**Full-Stack Angular and Spring Boot Application**

This project is a full-stack web application built using Angular and Spring Boot. It provides a user authentication system and product management functionality. The application allows users to register, log in, manage accounts, and edit, delete, and view products and delete product. Below is a detailed explanation of how each component functions.

**Frontend (Angular)**

Angular application consists of several components and services:

**1. Authentication System**

**AuthService:**

Handles API calls related to user authentication (registration, login, fetching user details).

Sends HTTP requests to the backend UserService API.

**RegisterComponent:**

Provides a form for new users to sign up.

Calls AuthService.registerUser(user) to send user details to the backend.

Displays success or error messages based on the backend response.

Redirects to the login page upon successful registration.

**LoginComponent:**

Allows users to enter their credentials to log in.

Calls AuthService.loginUser(user) to verify login credentials.

Stores authentication tokens in local storage for session management.

Redirects authenticated users to their dashboard.

**UserDetailsComponent:**

Displays all registered users in a table.

Allows admins to edit or delete user accounts.

Uses AuthService.getAllUsers() to fetch user details.

Uses modals (Bootstrap-based) for updating user details.

**2. Product Management System**

**ProductService:**

Handles API calls related to product management.

Sends HTTP requests to the backend ProductService API.

**AddProductComponent:**

Provides a form for adding new products.

Calls ProductService.addProduct(product) to send product data to the backend.

Displays success or error messages based on the backend response.

**ProductListComponent:**

Fetches and displays all available products from the backend.

Uses ProductService.getAllProducts() to retrieve data.

Allows users to delete products.

**Angular Routing**

Angular Router to navigate between pages:

/register → Registration page.

/login → Login page.

/user-details → Admin page showing all users.

/add-product → Page to add new products.

/product-list → Page to list and manage products.

**Backend (Spring Boot)**

My backend is built using Spring Boot with RESTful APIs for user authentication and product management.

**1. UserService API**

UserController.java

Handles user registration, login, retrieval, updating, and deletion.

API Endpoints:

POST /register → Registers a new user.

POST /login → Authenticates a user.

GET /users → Fetches all registered users.

PUT /users/{id} → Updates user details.

DELETE /users/{id} → Deletes a user.

UserService.java

Contains business logic for handling user-related operations.

UserRepository.java

Uses Spring Data JPA to communicate with the database.

User Entity (User.java)

Defines the database schema for users (username, email, password).

---

**2. ProductService API**

ProductController.java

Handles adding, retrieving, updating, and deleting products.

API Endpoints:

POST /products → Adds a new product.

GET /products → Retrieves all products.

PUT /products/{id} → Updates product details.

DELETE /products/{id} → Deletes a product.

ProductService.java

Contains business logic for handling product-related operations.

ProductRepository.java

Uses Spring Data JPA to interact with the database.

Product Entity (Product.java)

Defines the database schema for products (name, price, description).

**Overall Workflow**

1. A new user registers via the frontend, which sends data to UserService in the backend.

2. The user logs in, and an authentication token is stored.

3. The authenticated user adds or views products via ProductService.

4. Admins manage users and products (edit, delete).

5. The frontend dynamically updates based on backend responses.

**Technologies Used**

**Frontend**

Angular CLI: To create and manage the Angular project.

TypeScript: For developing the frontend logic.

Angular Forms: For handling form validation and user input.

Angular Router: For navigation between components.

Bootstrap: For styling and responsive design.

HTTP Client Module: For API communication.

**Backend (Spring Boot)**

Spring Boot: To build a RESTful backend.

Spring Data JPA: For database interaction.

MySQL: For storing user and product data.

