Training Report

On

React

Submitted in Fulfillment of the Requirements for the Award
of the degree of

Masters of Computer Application

(Batch 2022-2024)

By

Manish Kumar
1322748

Under the Supervision
Of

Ms. Nikki Sorout
(Assistant Professor)

M.M. Institute of Computer Technology & Business Management
Maharishi Markandeshwar (Deemed to be University),
Mullana (Ambala), Haryana, India
(NACC accredited Grade 'A++' University)

CANDIDATE DECLARATION

I hereby declare that the work which is being presented in this report entitled **"React Training",** in partial fulfillment of the requirement for the award of the degree of **MASTER OF COMPUTER APPLICATIONS** submitted at M.M. Institute of Computer Technology & Business Management, **Maharishi Markandeshwar (Deemed to be University), Mullana, Ambala** is an authentic work done by me during a period from June 15, 2023 to July 27, 2023 under the Guidance of **Ms. Nikki Sorout (Assistant Professor).**

The work presented in this report has not been submitted by me for the award of any other degree of this or any other Institute/University.

Manish Kumar
1322748

This is to certify that the above statement made by the candidate is correct to best of my knowledge and belief.

Date:

Name & Signature of Supervisor
Assistant Professor
Department of MMICT & BM

CERTIFICATE OF TRAINING

INTERNSHALA TRAININGS

Certificate of Training

Manish Kumar

from MM Institute Of Computer Technology & Business Management has successfully completed a 6-week online training on **React** from 15th June 2023 to 27th July 2023. The training consisted of Introduction, Tic Tac Toe Game, Box Office App, Chat Application, Custom Backend, and Assignment & Summary modules.
Manish is a top performer in the training.
We wish Manish all the best for future endeavours.

Sarvesh Agarwal
FOUNDER & CEO, INTERNSHALA

Date of certification: 2023-07-27 Certificate no. : 104i43y8jl9

For certificate authentication, please visit https://trainings.internshala.com/verify_certificate

ACKNOWLEDGEMENT

First of all I feel great pleasure in acknowledging my deepest gratitude to my revered guide and mentor, **Ms. Nikki Sorout, Assistant Professor,** Maharishi Markandeshwar Institute of Computer Technology and Business Management, under whose firm guidance, motivation and vigilant supervision we are accomplishing my work. She infused into us the enthusiasm to work on this topic. Her tolerant nature accepted our shortcomings and she synergized his impeccable knowledge with our curiosity to learn into this fruitful result.

I sincerely thank **Dr. Sumit Mittal, Principal**, Maharishi Markandeshwar Institute of Computer Technology and Business Management, who is always very helpful and constructive. Last but not the least, we find it hard to express our gratefulness to the God in words for providing us his deepest blessing and giving us the opportunity in the form of life of a human being and for the warmth and kindness he has showered upon us by giving us life's best.

Manish Kumar

1322748

ABSTRACT

The React.js training program aimed to equip participants with the necessary skills to build modern web applications using this popular JavaScript library. Throughout the training, the focus was on understanding React's component-based architecture, virtual DOM, and state management, which are vital for creating responsive and scalable user interfaces. The chat application project served as a practical application of the learned concepts, providing hands-on experience in designing and implementing real-time communication features.

During the development of the **Chat Application**, several challenges were encountered, including handling real-time data synchronization between multiple users, optimizing performance to ensure smooth user experience, and implementing a user-friendly interface. By addressing these obstacles, valuable insights were gained into the best practices and solutions in React.js development. Additionally, the project provided opportunities to explore and integrate other relevant technologies, such as Firebase to enable real-time messaging functionality.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Definition
API	Application Program Interface
CLI	Command Line Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	Hyper Text Markup Language
ID	Identification
IDE	Integrated Development Environment
JS	Java Script
JSON	JavaScript Object Notation
JSX	Java Script XML
NOSQL	Non-Structured Query Language
NPM	Node Package Manager
UI	User Interface
XML	Extensible Markup Language

2. INTRODUCTION

In this chapter, the experience of work placement and over all tasks of the Chat App is discussed.

2.1. Organization Profile

Internshala is India's largest online platform for internships and trainings, founded in 2010. The organization's mission is to bridge the gap between classroom education and practical work experience, empowering students and recent graduates with hands-on learning opportunities. Internshala offers a diverse range of internships across various industries, connecting millions of students with meaningful work experiences. Additionally, the platform provides skill-based trainings and online courses, enabling individuals to acquire valuable skills in areas such as programming, digital marketing, and design. With a focus on innovation and social responsibility, Internshala continues to make a significant impact on the employability and professional growth of the youth, contributing to a skilled and competitive workforce in India. I t provides the following services:

1. Internships:

Internshala offers a wide range of internships catering to various fields such as engineering, marketing, content writing, design, finance, and more. These internships are available in both paid and unpaid formats, giving students the flexibility to choose opportunities that align with their interests and financial requirements.

2. Trainings:

Recognizing the importance of continuous learning, Internshala provides high-quality skill-based trainings across domains like programming, digital marketing, data science, graphic design, and entrepreneurship. These training programs are designed to impart practical knowledge and are conducted by industry experts.

3. Online Learning:

Apart from internships and trainings, Internshala also offers an extensive array of online courses, enabling students to learn at their own pace and convenience. These courses cover a broad spectrum of subjects, ranging from technical skills to soft skills and personal development.

2.2. Introduction of Project

The "Chat Application using React and Firebase" is a dynamic web-based platform designed to facilitate real-time communication between users through a secure and intuitive chat interface. The project aims to create a seamless experience where users can log in using their email IDs, select chat rooms based on their interests or needs, and engage in instant messaging with other participants. Leveraging the power of React, a popular JavaScript library for building user interfaces, and Firebase, a robust backend-as-a-service platform, the application ensures efficient data synchronization and persistence.

With the growing demand for online collaboration and communication, this project holds significant value in catering to the need for interactive platforms that enable people to connect and share ideas effortlessly. By integrating modern technologies, the Chat Application offers users a user-friendly and feature-rich environment for seamless conversation and the creation of new chat rooms, encouraging community engagement. From a development standpoint, this project serves as an excellent learning opportunity to understand the implementation of real-time messaging and data storage using Firebase, making it a valuable asset for developers looking to enhance their skills in web application development.

2.3. Problem Definition

In today's fast-paced digital age, effective communication and collaboration play a crucial role in connecting people from diverse backgrounds and

fostering knowledge exchange. However, traditional communication channels often lack the speed and interactivity demanded by modern users. This poses a challenge for individuals seeking real-time interactions and seamless conversations within specific interest groups or communities.

The "Chat Application using React and Firebase" project addresses this problem by offering a robust and user-friendly solution for instant messaging and community engagement. The primary issue this project aims to solve is the absence of an efficient and secure platform that enables users to log in with their email IDs, choose from a selection of chat rooms, and engage in real-time communication with others. Existing chat applications might lack the desired level of responsiveness, data persistence, or scalability, inhibiting smooth communication and seamless conversation experiences.

Additionally, creating new chat rooms tailored to specific topics or interests might be cumbersome or restricted in some applications. This project strives to overcome these limitations by providing users with the freedom to create new chat rooms, promoting inclusivity and encouraging discussions on diverse subjects. By leveraging the power of React for a dynamic user interface and Firebase for secure data storage and synchronization, this application seeks to deliver a reliable, efficient, and interactive solution.

2.4. Limitations of existing system

Before the development of the "Chat Application using React and Firebase," existing chat applications in the market may have exhibited several limitations that hindered the user experience and efficiency of real-time communication. Some common limitations of existing chat applications include:

1. Limited Real-time Updates: Some chat applications provided real-time messaging, but the updates were not instant or seamless. Users might

experience delays in message delivery or inconsistent synchronization across devices.

2. Lack of User Authentication: Certain chat applications lacked robust user authentication systems, leading to privacy and security concerns. Users might not have the assurance of communicating only with verified individuals.

3. Complex User Interfaces: Many existing chat applications had cluttered or confusing user interfaces, making it challenging for users to navigate and use the application effectively.

4. No Customization Options: Some chat applications offered limited or no customization options, leaving users with a standardized interface and restricting their ability to personalize their chat experience.

5. Inefficient User Management: User management and moderation capabilities were sometimes lacking in existing chat applications, making it difficult to manage users, control access, and handle abusive behavior.

Overcoming Limitations in the "Chat Application using React and Firebase

1. Real-time Messaging with Firebase
2. Firebase Authentication
3. User-Friendly Interface
4. Customization with SCSS Styling
5. Effective User Management

By addressing these limitations, the "Chat Application using React and Firebase" sought to offer an improved and user-centric real-time communication experience.

2.5. Objective of the Project

The primary objective of the "Chat Application using React and Firebase" project is to develop a feature-rich web-based chat application that enables seamless and real-time communication among users. The project aims to achieve the following specific objectives:

1. User-friendly Interface: Design and implement an intuitive and user-friendly interface that allows users to easily log in using their email IDs, select chat rooms of interest, and engage in real-time messaging without any technical barriers.

2. Real-time Communication: Enable instant messaging and real-time updates within the selected chat rooms, ensuring that users can exchange messages with minimal latency, enhancing the overall communication experience.

