**Full Stack Development with MERN**

**Frontend Development Report**

|  |  |
| --- | --- |
| Date | 12 July 2024 |
| Team ID | SWTID1719933836 |
| Project Name | Connectify |
| Maximum Marks |  |

**Project Title: Connectify**

Date: 12 July 2024

Prepared by: Aakarsh Sinha

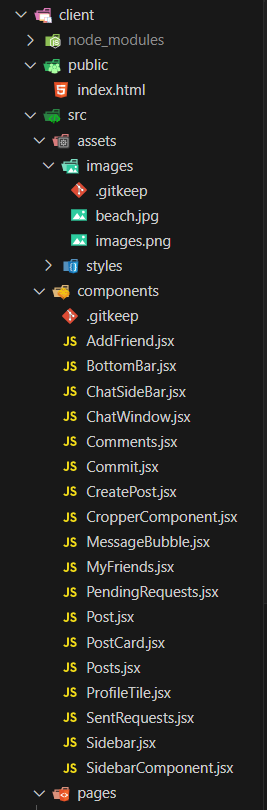
**Objective**

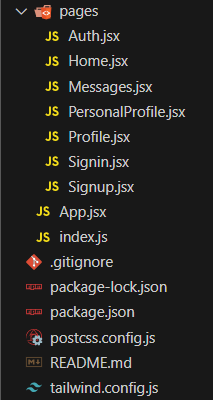
The objective of this report is to document the frontend development progress and key aspects of the user interface implementation for the Connectify project.

**Technologies Used**

* **Frontend Framework:** ReactJS, Tailwind
* **State Management:** Local State (Hooks)
* **UI Framework/Libraries:** React-Bootstrap, React Router
* **API Libraries:** mongoose, socket.io, axios, body-parser, cors, JWT(jsonwebtoken), express-validator, Postman

**Project Structure**

****

****

**Key Components**

1. **App.jsx**
   * Responsible for routing and main application layout.
2. **/components**
   * Contains reusable UI components used across the application.
3. **/pages**
   * Includes different pages for Web App.

**Routing**

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

* **/** - Landing page of the application.
* **/signup** – Sign Up page
* **/signin** – Sign In page
* **/home** – The user’s home page
* **/addfriend** – Adding friends for a user
* **/myfriends** – Display the user’s friend list
* **/pendingrequests** – Displays the pending friend requests
* **/sentrequests** – Displays friends requests sent by user
* **/personalprofile** – User profile management
* **/messages** – Chat dialogue of the user

**State Management**

State management is achieved using Local State (Hooks).

**Integration with Backend**

The frontend communicates with the backend APIs hosted on http://localhost:5000/api. Key endpoints include:

* **POST /api/user/signup** – Handles account creation
* **POST /api/user/signin** – Handles user authentication
* **GET /api/user/getusers** – Retrieves list of users
* **GET /api/user/userposts** – Retrieves a user’s posts
* **GET /api/user/getprofile** – Fetches user profile
* **GET /api/user/getpfp –** Retrieves user’s profile picture
* **GET /api/user/getfriends** – Retrieves a user’s friends
* **GET /api/user/receivedrequests** – Retrieves the friend requests received by a user
* **POST /api/user/acceptrequest –** The user can accept friend requests
* **POST /api/user/sendrequest –** The user can send a friend request
* **GET /api/user/isfriend –** To check if a user is a friend
* **GET /api/user/sentrequests** – Retrieves the friend requests sent by a user
* **GET /api/user/messages** – Retrieves a user’s messages
* **GET /api/comments/:id –** Retrieves the comments on a post
* **POST /api/comments/:id –** To add a comment on a post
* **POST /api/user/createpost –** Handles post creation
* **GET /api/user/getposts** – Retrieves posts for the user’s feed
* **POST /api/posts/like/:id –** To like a post

**User Interface (UI) Design**

* The UI design follows a MVC (Model-View-Controller) architectural design. This pattern emphasizes a separation between the software’s business logic and display, providing a better division of labour and improved maintenance.

**Model**: The Model represents the data and business logic of the application. It manages the data and provides methods for interacting with the data. The Model is responsible for encapsulating the data and providing a way to access and manipulate it.

**View**: The View is responsible for rendering the user interface (UI) of the application. It receives data from the Model and displays it in a user-friendly format. The View is responsible for handling user input and sending it to the Controller.

**Controller**: The Controller acts as an intermediary between the Model and View. It receives input from the View, interacts with the Model to perform actions, and then updates the View with the results. The Controller is responsible for managing the flow of data and logic between the Model and View.

* Implemented using React, react-bootstrap, styled-components, Tailwind, React Router.

**Third-Party Integrations**

* Socket.io
* Axios
* jsonwebtoken
* mongoose
* cors
* body-parser
* express-validator
* Postman