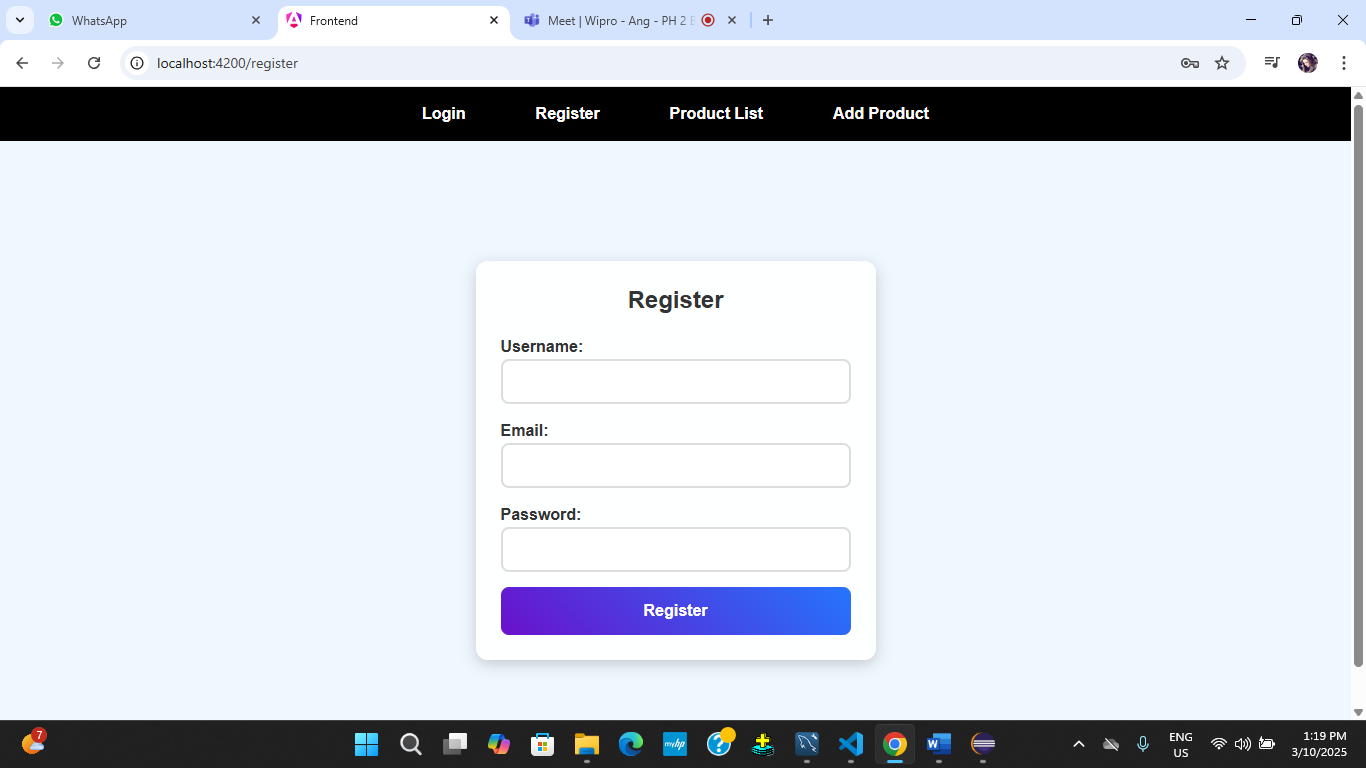
REST APIs: For client-server communication.