3. Secure User Authentication: Implement a robust and secure user authentication system using Firebase, ensuring that only authorized users can access the application and participate in chat room discussions.

4. Dynamic Chat Rooms: Allow users to create new chat rooms based on their interests or topics of discussion, fostering community engagement and encouraging diverse interactions.

5. Data Synchronization and Persistence: Utilize Firebase's database capabilities to ensure seamless data synchronization across multiple users and devices, ensuring that chat messages and room details persist and remain up to date.

6. Scalability and Performance: Design the application with scalability in mind, ensuring it can handle a growing number of users and chat rooms without compromising performance or user experience.

3. SYSTEM ANALYSIS

This chapter explains the system analysis of the Chat App in details.

3.1. Feasibility study

A feasibility study is a crucial step in evaluating the practicality and viability of a project. For the "Chat Application using React and Firebase," we will assess the technical, economic, operational, and scheduling aspects to determine whether the project is feasible and can be successfully executed. Let's explore each aspect in detail:

1. Technical Feasibility:

This aspect evaluates the technical capabilities required to develop the chat application. The technical feasibility of the project is determined by:

a. Technology Stack: React and Firebase are widely recognized and robust technologies for building dynamic web applications and real-time database synchronization. Both technologies have a strong developer community, extensive documentation, and support, making them highly feasible for the project's implementation.

b. Integration and Compatibility: Assessing the compatibility of React and Firebase and ensuring seamless integration is essential for the successful development of the chat application. Fortunately, React and Firebase have a proven track record of compatibility, making the technical feasibility favorable.

2. Economic Feasibility:

Economic feasibility analyzes the cost and financial viability of the project.

Factors considered are:

a. Development Cost: The cost associated with hiring skilled developers with expertise in React and Firebase, software licenses, and any third-party libraries or tools required for development.

b. Operating Cost: Ongoing expenses related to hosting the application, maintenance, and infrastructure.

c. Return on Investment (ROI): Assessing the potential benefits and financial returns derived from the application, such as revenue from advertisements or premium features (if applicable).

3. Operational Feasibility:

Operational feasibility assesses whether the proposed chat application aligns with the organization's processes, resources, and capabilities:

a. User Adoption: Analyzing whether the target audience would readily adopt the application, considering factors such as ease of use, user interface design, and familiarity with similar chat applications.

b. Support and Maintenance: Evaluating the organization's capacity to provide ongoing support, updates, and maintenance for the application to ensure its continuous functionality and user satisfaction.

4. Scheduling Feasibility:

Scheduling feasibility evaluates the project timeline and whether the proposed development timeframe is achievable:

a. Development Timeframe: Estimating the time required for design, development, testing, and deployment phases.

b. Resource Availability: Ensuring that there are adequate resources, including skilled developers, to meet the project's timeline.

Based on the comprehensive feasibility study, it is evident that the "Chat Application using React and Firebase" project is technically feasible due to the compatibility and strong community support of the chosen technologies. From an economic standpoint, the potential return on investment and operational feasibility are favorable, given the widespread demand for real-time communication applications. Lastly, with proper planning and resource allocation, the project's development is feasible within a reasonable timeframe. As such, the proposed chat application project holds promise for successful implementation, contributing to a dynamic and interactive platform for users to engage in real-time conversations and community interactions.

3.2. Hardware and Software requirement

Hardware Requirements:

Since the "Chat Application using React and Firebase" is a web-based project, the hardware requirements are relatively minimal. Here are the essential hardware components needed for development and deployment:

1. Computer: A desktop or laptop computer with sufficient processing power and memory to handle web development tasks efficiently.

2. Internet Connection: A stable internet connection is necessary for accessing the development environment, collaborating with team members (if applicable), and deploying the application.

3. Storage: Adequate storage space to store project files, code, and other development-related data.

Software Requirements:

The project relies on specific software tools and technologies for development and deployment. Below are the essential software requirements for the "Chat Application using React and Firebase":

1. Code Editor: A code editor is required for writing and editing the application's source code. Popular code editors for web development include Visual Studio Code, Sublime Text.

2. Node.js and npm (node package manager): Node.js is essential for running JavaScript on the server-side and npm (Node Package Manager) is used to manage project dependencies and packages. Ensure you have the latest stable versions installed on your system.

3. React: The project utilizes React, a JavaScript library for building user interfaces. Install React and React-related dependencies using npm.

4. Firebase: Firebase provides the database and authentication services for the application. Create a Firebase account and set up a new Firebase project to access the necessary credentials.

5. Web Browser: For testing and debugging, a web browser is required. Google Chrome, Mozilla Firefox, or any modern browser with developer tools support is recommended.

6. Git (Optional): Git version control can be used to manage the application's source code and facilitate collaborative development if multiple developers are involved.

7. Operating System: The project can be developed and deployed on various operating systems, including Windows, macOS, or Linux.

4. DEVELOPMENT ENVIRONMENT

This chapter provides an overview of the software tools, libraries, and frameworks used during the development process. It details the setup and configuration required to create, test, and deploy the chat application. This chapter serves as a guide for developers and stakeholders to understand the technical aspects of the project and replicate the environment for further development or maintenance**.**

4.1. Introduction of used Technology

1. HTML: HTML is a programming language which was developed by Berners-Lee in 1980 but standardized in 1995. HTML stands for Hyper Text Mark-up Language. It is one of the most popular languages on the planet used for developing web pages and websites. HTML is used for the structure of web documents i.e. headlines, paragraphs and images etc. HTML is known as the basics of all programming language in the globe. HTML is highly used in other programming languages such as JavaScript and CSS for the display attributes. It consists of opening and closing tags as <html></html>with the file extension .html

Advantages of HTML:

1. HTML is easy to learn and use without any prerequisite's knowledge in programming.
2. HTML has no license fees.
3. HTML is supported by all browsers on the web and supported by all platforms.
4. HTML is lightweight code so that fast to download.
5. HTML is very easy to edit in any text editor such as Notepad or Notepad ++.
6. HTML is the most search-friendly programming on the globe.

2. SCSS: SCSS, which stands for "Sassy CSS," is a popular extension of CSS that adds powerful features and capabilities to traditional CSS syntax. SCSS is a pre-processor for CSS, meaning it is compiled into standard CSS before being deployed to the web browser. This leads to cleaner and more organized code

Advantages of SCSS:

1. Variables: SCSS allows the use of variables, which is one of its most significant advantages. Developers can define variables and store commonly used values, such as colours, font sizes, or spacing, in a single place. This promotes consistency and makes it easier to update the entire project by changing just one variable.

2. Nesting: SCSS supports nesting of CSS rules within one another, which mirrors the HTML structure. This leads to cleaner and more organized code, as it reflects the visual hierarchy of the elements. Nesting reduces repetition and enhances readability.

3. Mixins: Mixins in SCSS are reusable blocks of CSS code that can be included in multiple places. They are beneficial for creating consistent styles for elements with similar properties. Mixins help avoid code duplication and improve code maintainability.

4. Partials: SCSS allows developers to split their style sheets into smaller, modular files called partials. This is useful for organizing the CSS codebase into manageable sections, making it easier to navigate and maintain, especially in large projects.

5. Inheritance: SCSS supports CSS inheritance using the `@extend` directive. This allows one CSS class to inherit the styles of another, leading to more concise and DRY (Don't Repeat Yourself) code.

6. Operators and Calculations: SCSS supports mathematical operations and calculations within the CSS code. This feature can be handy for dynamically adjusting styles based on specific conditions or user interactions.

3. Visual Studio Code: Visual Studio Code is the part of visual studio family which is developed by Microsoft in November 2015. It is based-on Electron framework which is used for Node.js (node java script). It is written in Typescript, JavaScript and CSS

Features of VS Code:

1. It is open-source and Freeware text editor for private and commercial purposes
2. It is cross-platform source code editor debugger
3. It supports many different programming languages while only proper installation of the extension is required for React JS, Java, JavaScript, C++, C#, Python etc.
4. GitHub is built-in
5. Products availability as it explains the definition and show opening and closing of the brackets etc.
6. It provided unique customizations
7. Provides Visual Studio Keymap extensions for using Key binding
8. It provides in the portable mode that means it keeps data and settings in the same location of installation is possible even on a USB drive
9. It is available in many different language services
10. It is also available in Remote Development mode

4. React JS: React JS was created by Jordan Walke and deployed first at Facebook in May 2013. It is based on open-source JavaScript library. In other words, React JS is a great platform which is developed for the front-end mobile and web apps. It is also known as React JavaScript library, which is used for the developing of User Interfaces (UI). It consists of independent, isolated and reusable components which are pieces of UI.

Every React JS project has a main root component called as Application (App). Application root component contains other related child components. React JS application comprises of React element and React component. React component is the function or class which renders React element or HTML element.

