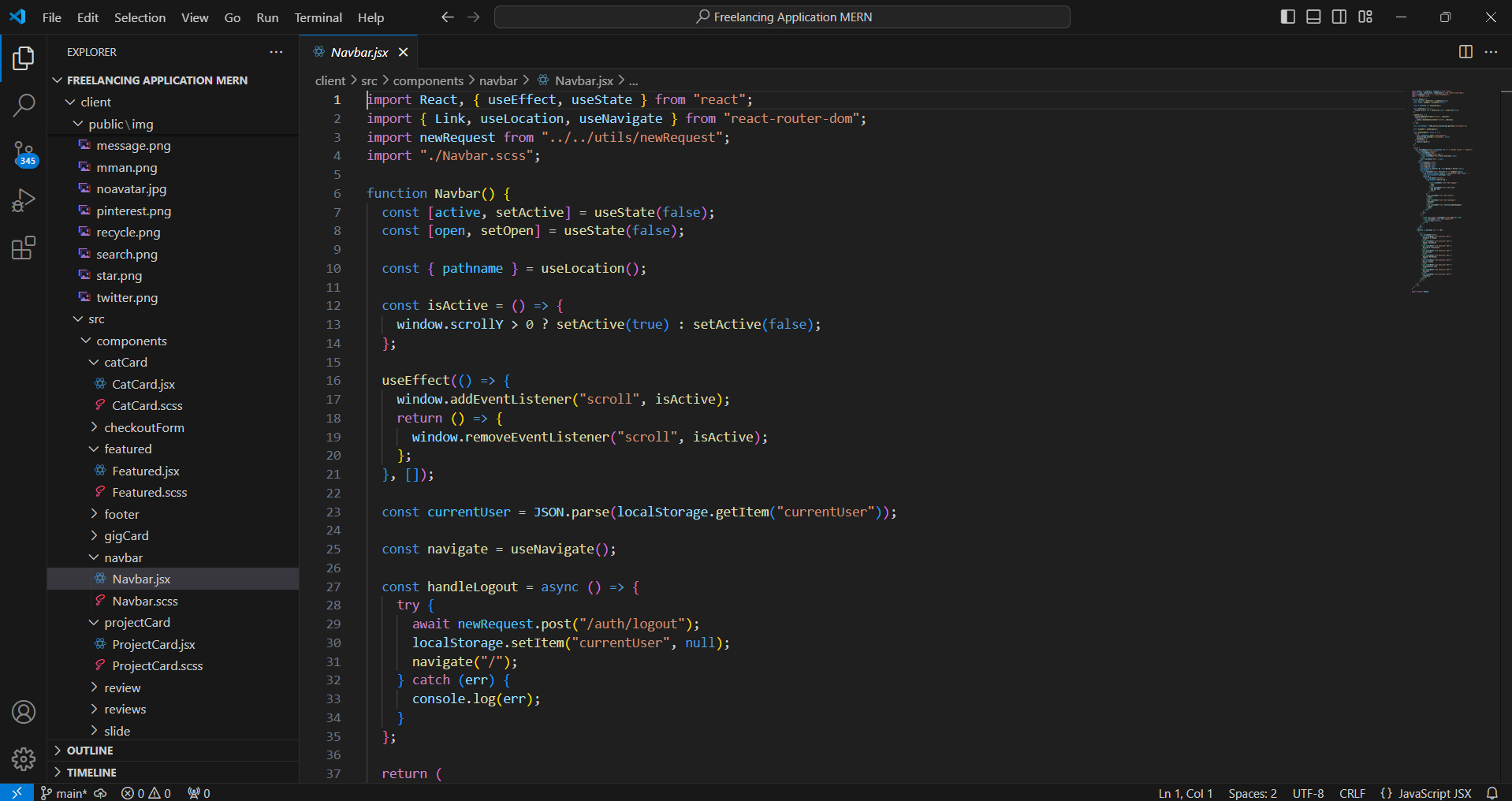


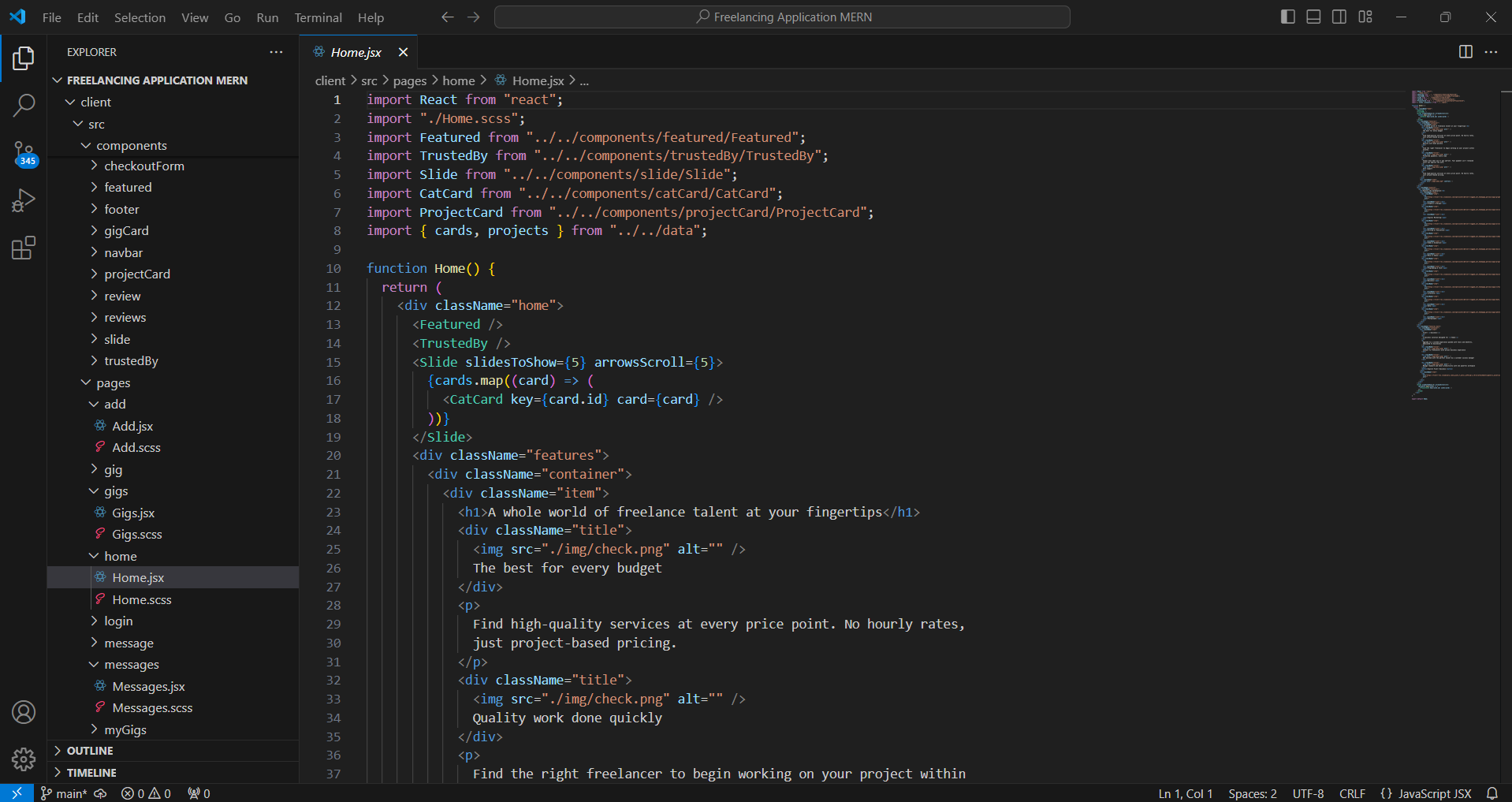


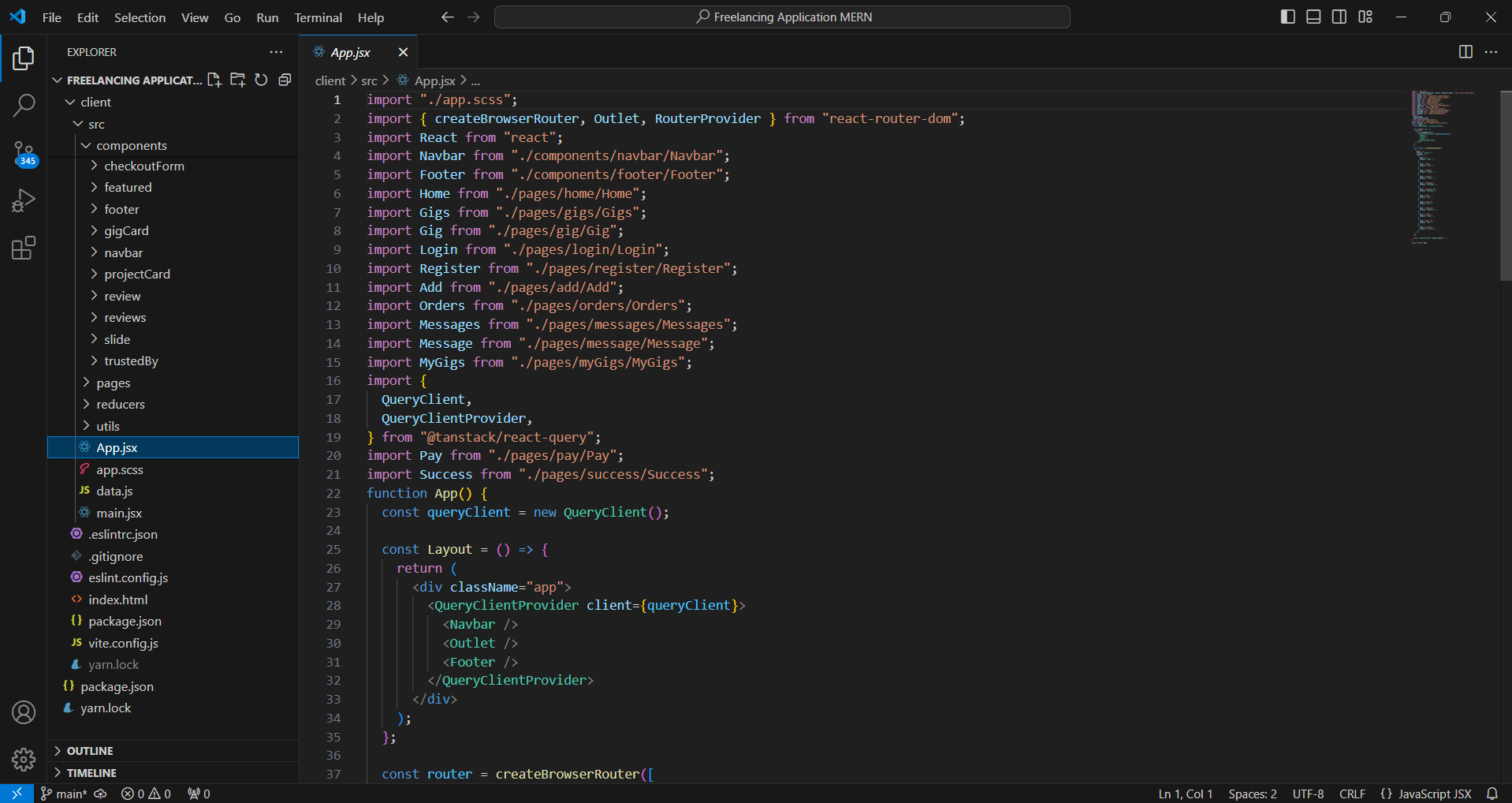


