Full Stack Development with MERN

Frontend Development Report

|  |  |
| --- | --- |
| Date | 20-07-2024 |
| Team ID | SWTID1720177025 |
| Project Name | Online Complaint Registration And Management System |
| Maximum Marks | 10 |

**Project Title: Online Complaint Registration And Management System**

Date: 20-07-2024

Prepared by: Bhavitha.G

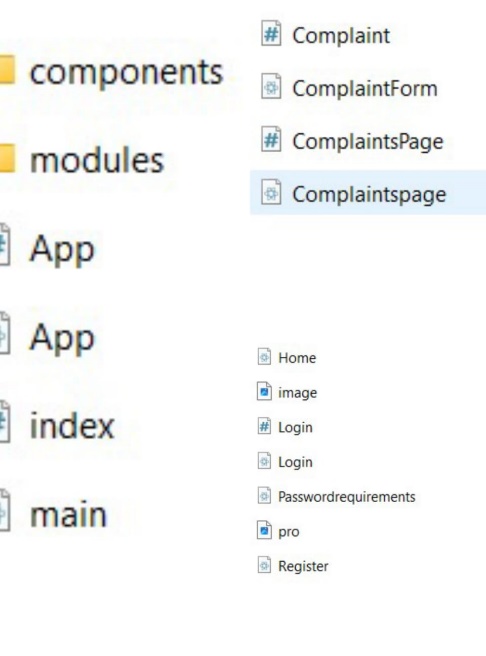
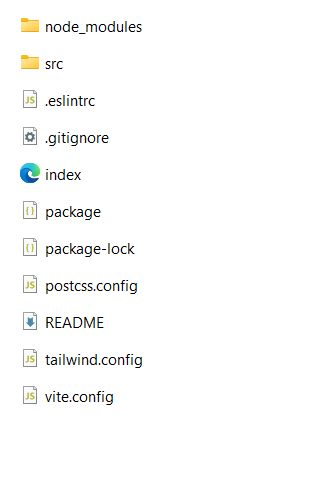
**Objective**

The objective of this report is to document the frontend development progress and key aspects of the user interface implementation for the **Online Complaint Registration and Management System** project.

**Technologies Used**

* **Frontend Framework:** React.js
* **State Management:** Context API
* **UI Framework/Libraries:** Tailwind CSS
* **API Libraries:** Axios

**Project Structure**

****

The assets`folder contains all the images and static resources essential for the application.

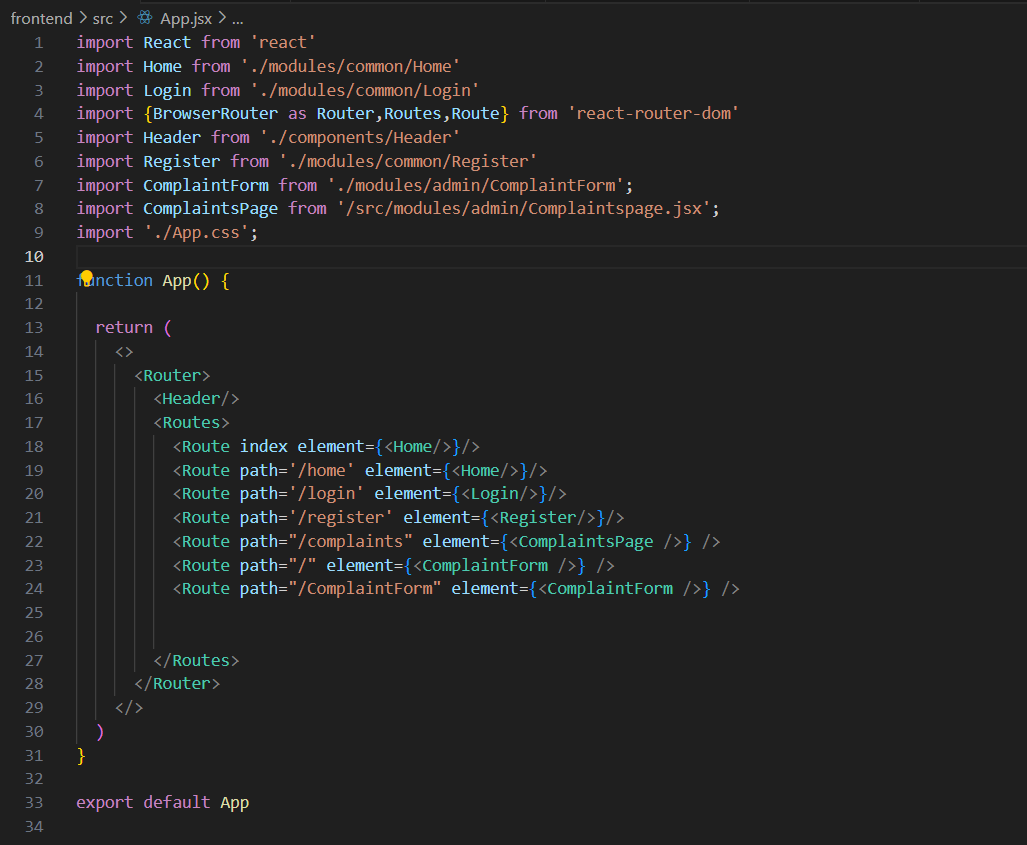
Inside the modules folder, the application's pages are organized into well-structured subfolders, supporting modular development and maintainability.

The vite.config.js file is utilized to enhance the React app's performance, providing faster load times compared to the default Webpack bundler.

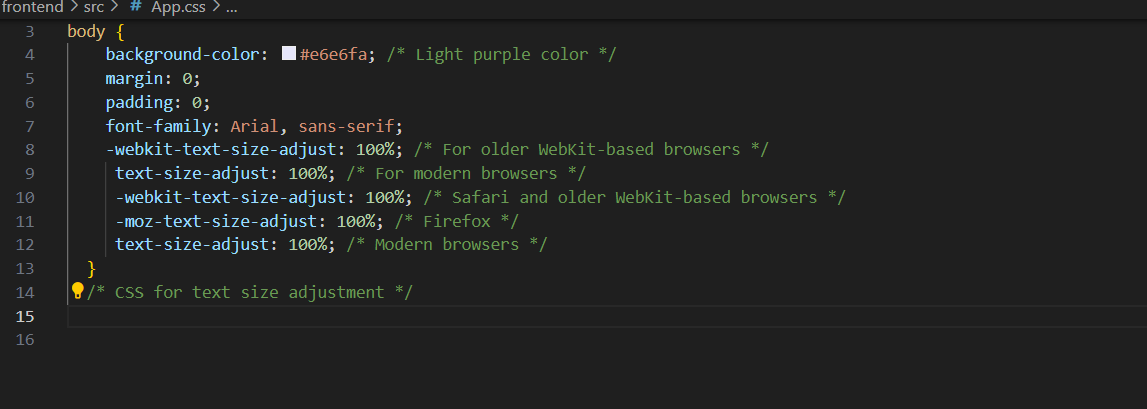
Additionally, the node\_modules folder encompasses all the required dependencies and libraries needed for the frontend to operate effectively.