Key Features of React JS:

1. Easy to learn: It is easy to learn as compare to other front-end frame works technologies
2. JSX: JSX is shorthand of JavaScript XML. React JS uses JSX instead of HTML which enables the user to read and write the components easily and clearly.
3. Speed: It allows the developers reuse the individual components on both client side and server-side without effects on the main application.
4. Flexibility: React JS code is easy to maintain and flexible due to its framework structure as compared to other front-end framework technologies
5. Performance: React JS uses virtual DOM and server-side rending thus it makes complex web apps runs on high speed
6. Code stability: It follows unidirectional data flow or downward data flow which means parent structure unaffected by any modifications in its child structure
7. Integration: React code easy to integrate with other frameworks like Angular
8. SEO- friendly: SEO stands for Search Engine Optimization. React offers less page load time and fast rendering speed.
9. It is easy to test UI cases with React code.
10. React renders at client side while SEOs do not use the data.

Virtual DOM and Real DOM:

DOM stands for Document Object Model, which is programming language platform interface for HTML and XML documents. It is a tree-like structure

which enables dynamically accessing and managing the elements. It has three main types:

1 HTML DOM: Standard model for HTML documents
2 XML DOM: Standard model for XML documents
3 Core DOM: Standard model for all types of documents
In addition to, DOM has two types Real DOM and Virtual DOM. React JS uses Virtual DOM

Real DOM

1. Its manipulation or updates is slower and costly.
2. It is heavy weight code.
3. It can update the HTML in the browser

Virtual DOM

1. It is not created by React, but it is used as for free.
2. It is a copy of Real DOM.
3. Its manipulation or updates is faster and easier.
4. It is light weight code
5. It cannot update the HTML directly in the browser

Life Cycle of React Components:

In life, we humans as well as tree have a life cycle like we are born, grow and finally we die. It is the same phenomena in software application´s life cycle, which takes birth as initialization, it grows by updating and changes occurs during the different stages by mounting and unmounting. React JS has four life cycle classifications as below:
1. Initialization: This is first and most important phase in which developer setup or constructs the component with the given props (properties) and default state.
2. Mounting: In this second phase, after creating state and props by developer. React JS component is ready to mount in the browser DOM.

3. Update: This is third stage, once the component gets added in browser DOM. React JS components updates it when a state or prop changes in the application.
4. Unmounting: This is fourth and final phase of the life cycle of React JS component in which the component is removed from browser DOM when it is not needed anymore.

5. Firebase: Firebase is the Firebase Cloud messaging as Google Cloud Messaging it software cross platform. It is also known was developed by Andrew Lee and James Tamplin 26 in 2011. Firebase provides cloud service, server and real-time backend server. API for the clients also helps to develop high quality applications. Most importantly, Firebase is a NoSQL database in which data is stored in a cloud. When full-fledged application is built, you have to rely on the backend service as well. Therefore, Firebase is a good choice because it saves time and allows you to focus on development. It stores the data in JSON file or JSON objects and we do not need to write any server codes, but it is ready-made API for developing and implementation of applications.
Firebase hosts the services to cloud functions which provide three different types of cloud services as below: 1. Cloud storage which stores data.
2. AutoML vision is the Google´s machine learning model which is used for building the image recognition.
3. Cloud Firestore is the Google´s NoSQL cloud platform which provides database for mobile, web and server development.

Features of Firebase:

1. Real-time Database: It synchronizes the sender´s data to Firebase which updates instantly, and client receives the updated data on real-time on any connected device.

2. File storage: Firebase enables the file storage directly from the client even in offline mode.

3. Authentication: Firebase provides its own pre-built or custom UI authentication or sign-in method email / password, phone number, Google id, Facebook id, yahoo or Apple id.

4. Hosting service: Firebase offers the implementation of static websites and single page applications which is built with HTML, CSS and JavaScript. It uses HTTPS and SSL protocols for the security of files and data delivery. It does not require Content Delivery Network (CDN), which is built in function in Firebase.

5. Free of cost: Firebase does not require payment for providing databases service to client to some extent. On the other hand, the client has to pay if the database memory is not enough for the specific services.

6. Deployment Platform: Firebase Hosting: For hosting the chat application, Firebase Hosting was chosen as the deployment platform. Firebase Hosting is a fast and reliable hosting service provided by Google Firebase, specifically designed for deploying and serving web applications. Its seamless integration with Firebase Real time Database and Firebase Authentication made it an ideal choice for the project. Here's a brief explanation of the deployment process using Firebase Hosting:

1. Setting up Firebase Project: To begin, a new Firebase project was created on the Firebase Console. The project settings were configured, including setting up the project's name, region, and other necessary configurations.

2. Firebase CLI Installation: The Firebase Command Line Interface (CLI) was installed on the development machine. The Firebase CLI allows developers to manage and deploy Firebase projects from the terminal or command prompt.

3. Building the Chat Application: Before deployment, the chat application was thoroughly tested and optimized for production. The necessary build

commands were executed to generate the production-ready build of the application.

4. Firebase Initialization: Using the Firebase CLI, the chat application was initialized with the Firebase project. This step establishes the connection between the local project directory and the Firebase project.

5. Deploying to Firebase Hosting: With the application initialized, the final step was to deploy the chat application to Firebase Hosting. The Firebase CLI's simple deployment command was used to upload the application files and assets to Firebase servers.

7. Version Control with Git: Git is a widely used version control system that allows developers to manage and track changes to codebase efficiently. It provides a way to collaborate on projects, maintain a history of changes, and revert to previous versions if needed. Here's how Git was utilized in the development of the chat application:

1. Repository Initialization: The version control process began by initializing a Git repository in the project directory. This created a local repository, enabling developers to track changes within the project.

2. Branching: Git allows for the creation of branches, which are separate lines of development. For the chat application, branches were used to work on different features or bug fixes independently. This way, the main codebase (usually the 'master' branch) remained stable while new features were developed in their respective branches.

3. Committing Changes: As changes were made to codebase, developers used Git to create commits. A commit represents a snapshot of the project at a specific point in time. Each commit includes a concise and descriptive message detailing the changes made.

4. Pushing and Pulling: To collaborate with team members, developers pushed their local commits to a remote repository hosted on a version control platform like GitHub. This made the changes available to other team members and ensured everyone was working with the most up-to-date code.

5. Merging and Pull Requests: Once a feature or bug fix was complete in a separate branch, developers created a pull request to propose merging the changes into the main branch. Code reviews were conducted, and any conflicts were resolved before merging the changes into the 'master' branch.

6. Version History and Tagging: Git maintains a detailed version history, allowing developers to view the evolution of the project over time. For significant milestones or releases, developers used tagging to label specific commits as version releases (e.g., v1.0.0).

7. Branch Management: Throughout the project's development, branches were managed efficiently. Unused or merged branches were deleted to keep the repository organized and prevent clutter.

8. Handling Conflicts: Occasionally, conflicts could arise when merging branches with different changes. Developers used Git's tools to resolve conflicts and ensure the final codebase was cohesive and functional.

5. SYSTEM DESIGN

In this chapter, the Chat App system design is explained in detail.

5.1. Application Architecture

Application architecture refers to the high-level structure and organization of a software application, defining how its components interact with each other and how data flows throughout the system. In the case of the "Chat Application using React and Firebase," the application architecture follows a client-server model. Let's explain each component of the application architecture in detail:

1. Client-Side (Front-end): The client-side, also known as the front-end, represents the user interface and user-facing functionalities of the chat application. It is responsible for rendering the application in the user's web browser and handling user interactions. In this project, the front-end is built using React, a popular JavaScript library for building user interfaces.

Key Characteristics and Responsibilities of the Front-end:

- Rendering the user interface: React components render the chat rooms, messages, user input forms, and other visual elements.

- Handling user interactions: React components manage user actions, such as sending messages, creating new chat rooms, and logging in or registering users.

- Real-time updates: The front-end communicates with the back-end (server) to receive real-time updates of new messages and chat room changes.

2. Server-Side (Back-end): The server-side, also known as the back-end, manages the data storage, business logic, and user authentication for the chat application. In this project, Firebase provides the back-end infrastructure, offering real-time database and user authentication services.

Key Characteristics and Responsibilities of the Back-end:

- Data storage and synchronization: Firebase Real time Database is used to store chat messages and chat room information. It ensures that all connected clients receive real-time updates whenever new messages are sent or chat rooms are created.

- User authentication: Firebase Authentication handles user login, registration, and account management. It ensures that only authenticated users can access the chat application and participate in chat room discussions

- Business logic: While the chat application may not require extensive business logic, the back-end may still handle tasks like validating user input, ensuring data integrity, and managing user permissions.

3. Client-Server Interaction: The client-side and server-side interact through API calls and real-time communication. When a user sends a message or creates a new chat room, the front-end sends the necessary data to the back-end through API calls provided by Firebase. Firebase real time Database updates the database with the new information and sends real-time updates to all connected clients, including the sender, ensuring that everyone in the chat room receives the new message instantly.

4. Data Flow: In the chat application, the data flow primarily occurs from the front-end to the back-end when a user sends a message or creates a chat room. The data is transmitted to Firebase real-time Database, which stores and synchronizes the data across all connected clients. Additionally, during

user authentication, the front-end sends user credentials to Firebase Authentication for verification and user account management.

5. Scalability and Performance: The client-server architecture and Firebase's real-time capabilities ensure good scalability and performance for the chat application. Firebase handles the data synchronization and real-time updates, allowing the application to handle multiple users and chat rooms simultaneously without significant performance bottlenecks.

Overall, the client-server architecture of the "Chat Application using React and Firebase" ensures seamless real-time communication and data synchronization between users, providing a dynamic and interactive platform for engaging in chat room discussions.

