**Full Stack Development with MERN**

**Frontend Development Report**

|  |  |
| --- | --- |
| Date | 18-07-2024 |
| Team ID | SWTID1720076124 |
| Project Name | Online Complaint Registration and Management System |
| Maximum Marks |  |

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

Date: 18-07-2024

Prepared by: Divyanshi Mittal, V. Sree Teja Pradeep, Manas Kamal Das, Shreya Mahakur

**Objective**

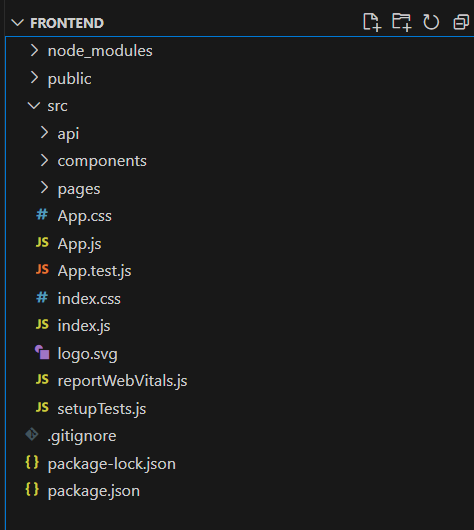
The objective of this report is to document the frontend development progress and key aspects of the user interface implementation for the Online Complaint Registration and Management project.

**Technologies Used**

* **Frontend Framework:** React.js
* **UI Framework/Libraries:** Material UI
* **API Libraries:** Axios

**Project Structure**

Screenshot of the frontend project structure:

****

**node\_modules/**: This directory contains all the dependencies and modules installed via npm.

**public/**: This directory typically contains the static assets of our project like the index.html file.

**src/**: This is the source directory where all our React code resides.

* **api/**: This directory contains the file related to API calls and services.
* **components/**: This directory is intended for storing reusable React components.
* **pages/**: This directory is intended for storing different pages of our application, separating them from reusable components.
* **App.css**: A CSS file for styling the App component.
* **App.js**: The main component file for your application, the root component.
* **App.test.js**: A file for testing the App component.
* **index.css**: A CSS file for global styles.
* **index.js**: The entry point of the React application. This file is responsible for rendering the App component into the DOM.
* **logo.svg**: An SVG file for the logo, typically used in the App component.
* **reportWebVitals.js**: A file for measuring the performance of your app.
* **setupTests.js**: A file for setting up testing configurations.

**Key Components**

1. **App.js**
   * Responsible for routing and main application layout. This file also uses session storage to identify the role of the user such as User/Agent/Admin and renders the Navbar accordingly by using conditional rendering
2. **/components**
   * Contains reusable UI components used across the application.
   * Our components folder consists of :  
     Admin Navbar, Agent Navbar, User Navbar, General Navbar and a user profile component which displays the details of the user and a Protected Route component which doesn’t let the users access pages which aren’t designed for their roles, for example, the USER will not be able to access any of the pages the AGENT can by typing the endpoint on their browser.
3. **/pages**
   * The pages including in our project are
     1. Admin complaint details page- Displays all the complaints registered to the admin.
     2. Agent complaint details page – Shows the assigned complaints to the agent and lets them change the status to resolve once the agent is finished with processing the complaint.
     3. Assign agents page – Lets the admin assign agents to the registered complaints.
     4. Home page – Home page for everyone.
     5. Sign in page – Page for every user to sign in, including the admin.
     6. Sign up page – Page for users and agents to register.
     7. Submit complaint – Users can submit their complaints here.
     8. User complaint details – Users can view their registered complaints in this page

**Routing**

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

* / - Home page
* /submitcomplaint – Routes to the page where user can submit his complaint
* /agentcomplaintdetails – Routes to the page where agents can view their assigned complaints and change the status of it when they are finished with processing the complaint
* /usercomplaintdetails – Routes to the page where user can view his registered complaints
* /userprofile – Routes to the component where the user can see his profile
* /assignagents – Routes to the page where the admin can see the pending complaints and assign the agents to those complaints
* /admincomplaintdetails – Routes to the page where the admin can see all the complaints
* /Sign-in – Routes to the sign in page
* /Sign-up – Routes to the sign up page

**Integration with Backend**

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

* **POST/signin** – A post request for the user to sign in using his credentials and his registered role, the api returns a json object with a message field saying Authorized or Unauthorized and his role and \_id field from the database to store it in the session storage to use it for sending requests to other endpoints and validating if the user is allowed in a certain page.
* **POST/signup** – A post request for the user to sign up
* **GET/getprofile/:userId -** A get request for the user to see his profile details, the userId field will be taken from the Session Storage as everytime he logs in his role and Id will be saved to user for purposes like this, and everytime the user logs out the Id and role will be deleted from the session storage
* **POST/complaintReg –** A post request for the user to submit his complaints
* **GET/getusercomplaints/:userId –** A get request to view all the requests that are in the database with this registered userID
* **GET/getagents –** A get request to view all the registered agents
* **GET/getallcomplaints –** A get request to see all the registered complaints
* **GET/pendingcomplaints –** A get request to see only the pending complaints
* **POST/assignagents –** A post request for admins to assign agents to the complaints
* **GET/viewagentcomplaints/:agentId –** A get request to view the complaints assigned to the particular agent, the agentId will be stored in the session storage as mentioned above
* **POST/updatecomplaintstatus –** A post request for agents to update to status after processing the issue.

**User Interface (UI) Design**

* **Consistency**: The UI design follows a consistent approach throughout the application. Elements such as buttons, icons, and fonts maintain uniformity to provide a seamless experience. This includes using consistent colour schemes, typography, and layout structures.
* **Simplicity:** The design emphasizes simplicity to ensure that users can navigate and interact with the application effortlessly. Unnecessary elements are eliminated, and the interface is kept clean and straightforward.
* **Clarity**: The design prioritizes clarity to make sure that all UI elements are easily understandable. Text is legible, and actions are clearly defined to avoid confusion.
* **Responsiveness**: The UI design is responsive, meaning it adapts to different screen sizes and orientations. This ensures a consistent experience across desktop, tablet, and mobile devices.
* Implemented using Material-UI (MUI).

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

No Third Party Integrations are used.