**Key Components**

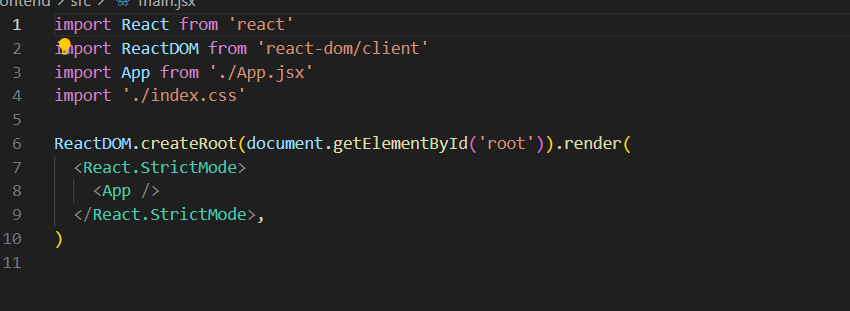
1. **App.js**
   * Responsible for routing and main application layout.
   * Front end source contain app.js and app.css



**App.css**

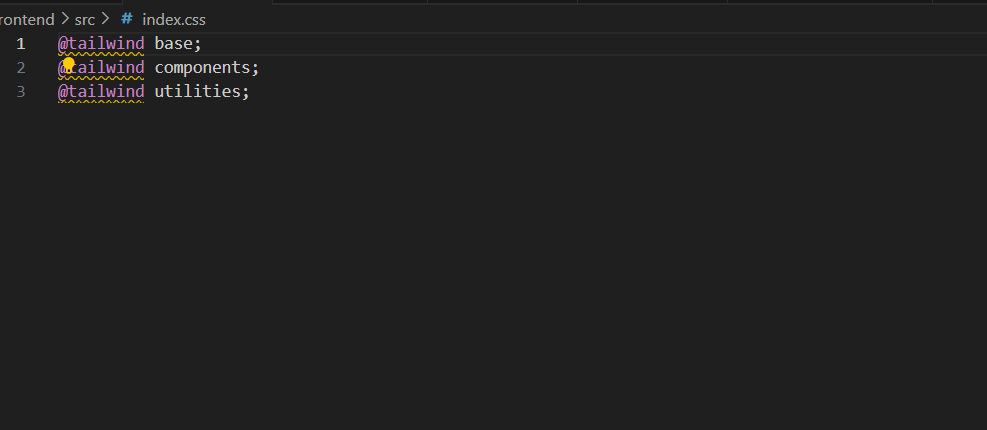


**Main.jsx**

****

**Index.css**

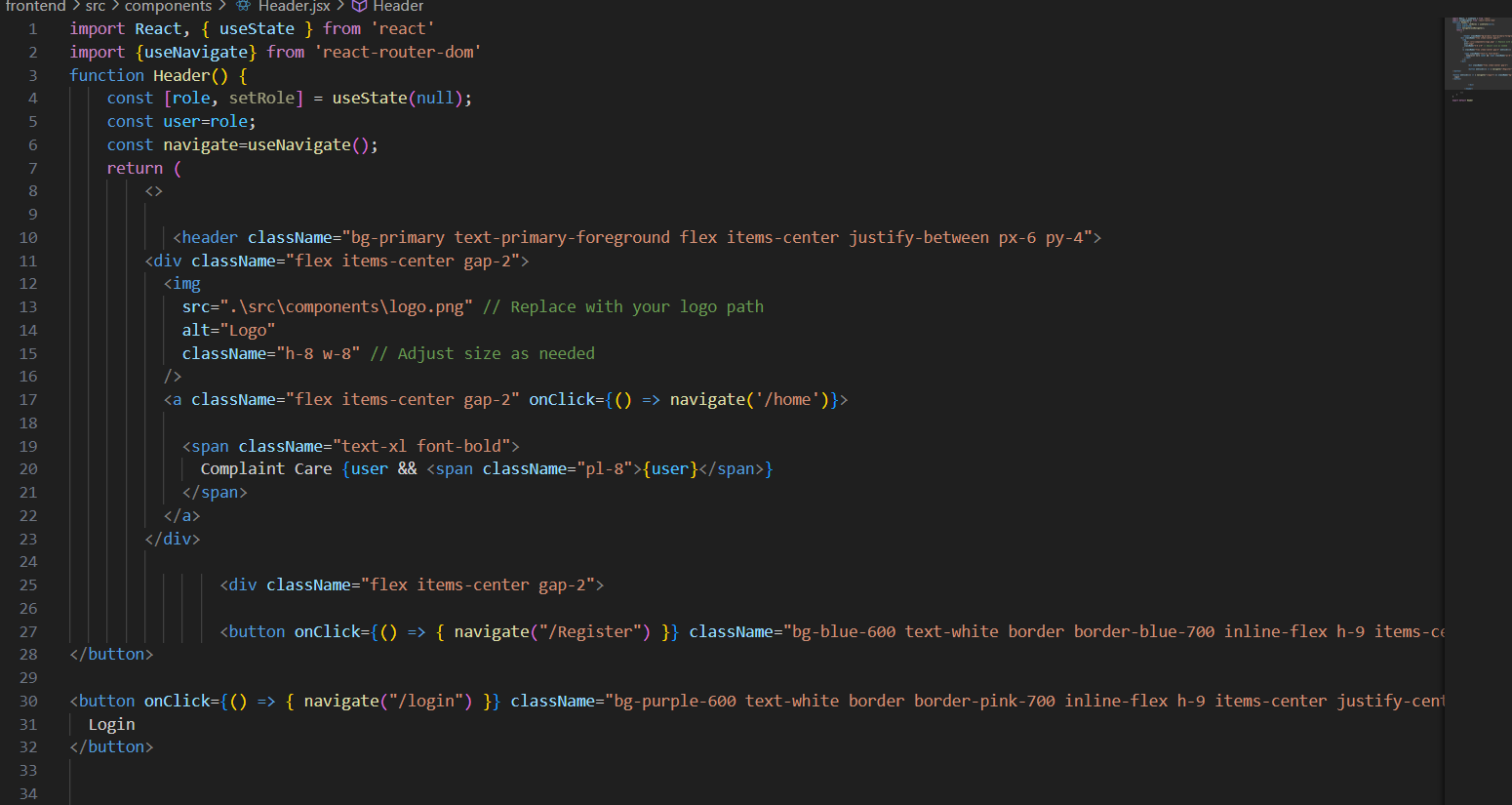