5.2. Modular Description

This topic explains the functionality of all the components of Chat App and Firebase backend.

Front-end Modules: The front-end of the "Chat Application using React and Firebase" consists of several key modules, each responsible for specific functionalities of the user interface and user interactions. Let's explain each front-end module in detail:

1. User Authentication Module: The User Authentication module manages user login, registration, and user account management. It allows users to authenticate themselves to access the chat application and participate in chat room discussions. This module interfaces with Firebase Authentication, which handles user authentication securely.

Key Features and Responsibilities

- User Login: Allows users to log in using their email address and password. The module communicates with Firebase Authentication to verify user credentials and grant access to the application.

- User Account Management**:** Provides options for users to manage their accounts, such as changing their names, updating profile information, or changing their avatar (profile picture).

2. Chat Interface Module: The Chat Interface module is responsible for rendering the chat room view, displaying messages, and handling user interactions related to messaging. It facilitates real-time communication and updates with Firebase Realtime Database, allowing users to engage in seamless conversations.

Key Features and Responsibilities:

- Message Display: Renders and displays chat messages within the chat room. Messages are retrieved from Firebase Realtime Database and presented in chronological order.

- Message Input: Provides a text input field for users to type and send new messages. Users can enter text and send messages by interacting with the message input interface.
- Real-time Updates: Facilitates real-time updates of chat messages. When a new message is sent, the module communicates with Firebase Realtime Database to synchronize the message across all connected clients in the chat room.

3. Create Chat Room Module: The Create Chat Room module allows users to create new chat rooms with specific names and topics. When users create chat rooms, this module communicates with Firebase Realtime Database to store the new chat room information.

Key Features and Responsibilities:

- Chat Room Creation Form: Presents a form with fields for users to input the name and topic of the new chat room they wish to create.

- Firebase Integration: When users submit the chat room creation form, the module interacts with Firebase Realtime Database to store the new chat room's details, making it available for other users to join.

4. Sidebar Module: The Sidebar module provides a list of available chat rooms for users to select from. It dynamically updates with Firebase Realtime Database to reflect the latest chat rooms in real-time.

Key Features and Responsibilities:

- List of Chat Rooms: Displays a list of chat rooms available for users to join. The module retrieves the chat room information from Firebase Realtime Database.

- Real-time Updates: Listens to changes in chat room data in Firebase Realtime Database, ensuring that the sidebar automatically updates whenever a new chat room is created or removed.

By implementing these front-end modules, the "Chat Application using React and Firebase" offers users a seamless and interactive experience for joining chat rooms, sending messages, and managing their accounts, making it a dynamic and user-friendly platform for real-time communication and collaboration.

Back-end Modules: The back-end of the "Chat Application using React and Firebase" leverages Firebase services to manage data storage, real-time updates, and user authentication. Let's explain each back-end module in detail:

1. Firebase Realtime Database Module: The Firebase Realtime Database module handles the storage and retrieval of chat messages and chat room information. It is a NoSQL cloud database that enables real-time data synchronization among connected clients, ensuring that all users in a chat room receive instant updates when new messages are sent or chat rooms are created.

Key Features and Responsibilities:

- Chat Message Storage: The module stores chat messages, associating them with the corresponding chat room. Each message is stored with its sender's details, timestamp, and content, forming a chronological message history.

- Chat Room Information: The module manages information about each chat room, including its name, topic, creator, creation timestamp, and other relevant details.

- Real-time Synchronization: The Firebase Realtime Database ensures that any changes to chat messages or chat room information are immediately synchronized and propagated to all connected clients. This enables real-time communication and updates within chat rooms.

2. Firebase Authentication Module: The Firebase Authentication module handles user authentication, allowing users to sign up, log in, and manage their accounts securely. It ensures that only authenticated users can access the chat application and participate in chat room discussions.

Key Features and Responsibilities:

- User Login: Users can log in to the application using their email and password. The module verifies user credentials with Firebase Authentication before granting access.

- User Account Management: The module provides features for users to manage their accounts, including options to change passwords, update profile information, and handle account recovery through password reset mechanisms.

- Authorization: Firebase Authentication enables the application to authorize users based on their authentication status, ensuring that only authenticated users can access certain application features.

  By utilizing these back-end modules powered by Firebase, the "Chat Application using React and Firebase" ensures secure user authentication, real-time data synchronization, and efficient management of chat messages and chat room information.

6. IMPLEMENTATION AND TESTING

The implementations of the Chat App are discussed with the codes and test cases of the application are examined in this chapter.

6.1. Implementation with source code

The implementation of the "Chat Application using React and Firebase" involved a meticulous and iterative development process, transforming the project's design and specifications into a fully functional and user-friendly web application. Below are the comprehensive details of the key steps and milestones achieved during the implementation phase:

1. Front-end Development:

- **React Components**: The front-end was developed using React, a powerful JavaScript library for building user interfaces. Various components were created to handle different aspects of the application, such as User Authentication, Chat Interface, Create Chat Room, and Sidebar. Each component was carefully designed to ensure a seamless user experience and smooth interactions.

- **SCSS Styling:** To enhance the application's visual appeal and maintain consistent styling, SCSS (Sass) was used for styling the user interface. SCSS features like variables, nesting, and mixins were skillfully utilized to keep the codebase organized and manageable.

- **Real-time Messaging:** One of the application's core features was real-time messaging. By integrating Firebase Realtime Database with the front-end, users could exchange messages in chat rooms and receive updates instantly. React components were designed to handle real-time updates gracefully, providing users with a dynamic and engaging messaging experience.

- **User Authentication:** The front-end was seamlessly integrated with Firebase Authentication, enabling users to register, log in, and manage their accounts securely. User registration and login interfaces were designed with user-friendly forms, while account management features allowed users to update their profile information and change passwords conveniently.

2. Back-end Integration:

- **Firebase Realtime Database:** The back-end incorporated Firebase Realtime Database as the primary data storage solution for chat messages and chat room information. This NoSQL cloud database allowed for real-time synchronization, ensuring that chat messages and chat room data were instantly available to all users in the application.

- **Firebase Authentication:** For secure user authentication, Firebase Authentication was seamlessly integrated into the back-end. It provided robust authentication mechanisms, allowing only registered users to access the chat application and participate in chat room discussions.

3. Real-time Updates and Synchronization:

- Real-time updates were a critical aspect of the application. Whenever a user sent a message in a chat room, the back-end, powered by Firebase Realtime Database, immediately synchronized the new message with all connected clients in the same chat room. This ensured that all users received new messages instantaneously, creating a smooth and dynamic communication experience.

4. Error Handling and Validation:

- Comprehensive error handling mechanisms were implemented throughout the application to manage unexpected scenarios, such as network errors or

unsuccessful API calls. Error messages were appropriately displayed to users to provide clear feedback on potential issues.

- Input validation was enforced both at the front-end and back-end levels. It ensured data integrity and prevented the submission of invalid or empty messages and chat room names.

5. User Experience and Interface:

- The user interface was thoughtfully designed to offer an intuitive and user-centric experience. The chat application featured a clean and visually appealing design, making it easy for users to navigate and interact with various components.

- Features like message timestamps, user avatars, and chat room previews were thoughtfully integrated to enhance the user experience and provide valuable context during chat interactions.

6. Testing and Bug Fixing:

- To ensure the application's stability and reliability, comprehensive testing was conducted, including unit testing and manual testing. Various test cases were executed to identify and resolve any bugs or issues that arose during the development process.

- Extensive debugging and bug fixing were undertaken to deliver a polished and smooth-functioning application, ensuring a seamless user experience.

7. Deployment and Hosting:

- After successful development and testing, the chat application was deployed and hosted on Firebase Hosting. This ensured that the application was readily accessible to users from any device with an internet connection.

The implementation phase was marked by diligent efforts, attention to detail, and collaborative problem-solving. The integration of React and Firebase technologies facilitated seamless real-time messaging, user authentication, and dynamic data updates. The end result was a fully functional and user-friendly chat application that met all the project objectives, providing users with a delightful and interactive platform for real-time communication and collaboration

Source Code:

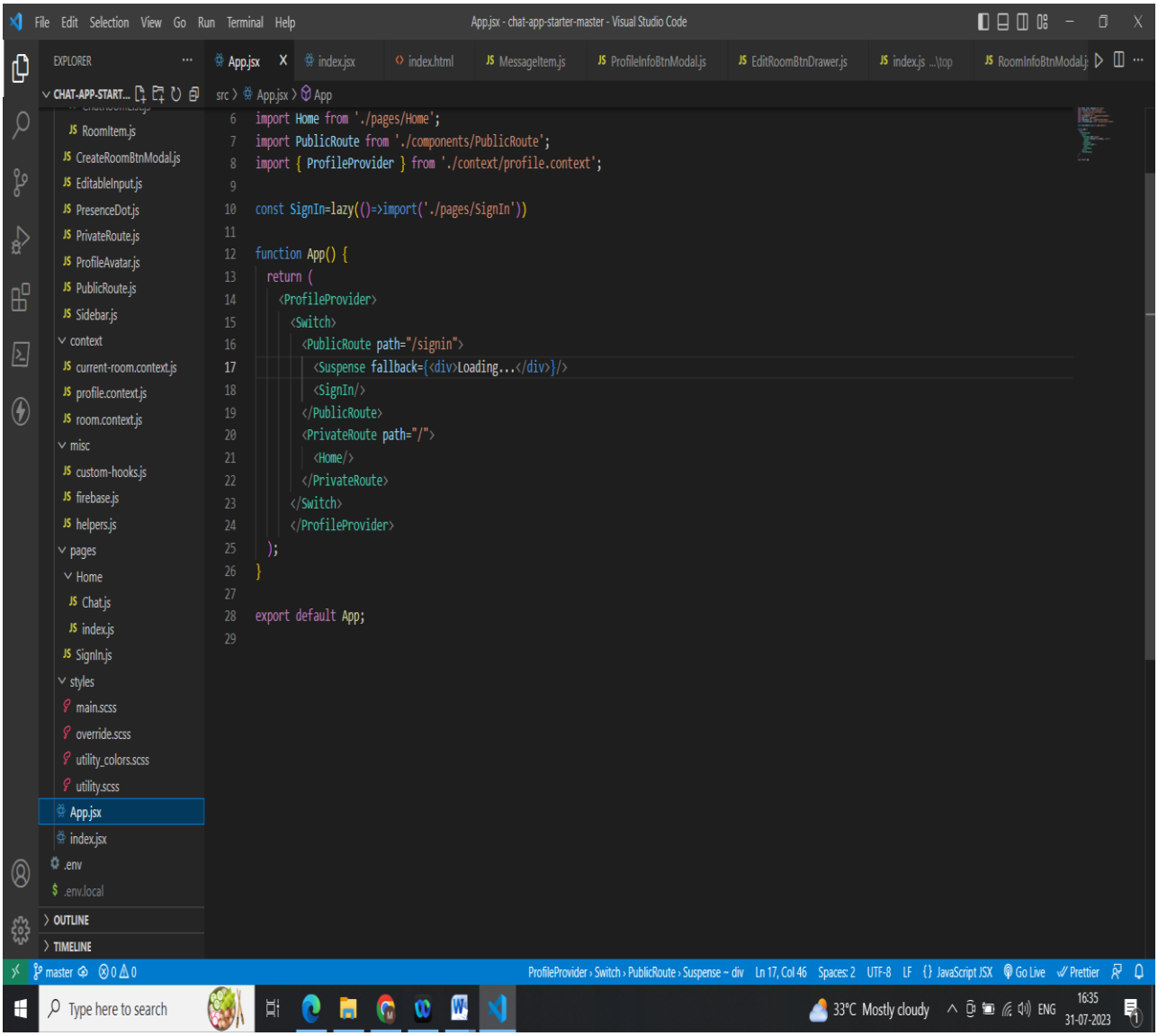

Figure 1. Screenshot of App.jsx

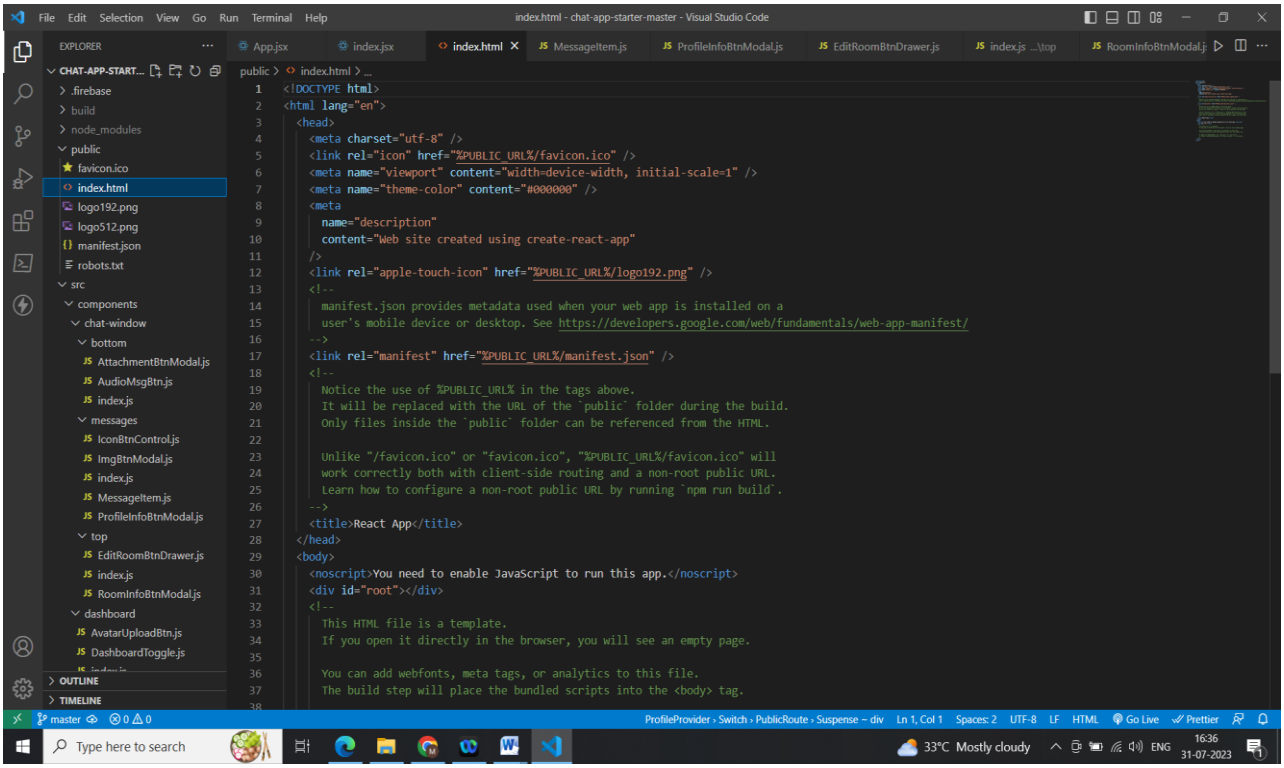

Figure 2. Screenshot of Index.html

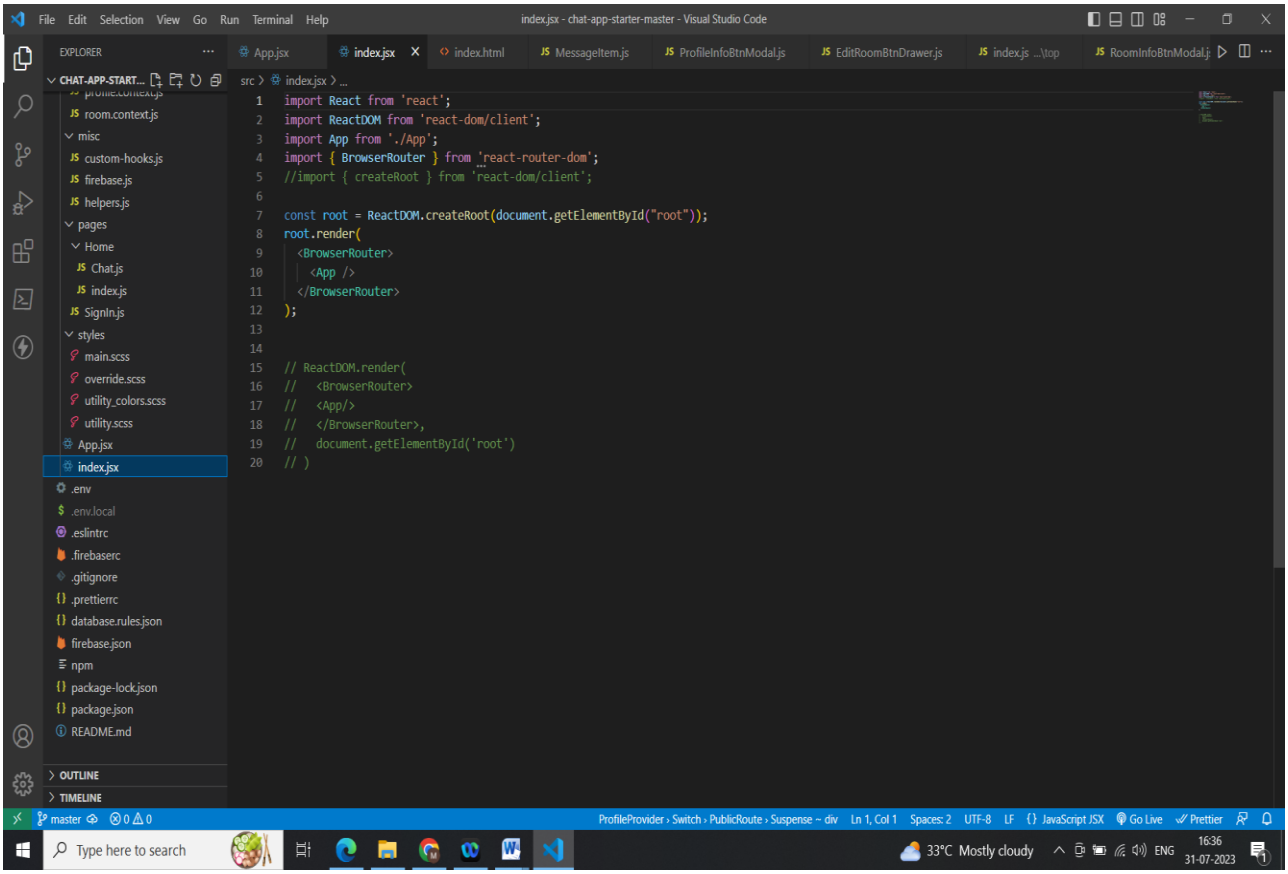

Figure 3. Screenshot of Index.js