**Conclusion**

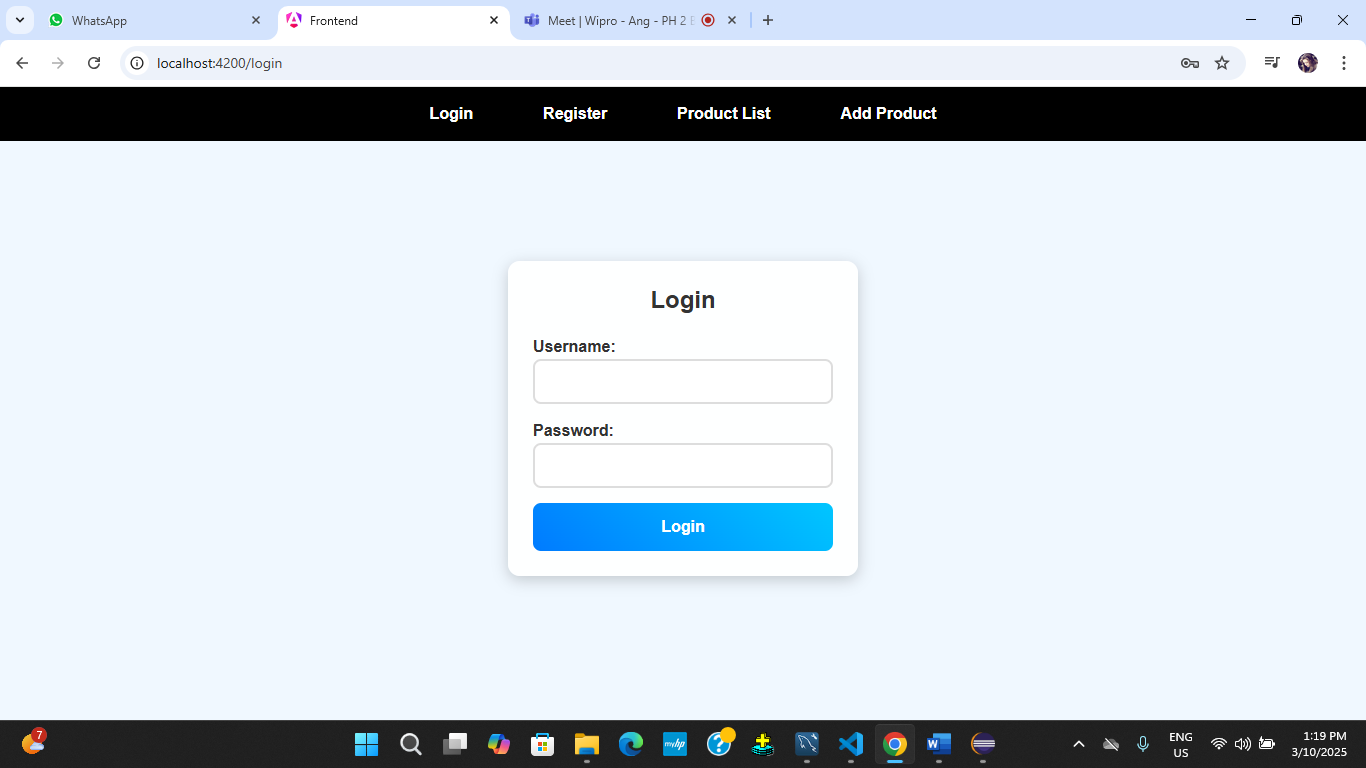
This full-stack project integrates Angular and Spring Boot to provide a secure authentication system and a product management platform. It follows RESTful API principles and best practices, making it scalable, maintainable, and efficient.