**Here we used tailwind library**

****

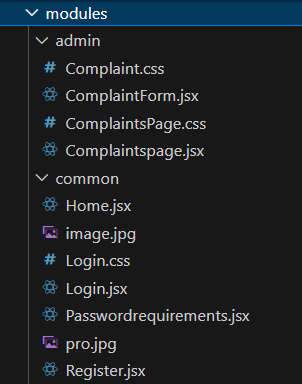
1. **/components**
   * Contains reusable UI components used across the application.

Here I have added only header component

**Header.jsx**

****

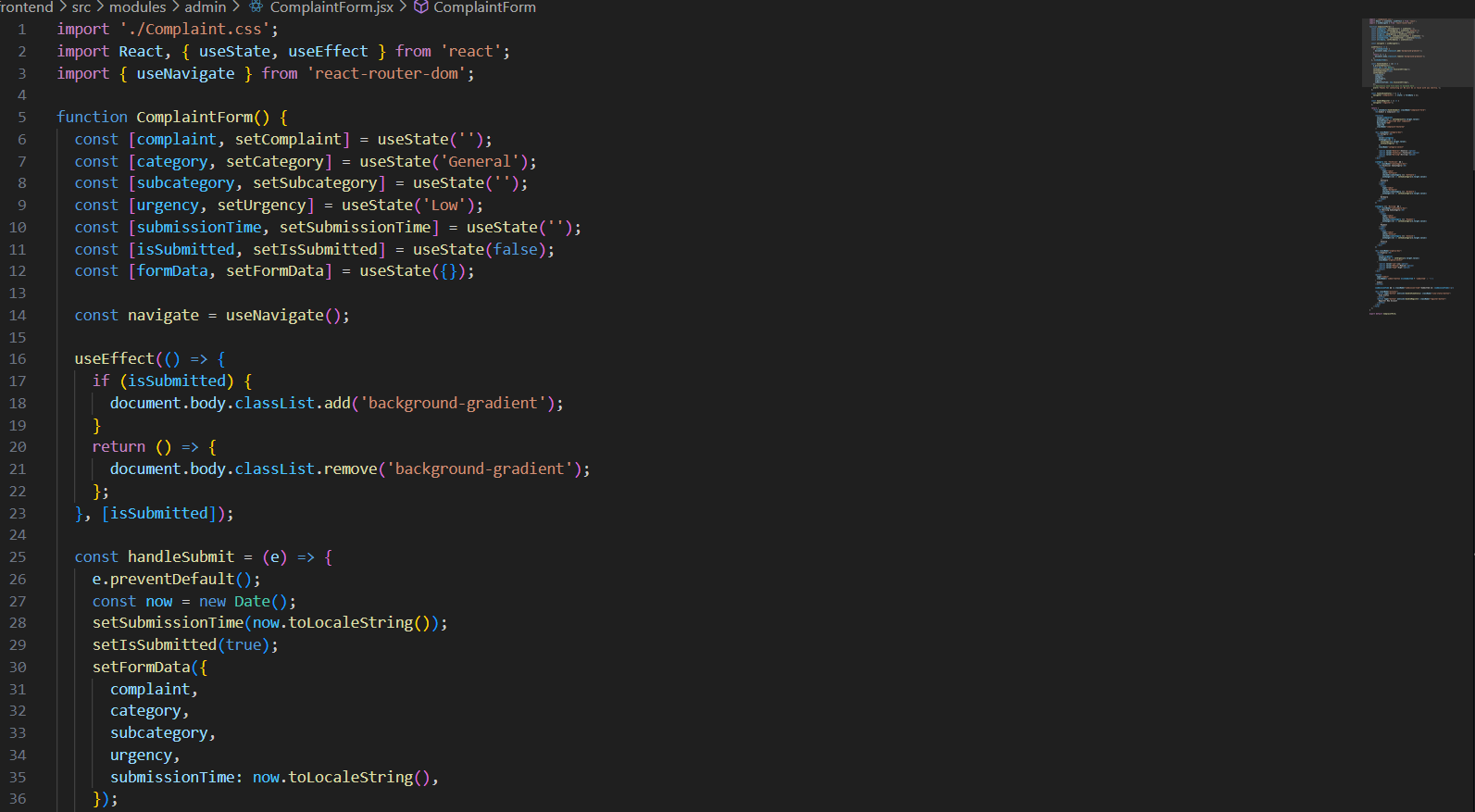
1. **/pages(/modules)**
   * Includes different pages for Web App.