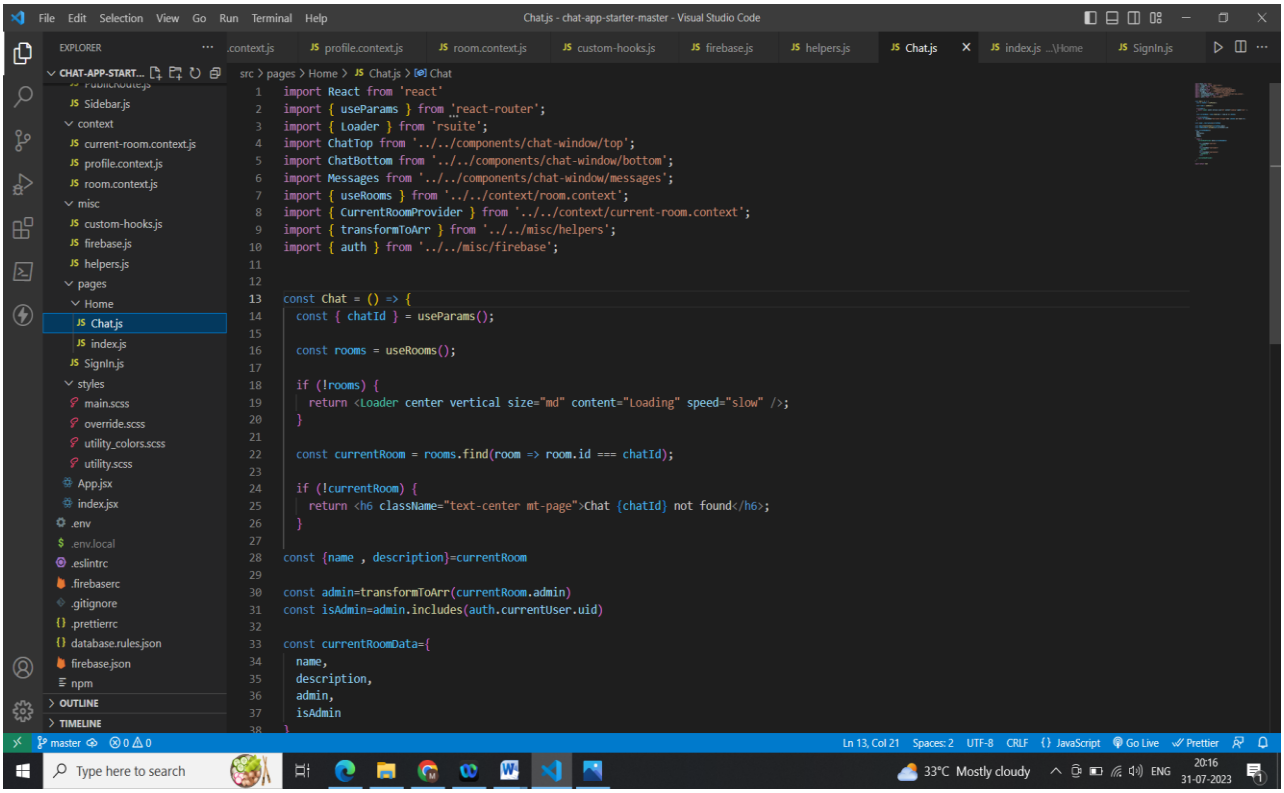

Figure 4. Screenshot Chat.js

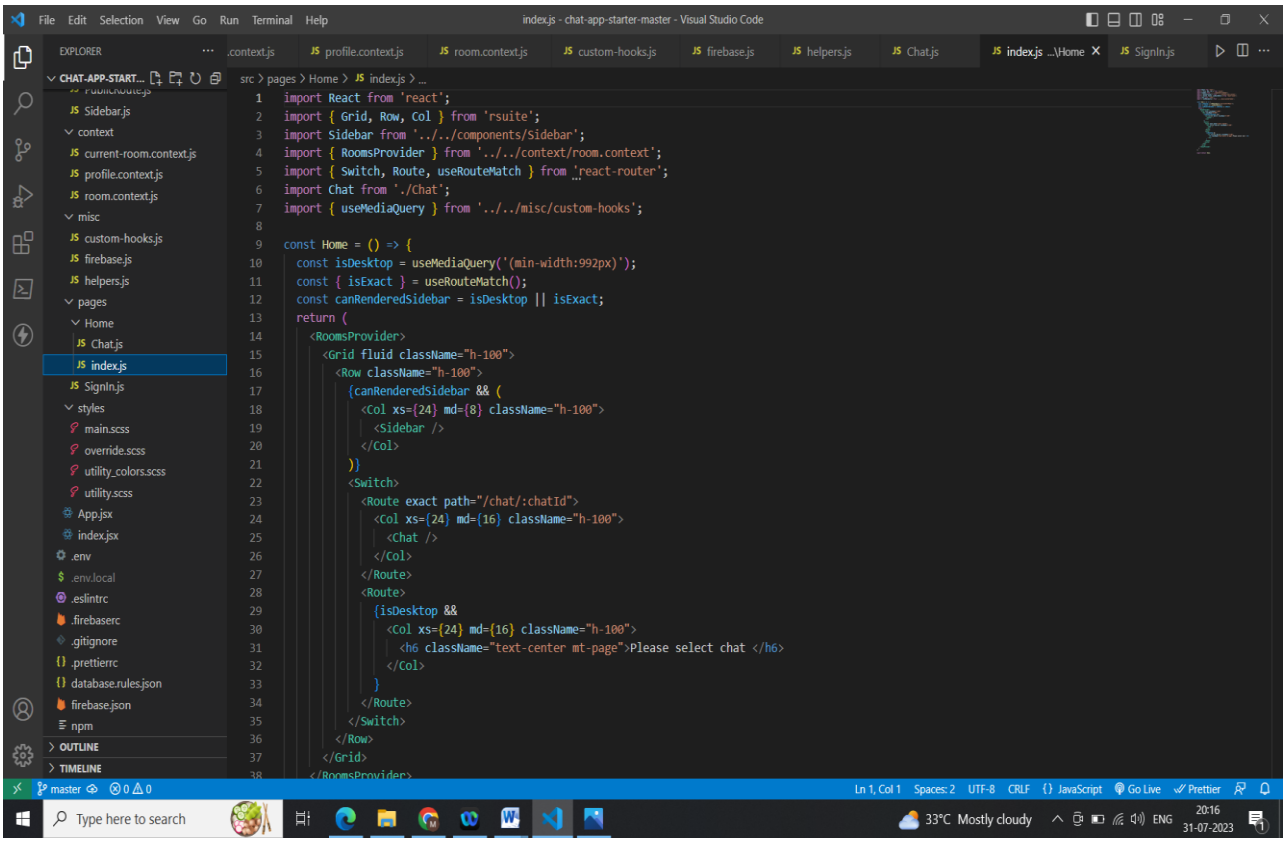

Figure 5. Screenshot of index.js

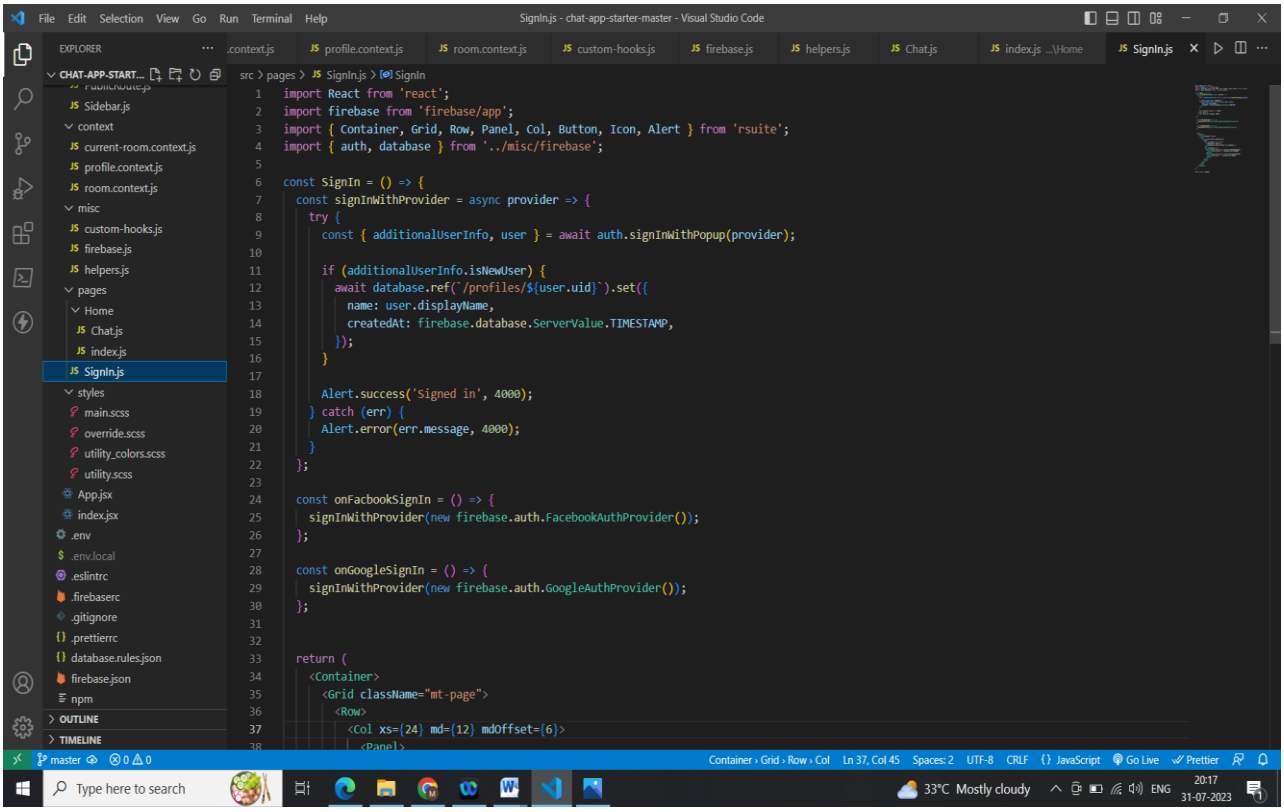