**Output Screenshots**

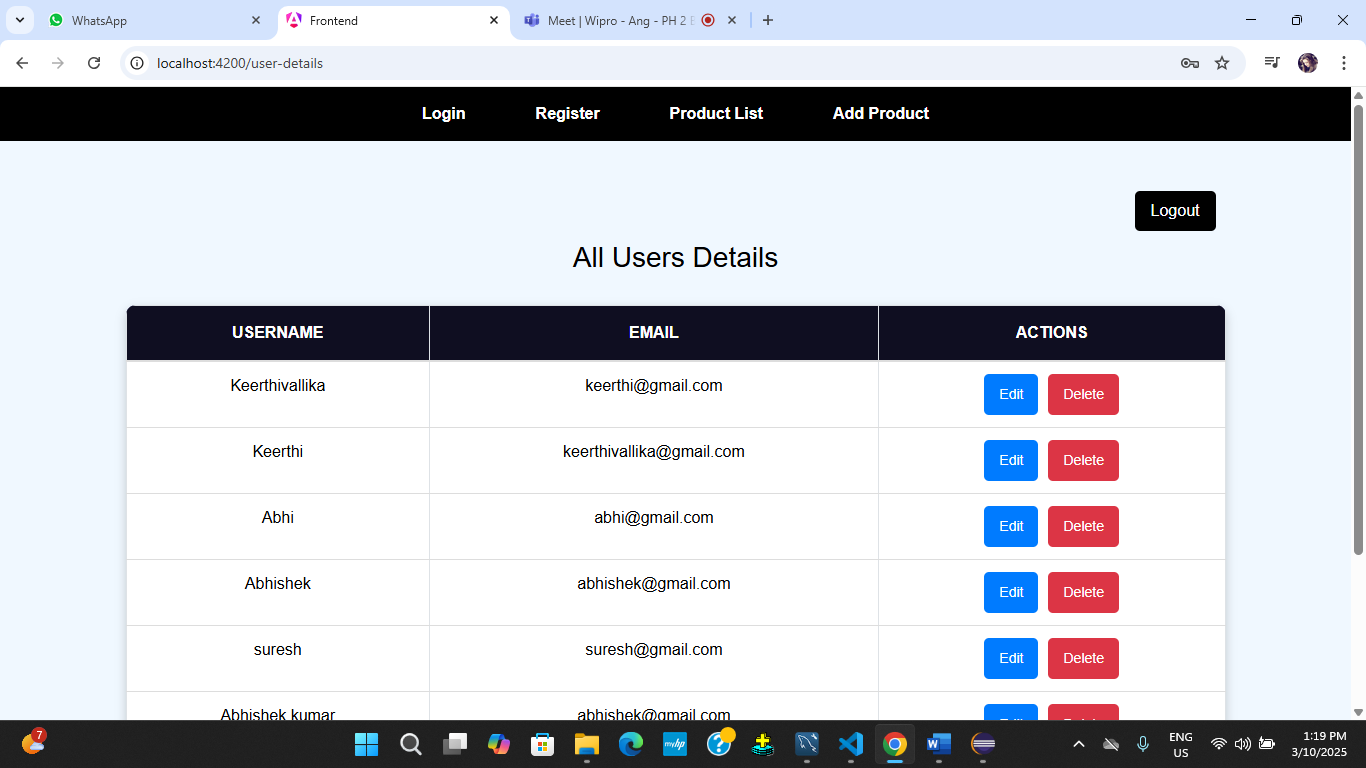
**User Registration Page**



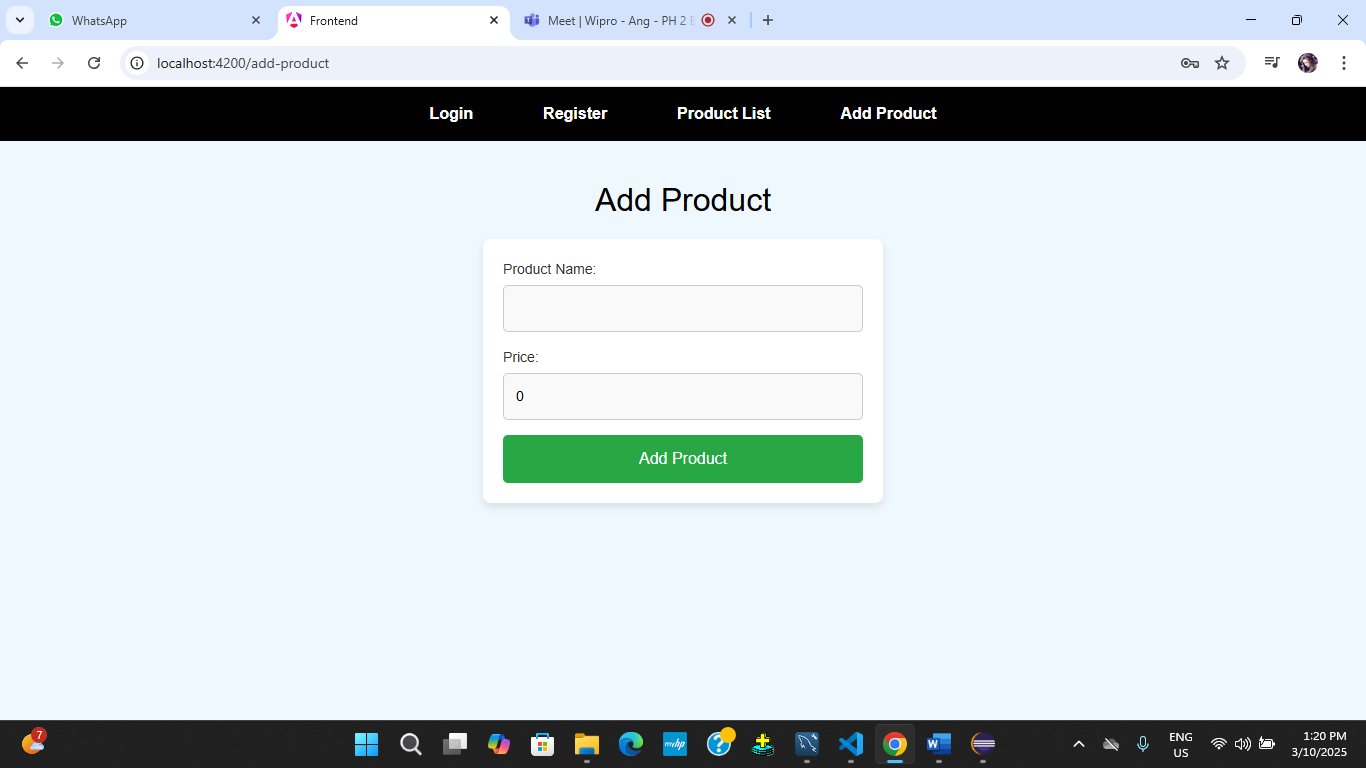