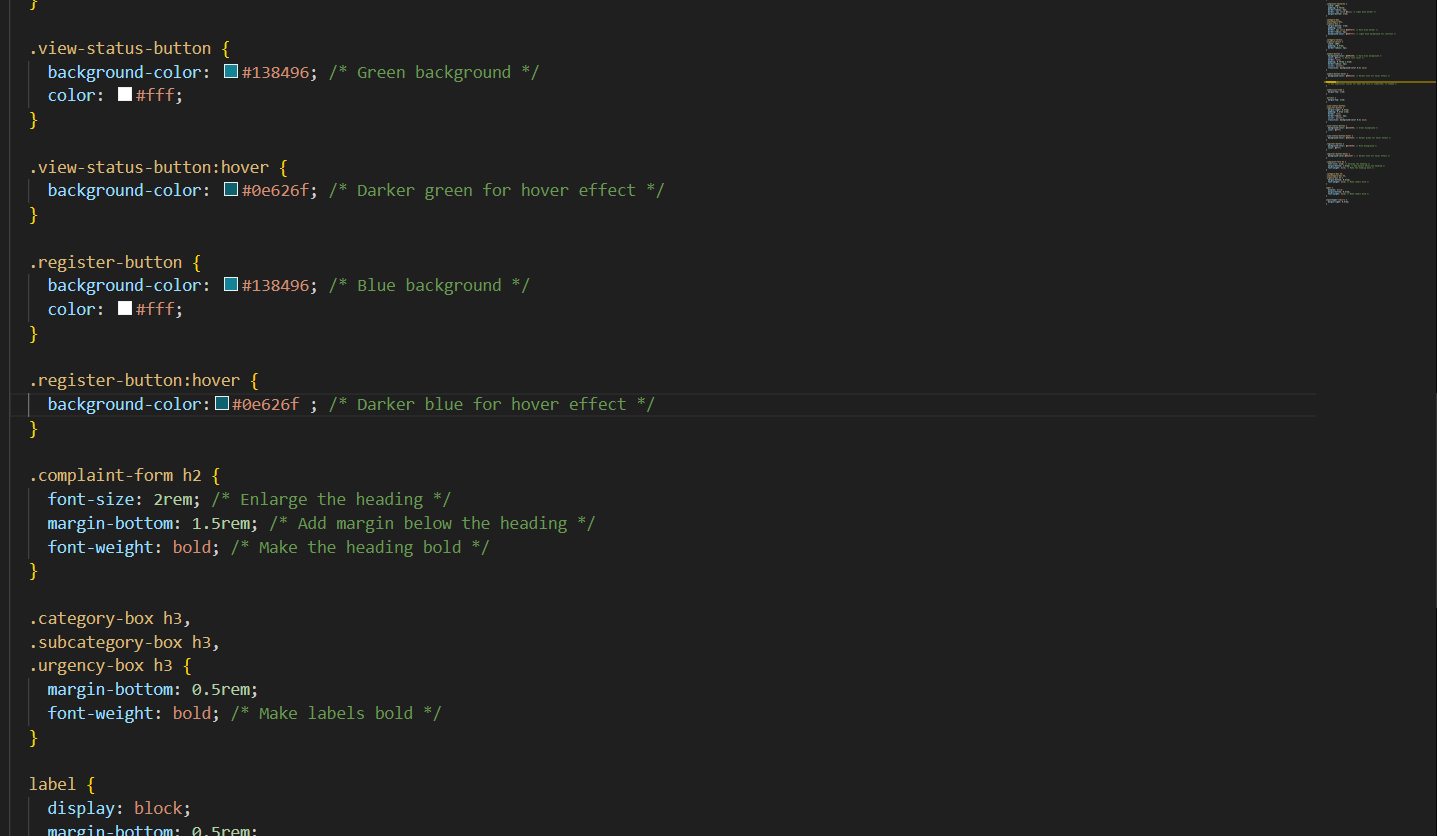
In modules we added 2 folder as

**/admin**

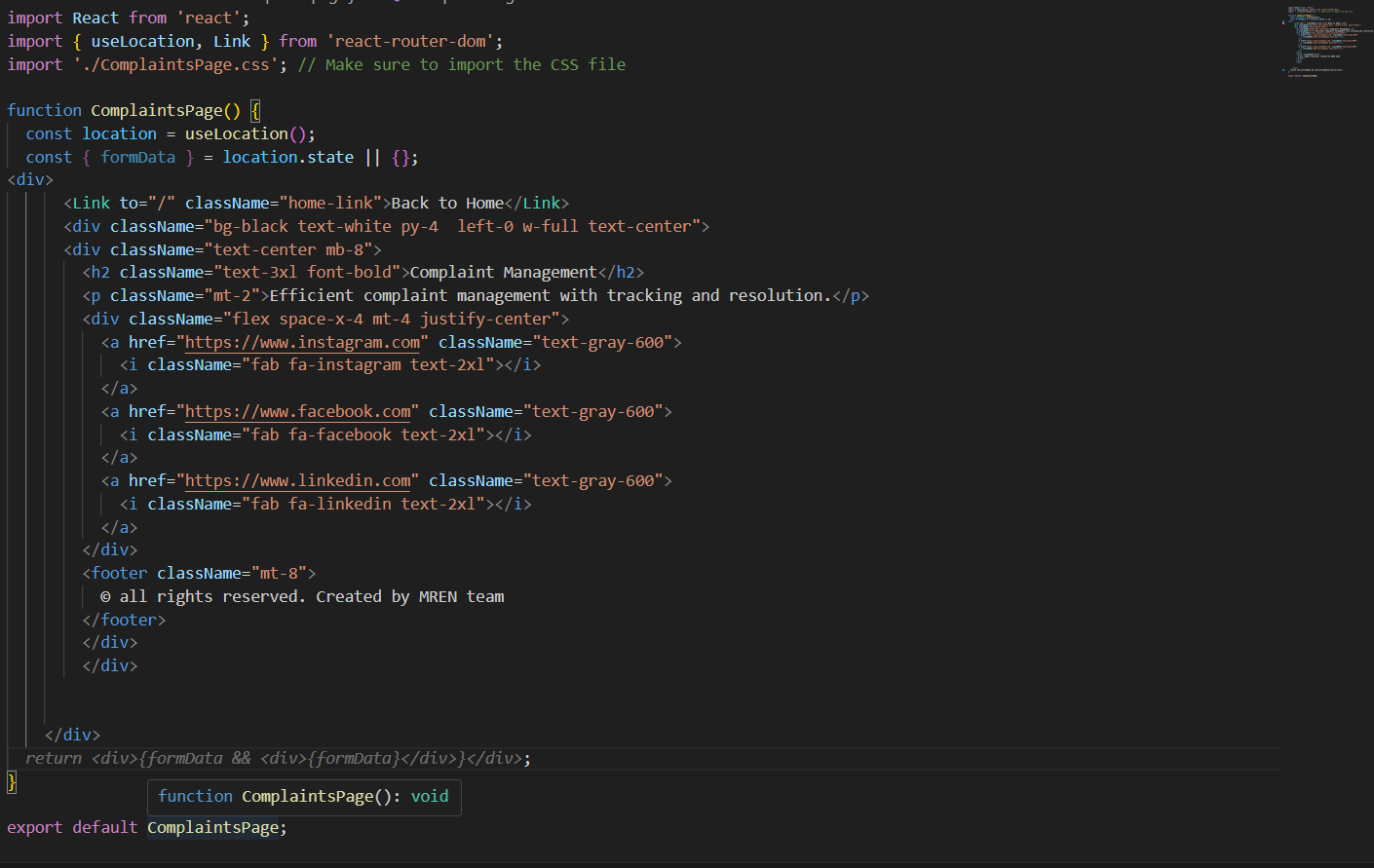