Figure 6. Screenshot of SignIn.js

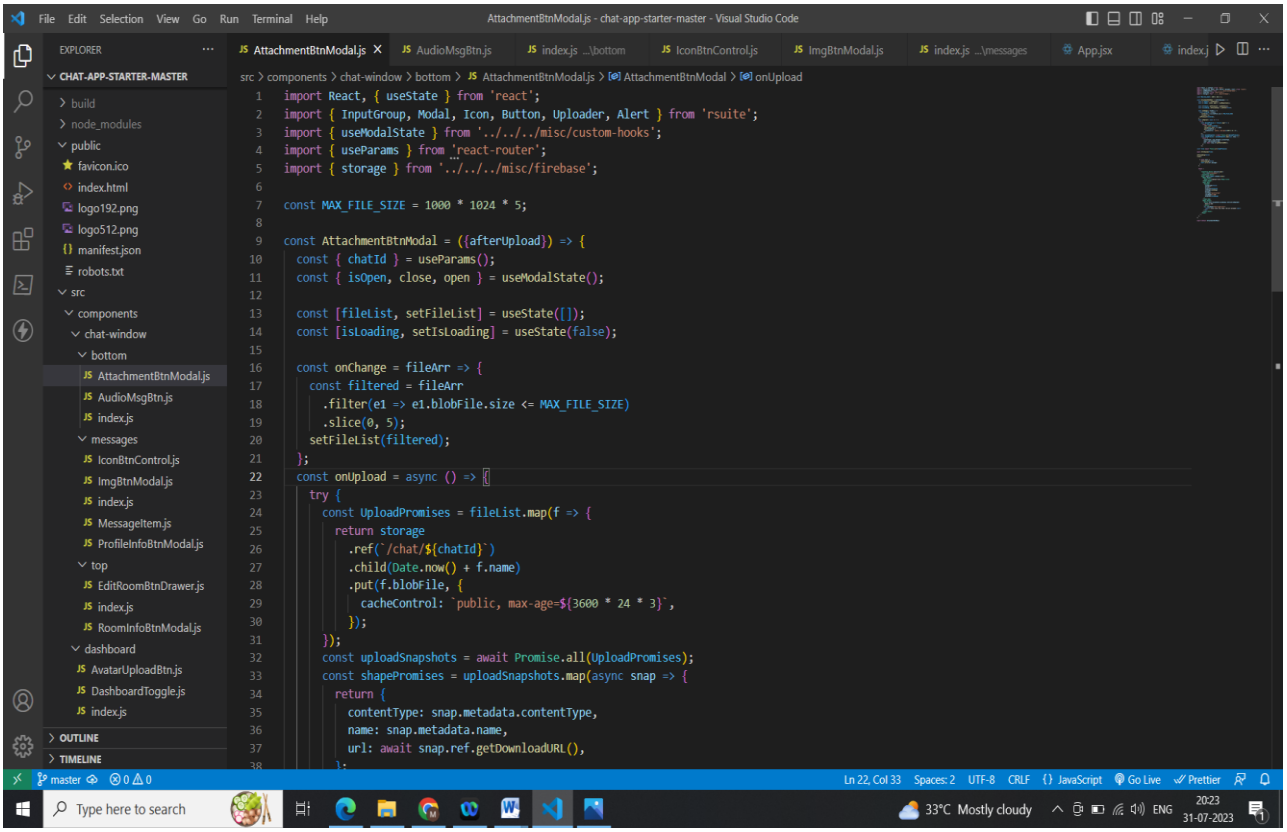

Figure 7. Screenshot of AttachBtnModal.js

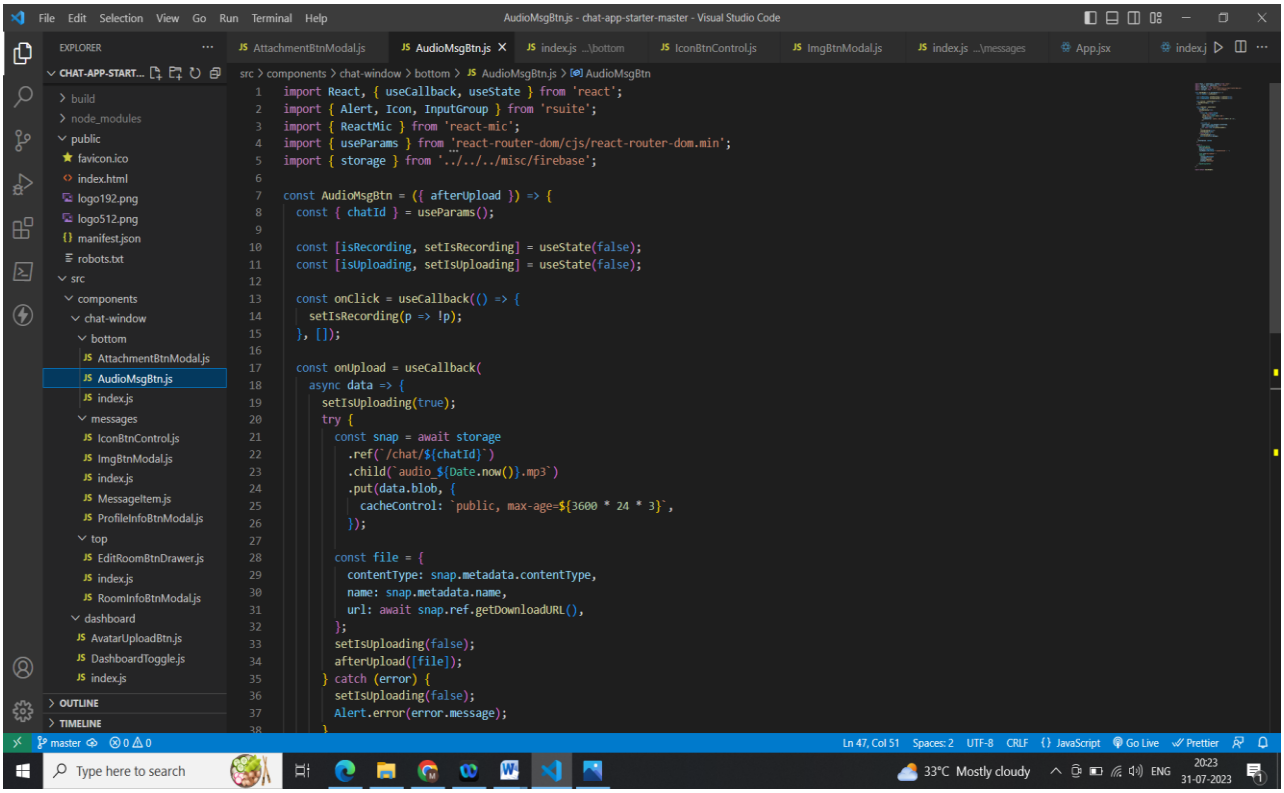

Figure 8. Screenshot of AudioMsgBtn.js

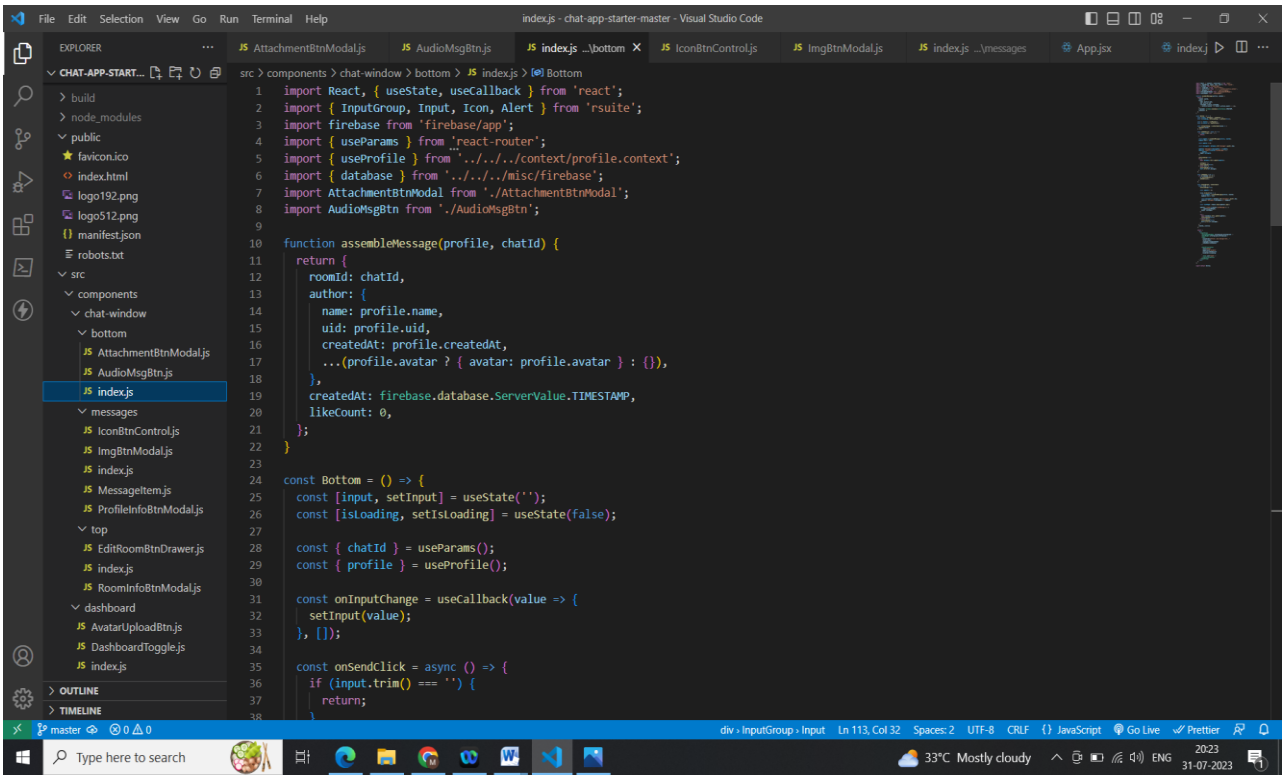