**User Login Page**



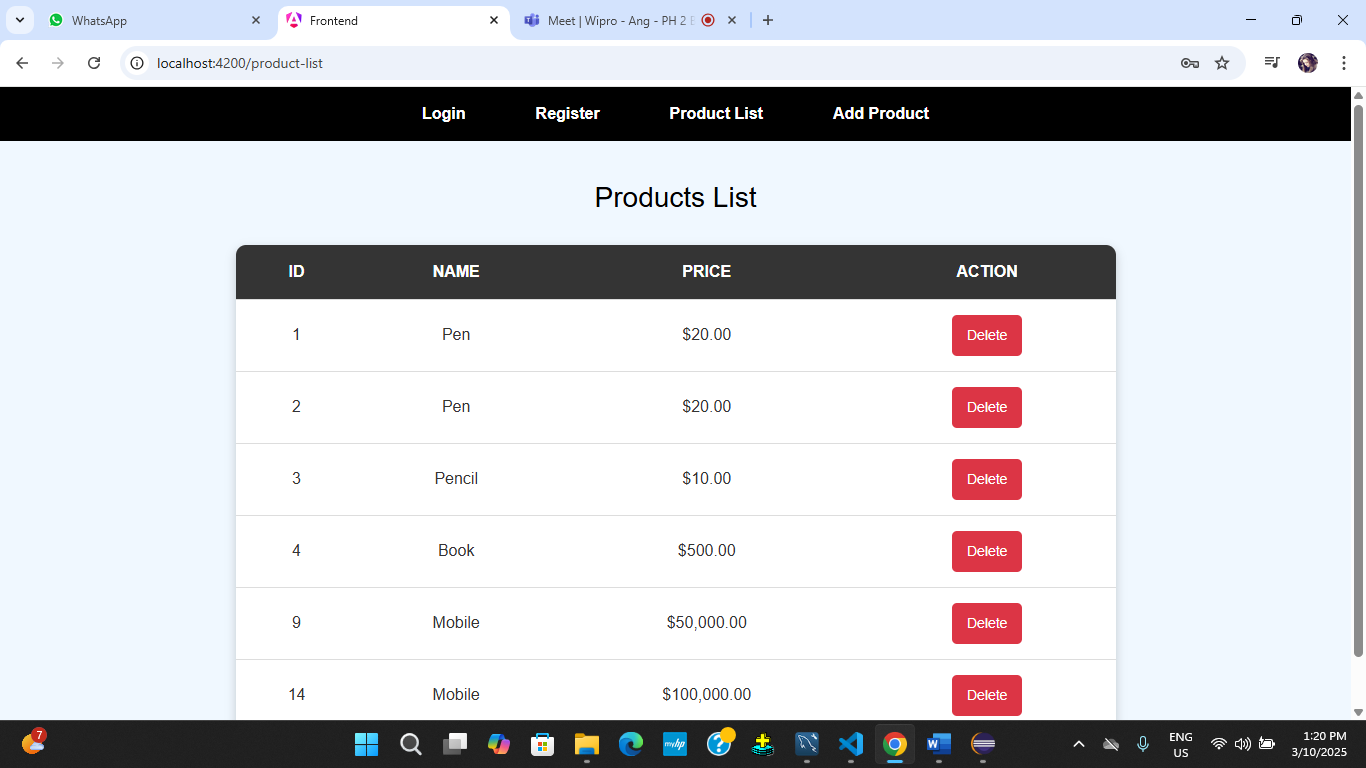
**User Details Page**



**Add Product Page**