**/ComplaintForm.jsx**

****

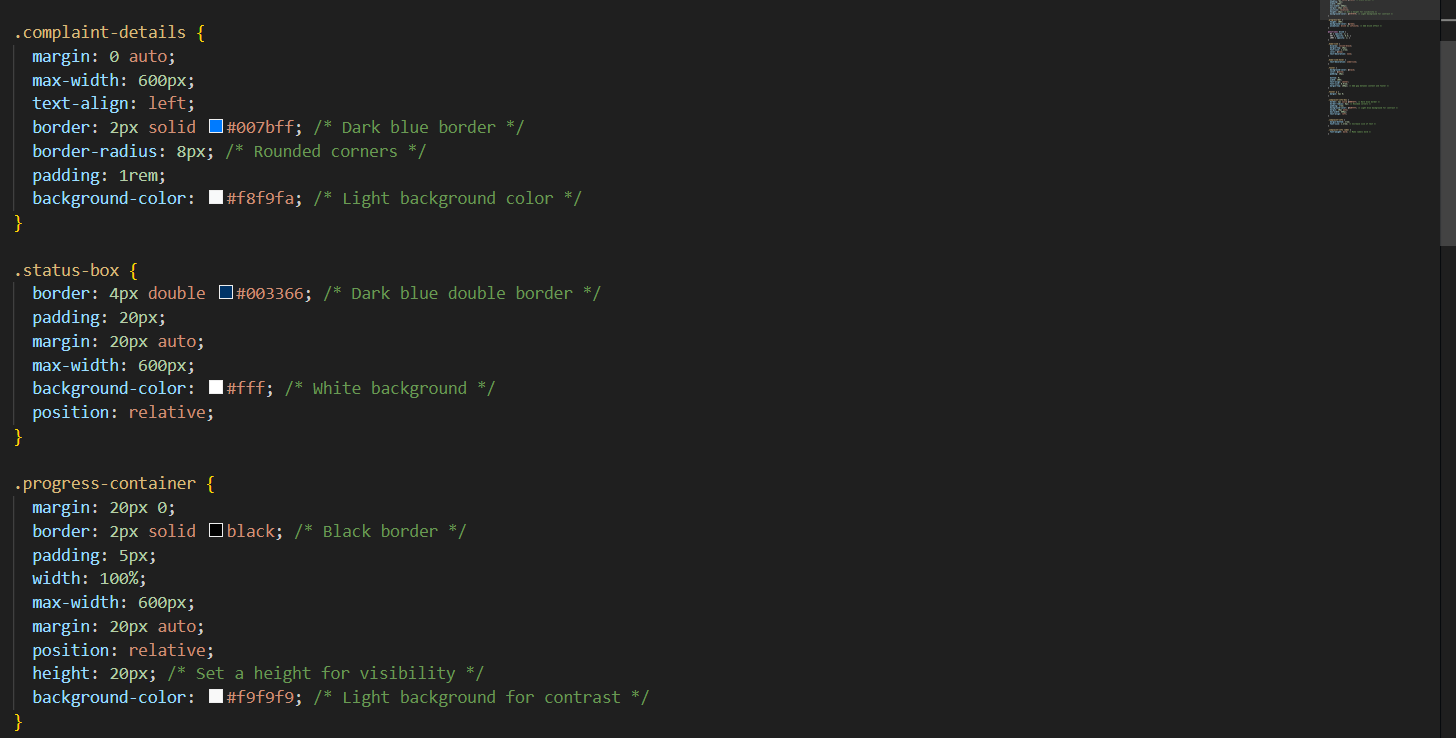
**/Complaint.css**

****

**/ComplaintsPage.jsx**