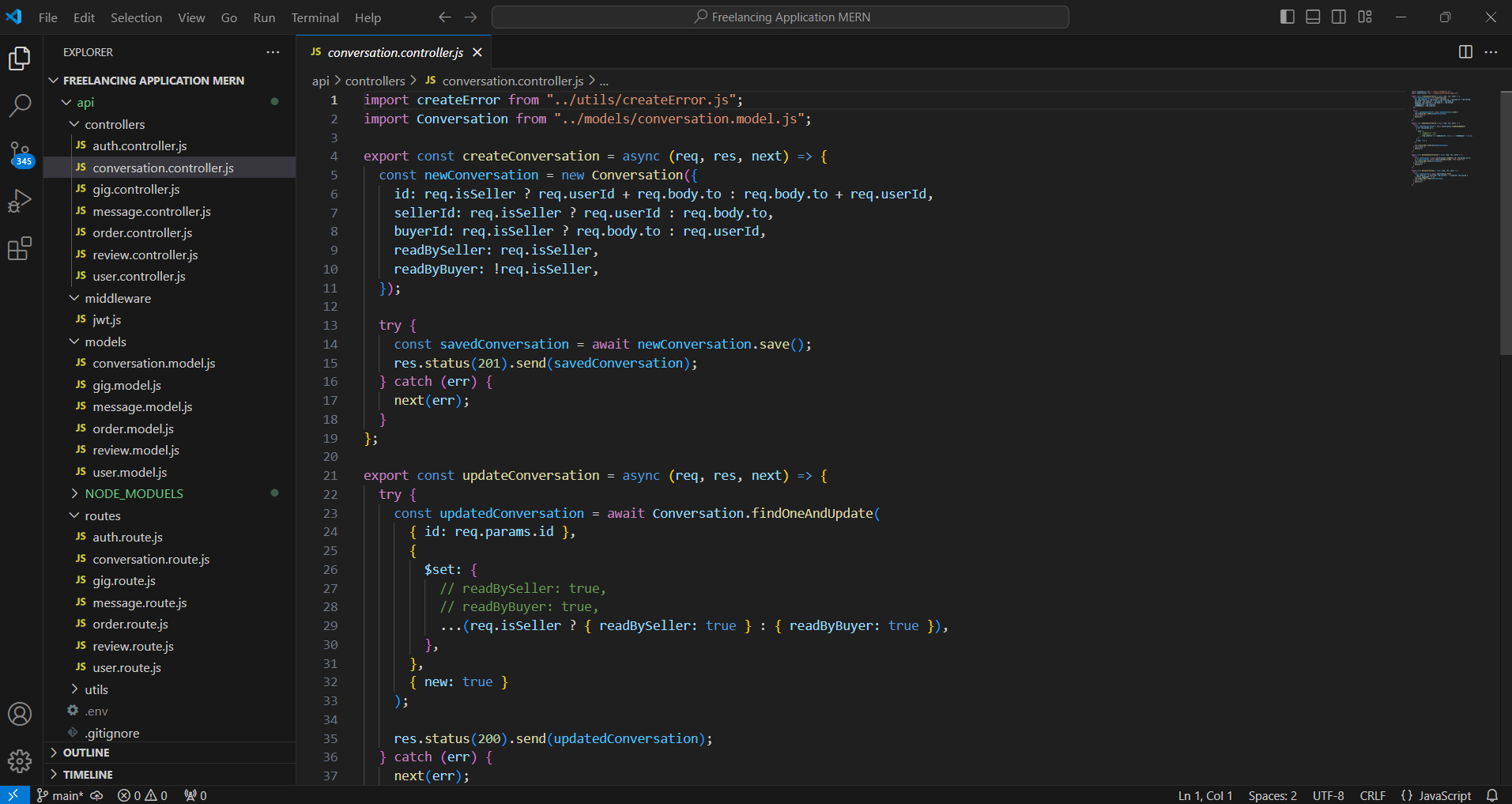


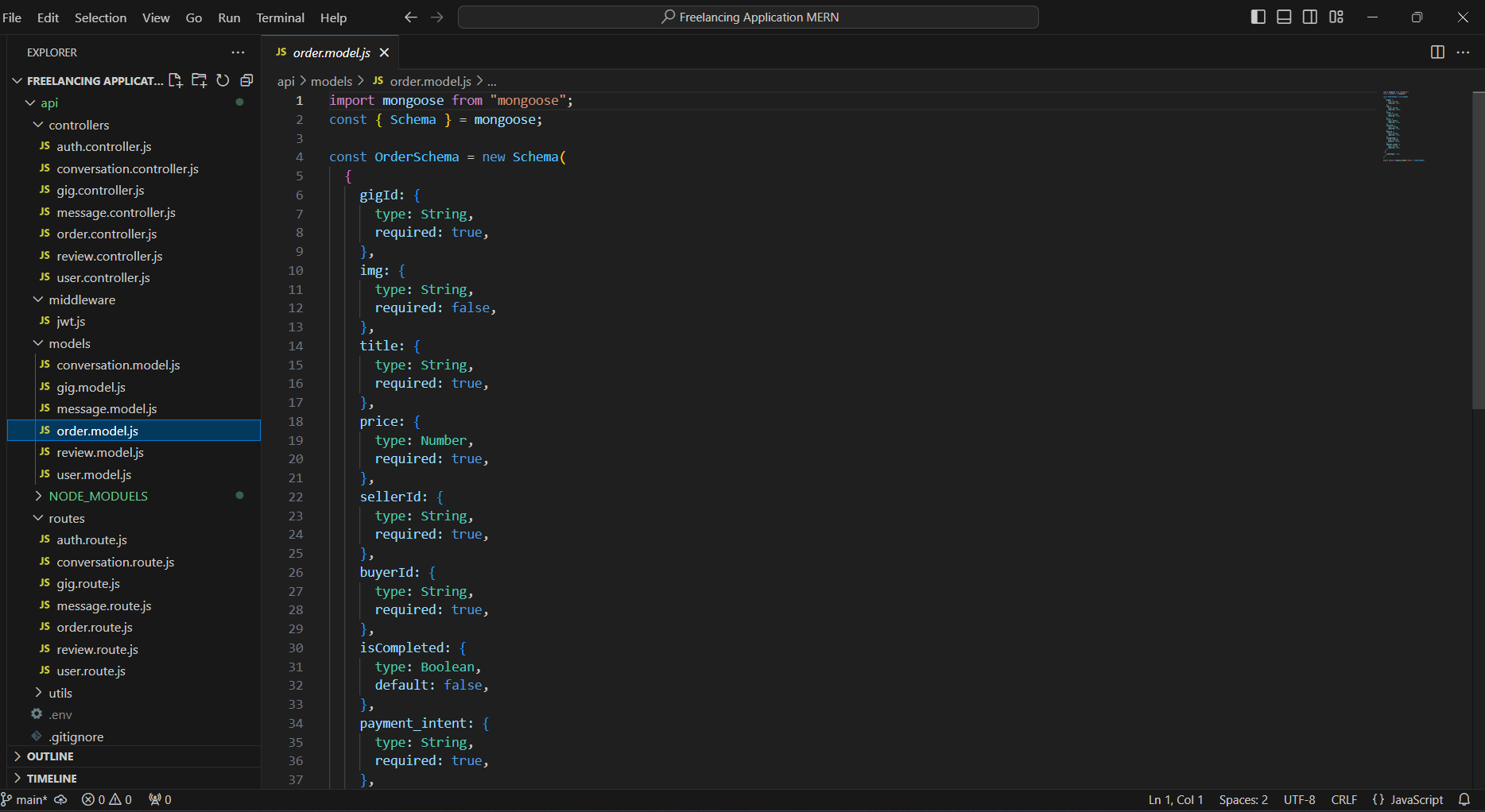