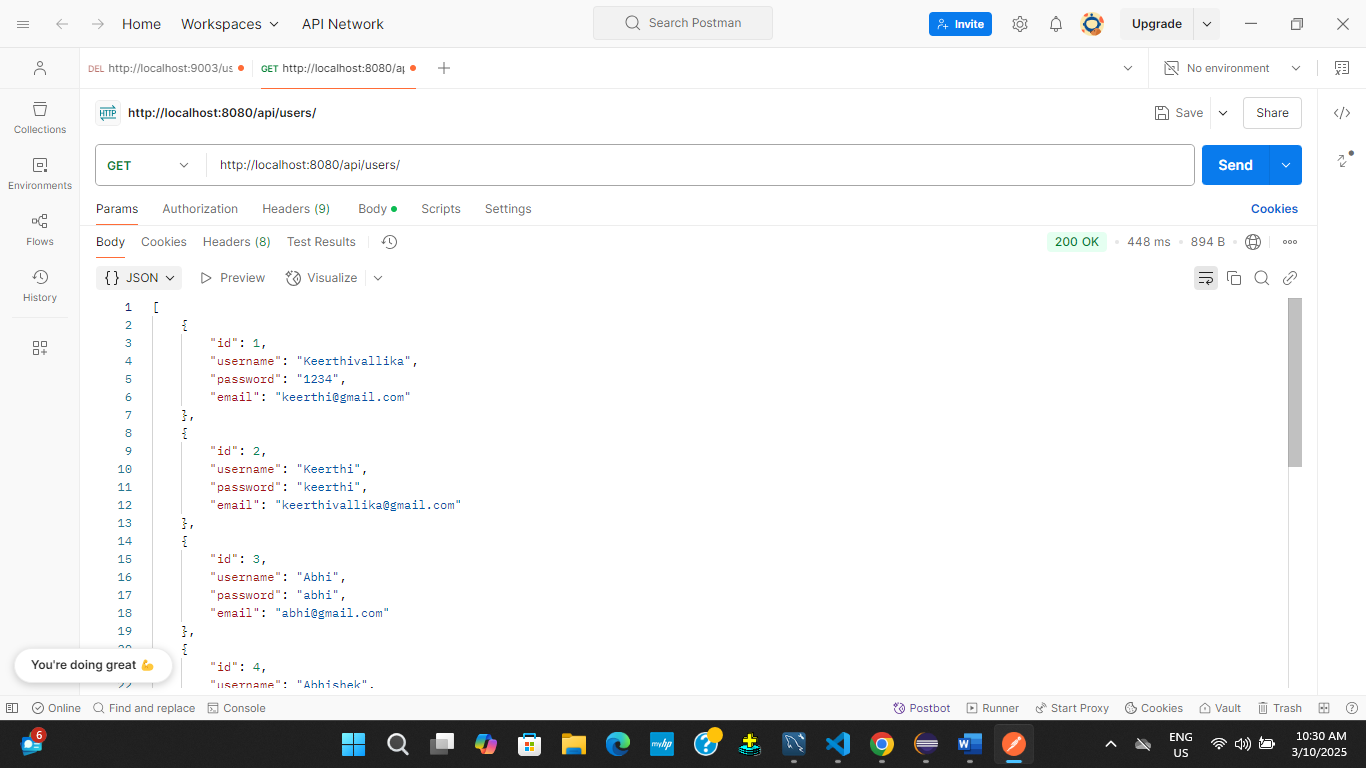


**Products List Page**



**API Responses**

**User Details**



**Product Details**