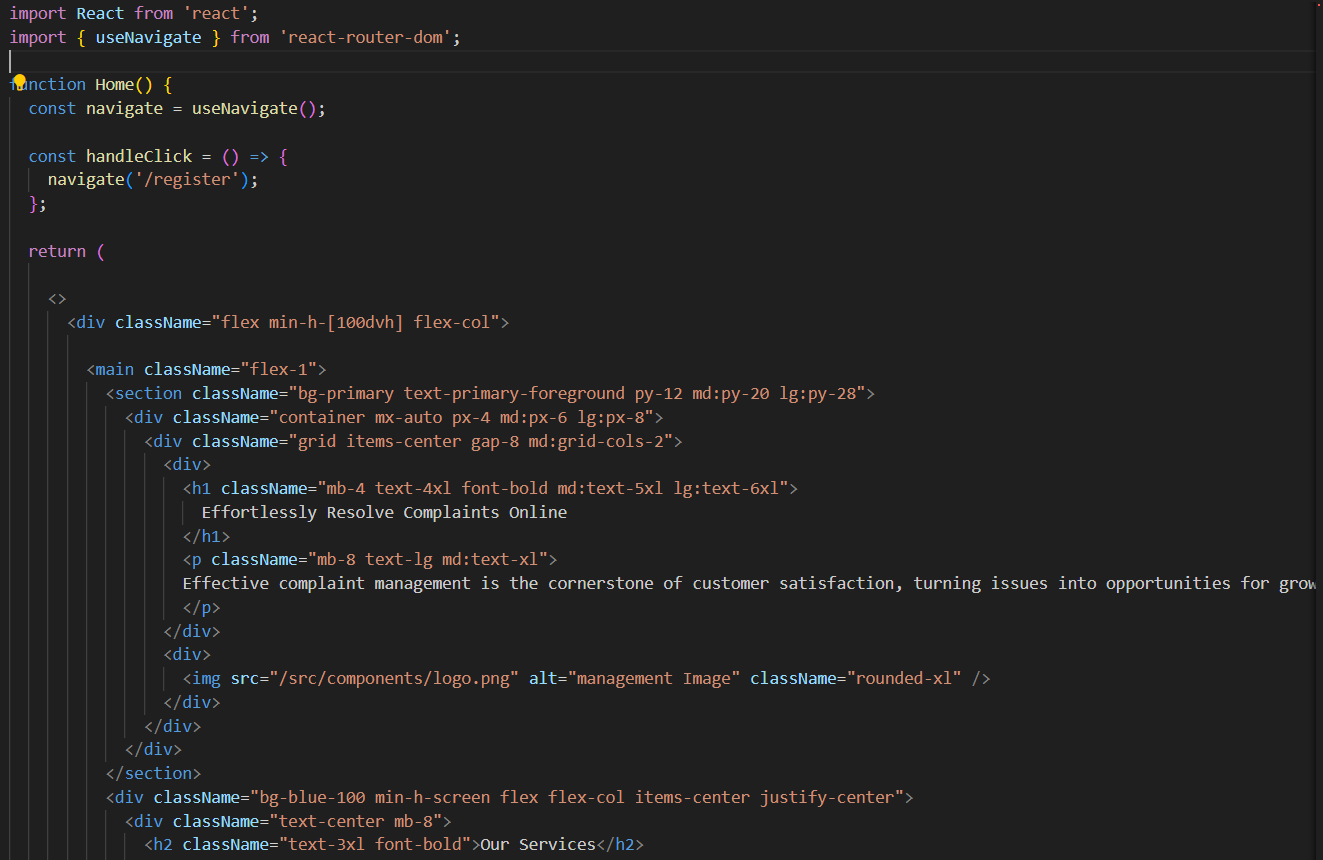
****

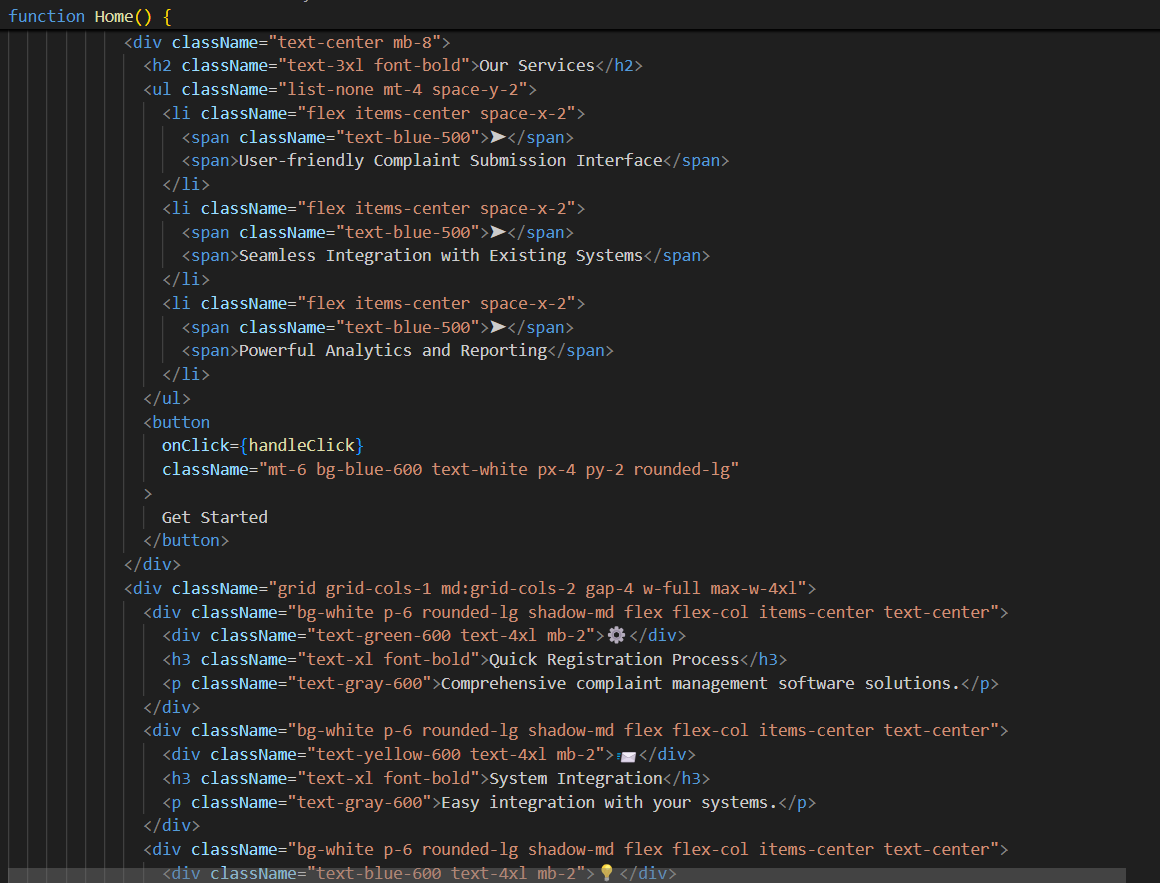
**/ComplaintsPage.css**

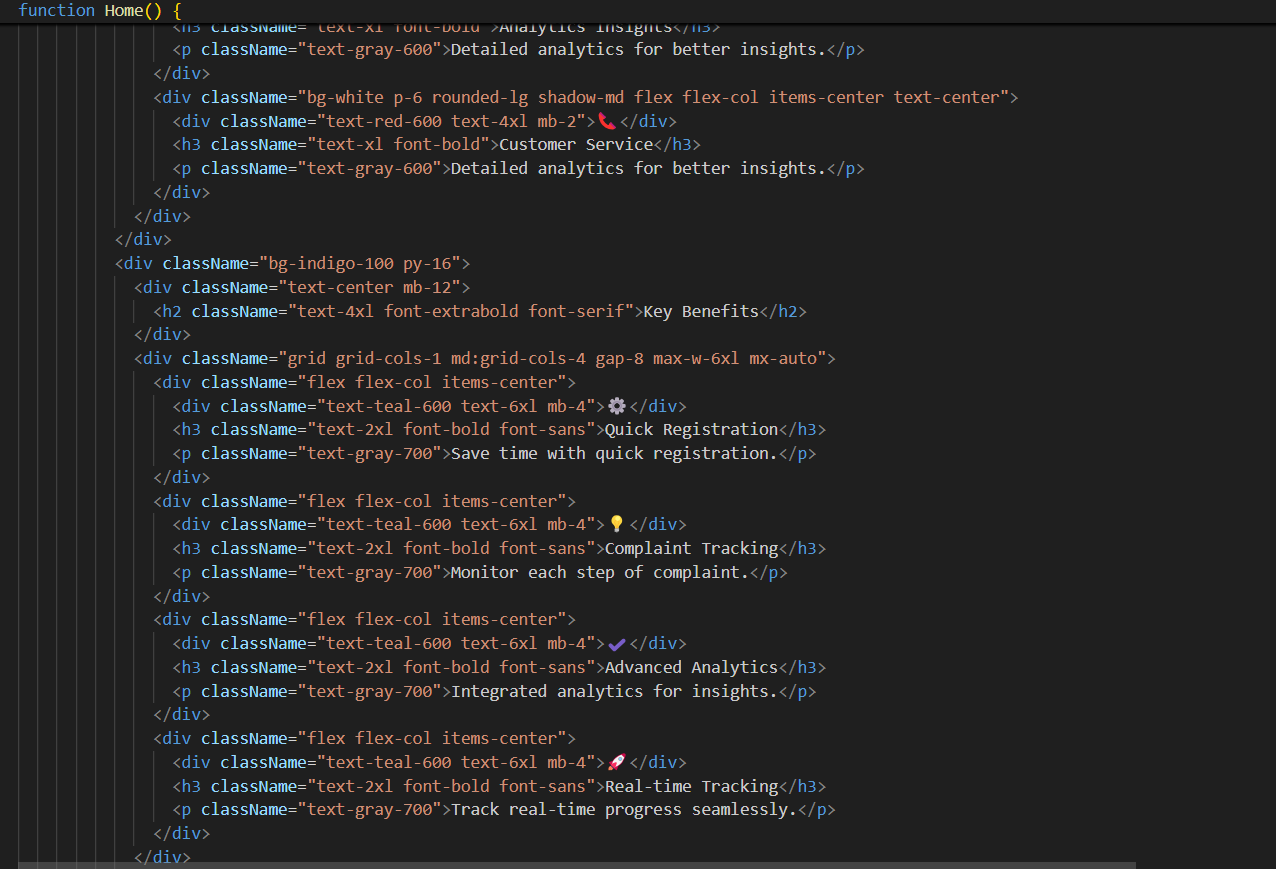