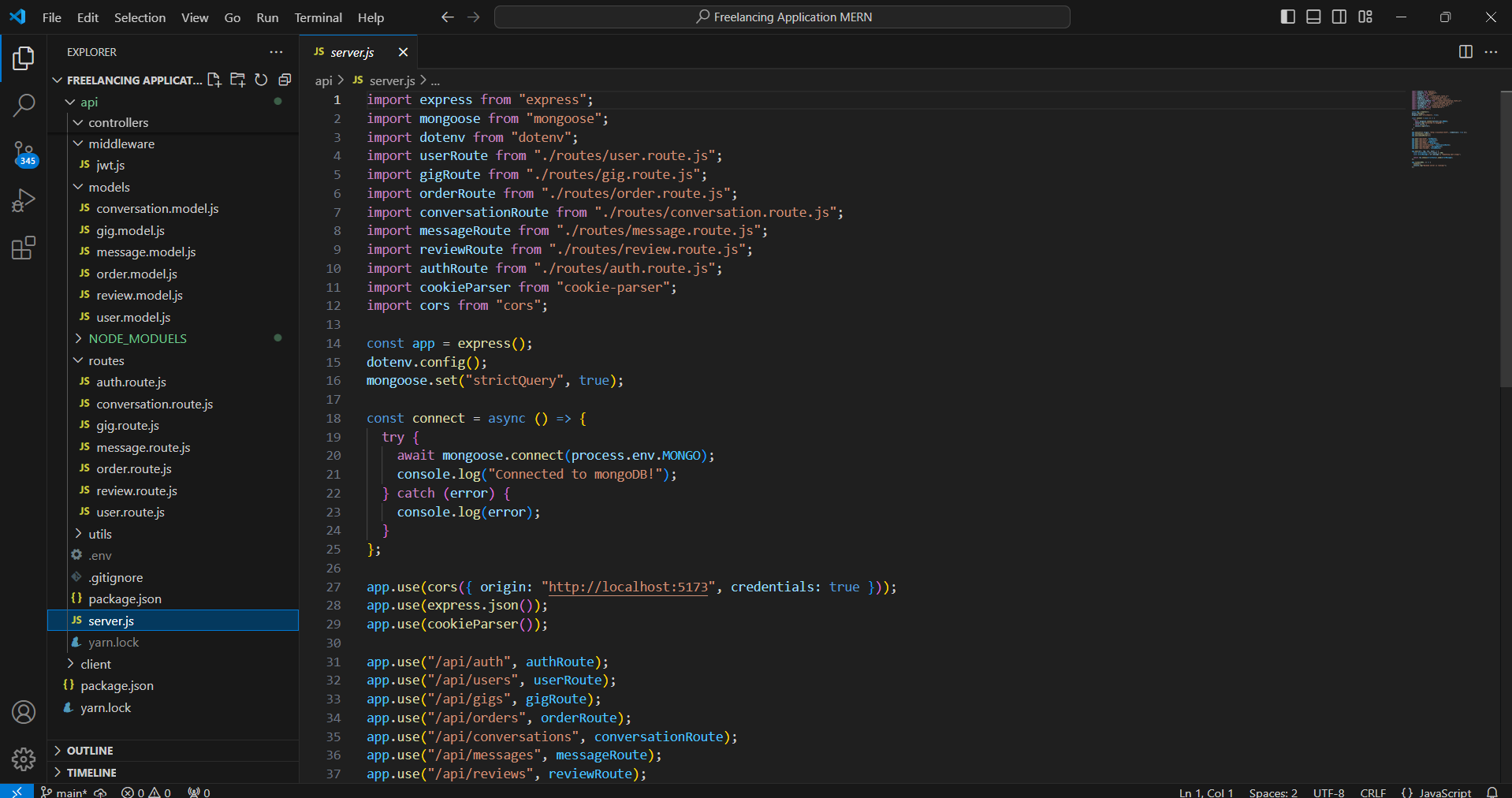












**Testing:**

* Unit Testing: Using Jest and React Testing Library to test individual functions and components in isolation.
* Integration Testing: Using Jest, React Testing Library, and Supertest to test interactions between components and backend APIs.
* End-to-End Testing: Using Cypress to simulate real user workflows and verify that the application functions correctly from start to finish