****

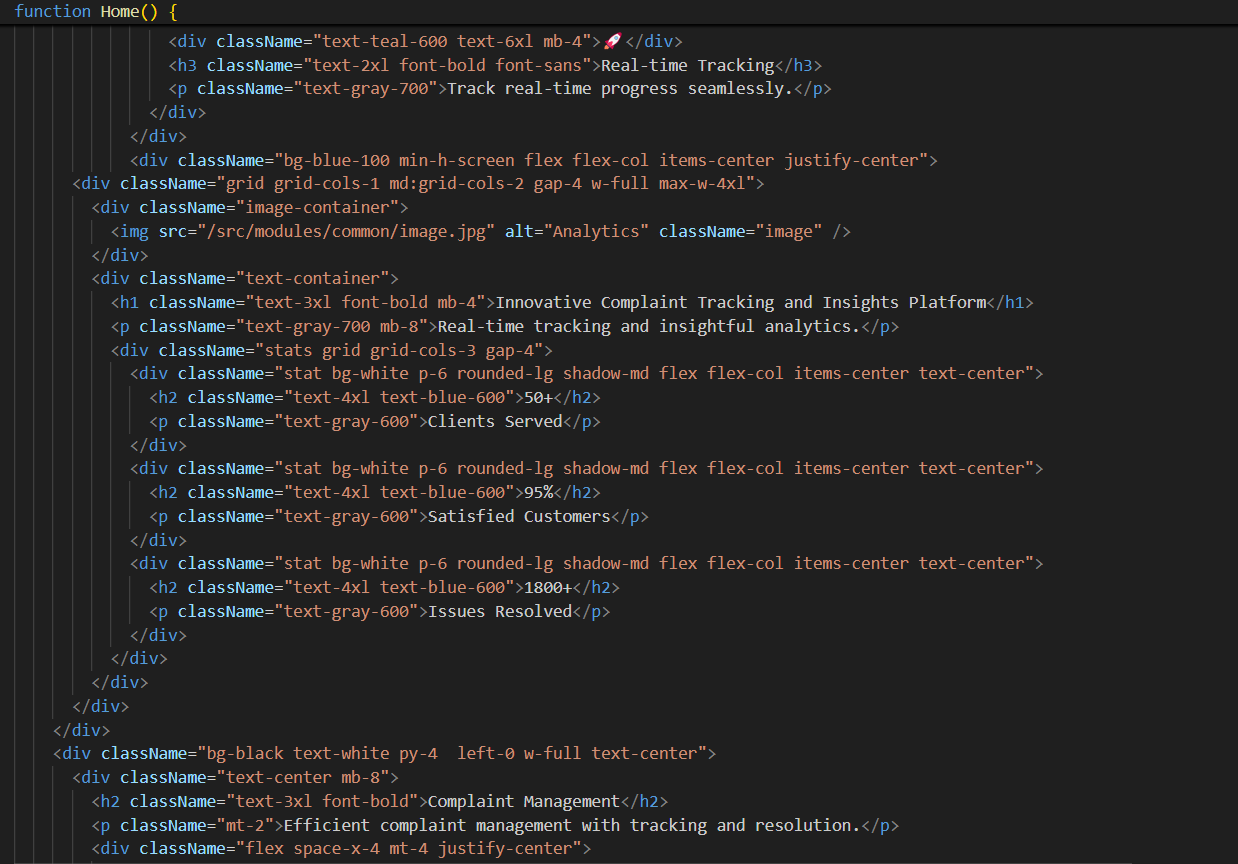
* **We have home.jsx as frontpage /home** - Landing page of the application.

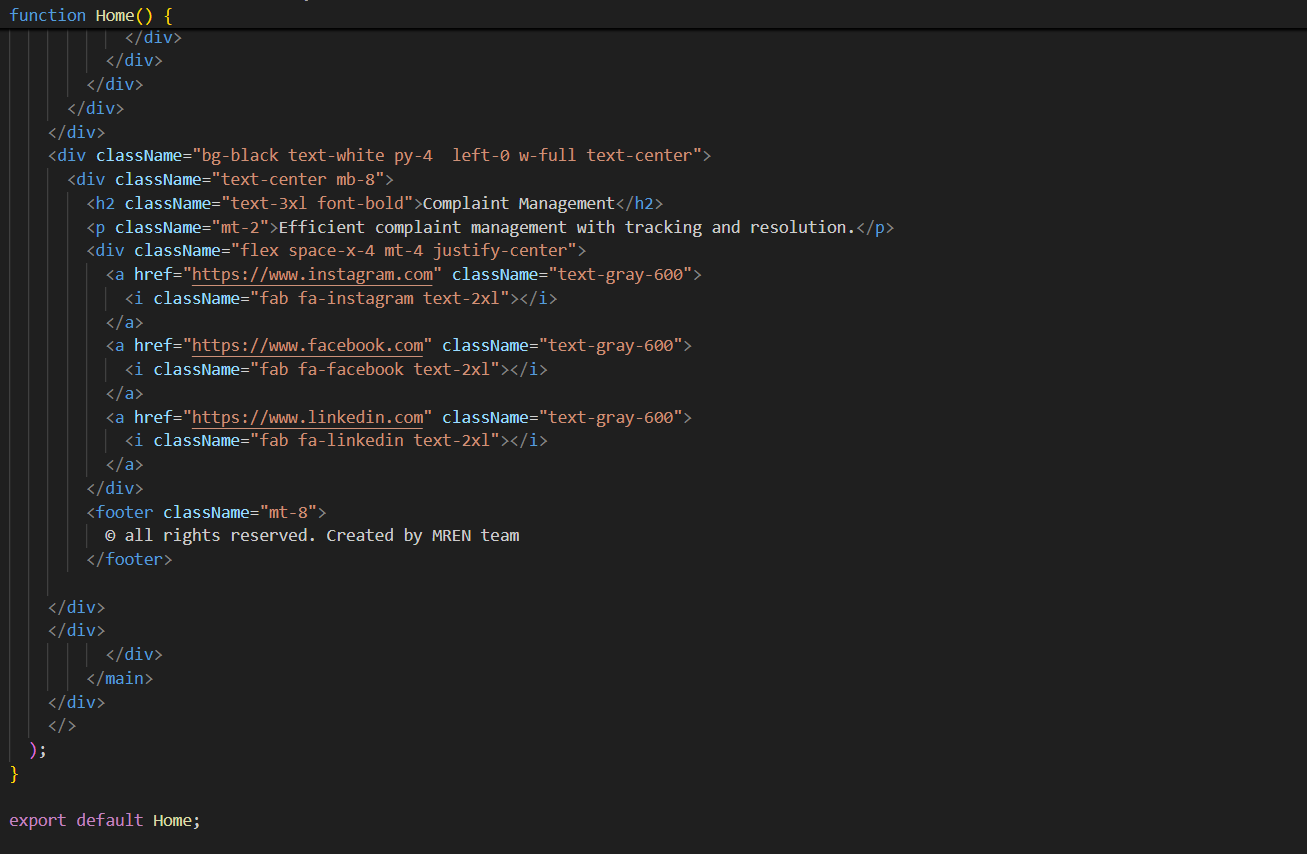
**/home**.jsx





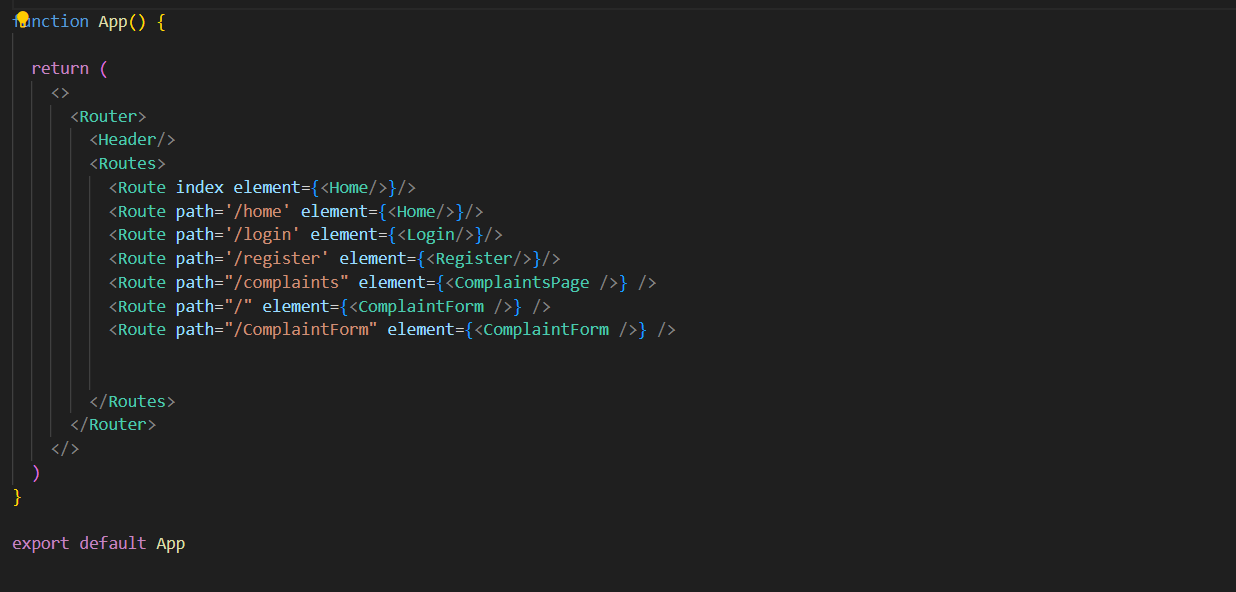






**Routing**

Routing is managed using React Router. Here are the main routes:



* **/home - Routes to home page**
* **/login - Routes to Login page**
* **/register-Routes to Register page**
* **/compaints form -Routes to add complaint format**
* **/complaints page -Routes to view status**

**State Management (If Applicable)**

State management is achieved using [Context API].

The Context API is utilized to manage and share state across various components within your React application. This technology helps avoid prop drilling and makes state management more efficient and centralized.

**Integration with Backend**

The frontend communicates with the backend APIs hosted on [backend URL]. Key endpoints include:

* GET /api/data - Retrieves data for display.
* Post/api/users/register- Handles user authentication.
* Post/api/login- Handles user authentication
* GET /home - displays quote and buttons
* GET /about– Retrieves all details about us
* GET /details – User describes about their complaint
* GET /category – Add the category of the complaint
* GET /urgency – Select the urgency of the Complaint to be solved
* POST/submit – Retrieves the user complaint and send to the database
* POST/status – Display the complaint details and the status of the complaint

**User Interface (UI) Design**

* The UI design follows a minimalist design principle, emphasizing simplicity and functionality with clean lines and ample whitespace.
* Implemented using the React framework for building dynamic and responsive user interfaces, and Tailwind CSS for CSS styles, providing a utility-first approach to rapidly apply styling through predefined classes.

**Third-Party Integrations (If any)**

* **Axios**: For handling HTTP requests and integrating with the backend API.
* **Tailwind CSS**: A utility-first CSS framework used for styling the application.
* **React Router**: For managing navigation and routing within the React application.
* **React Hook Form**: For handling form validation and submission efficiently.
* **Formik**: For form management and validation (if used alongside or instead of React Hook Form).
* **React Icons**: For including a wide variety of customizable icons in the application.
* **Lodash**: For utility functions to simplify common programming tasks.
* **Moment.js**: For date and time manipulation (if used).
* **dotenv**: For managing environment variables in the development process.
* **jsonwebtoken**: For handling JWT (JSON Web Token) authentication (if used in backend integration).