Figure 9. Screenshot of index.js

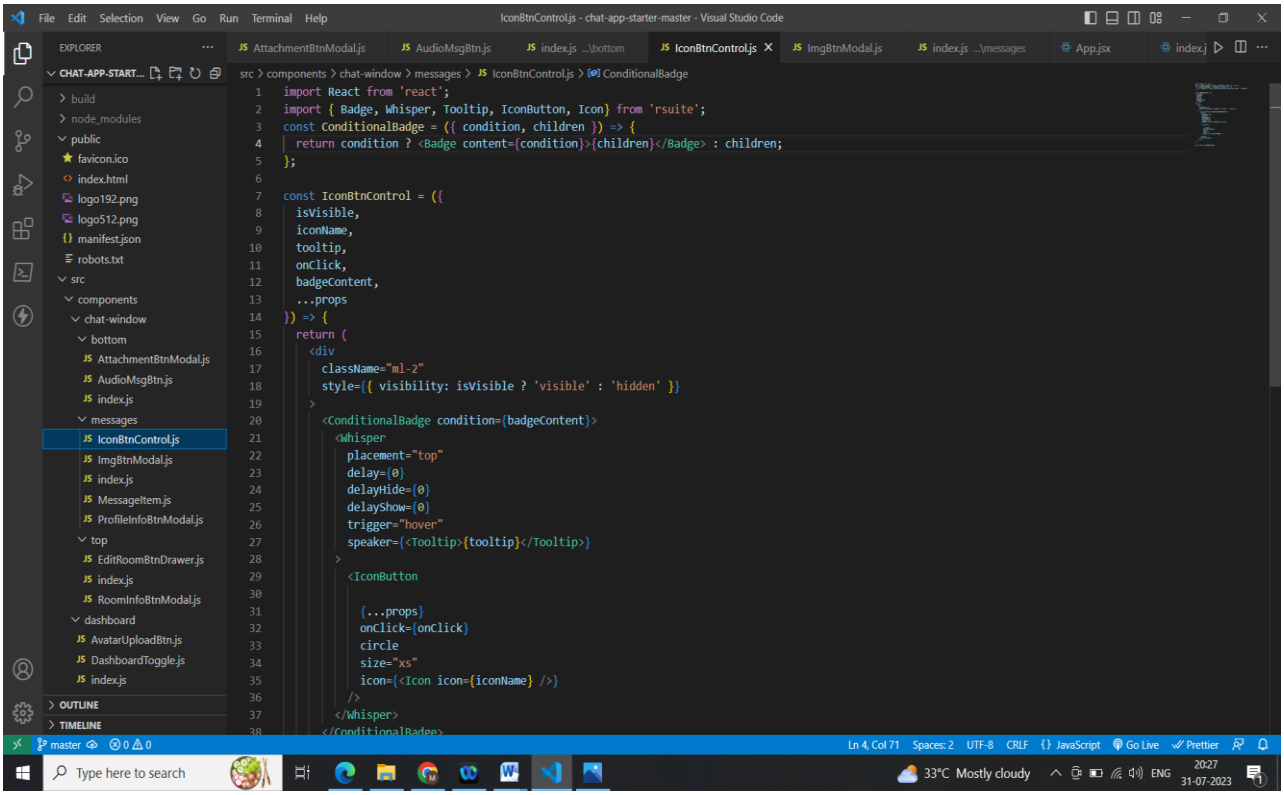

Figure 10. Screenshot of IconBtnControl.js

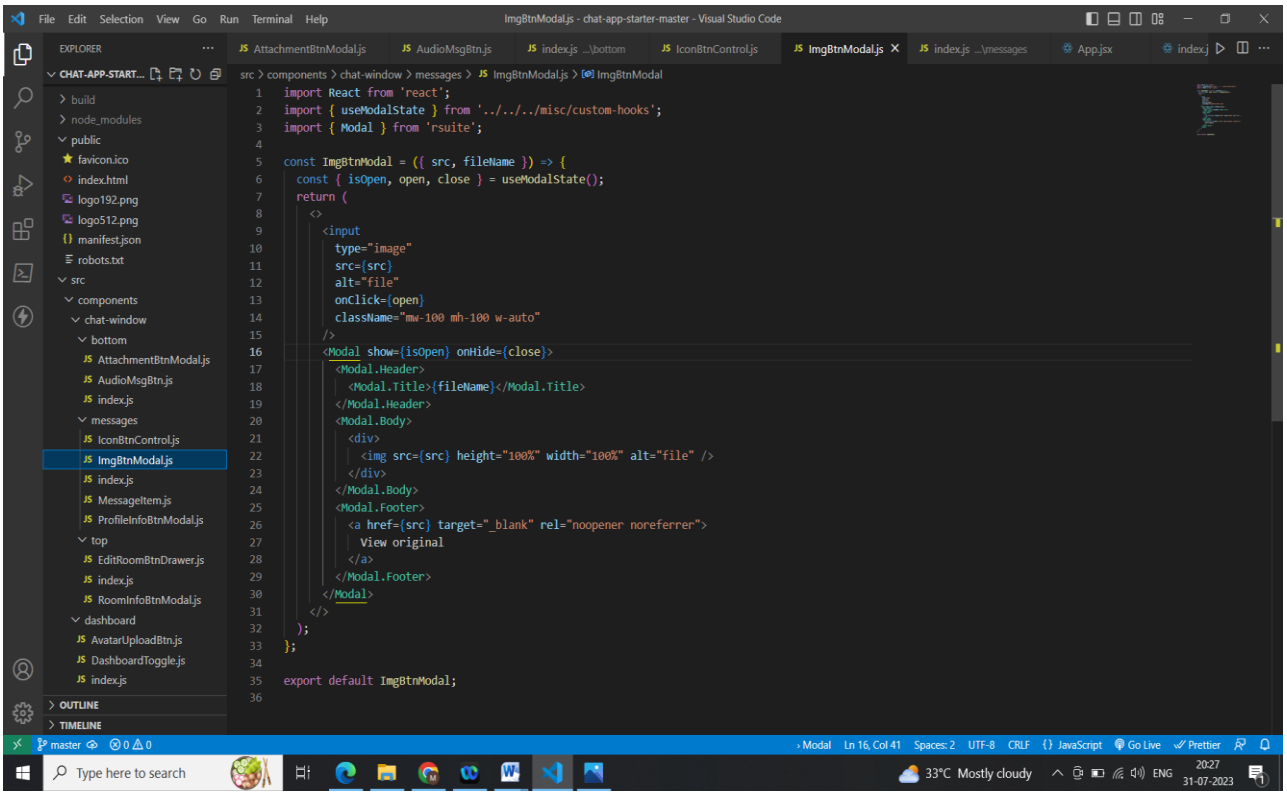

Figure 11. Screenshot of ImgBtnModal.js

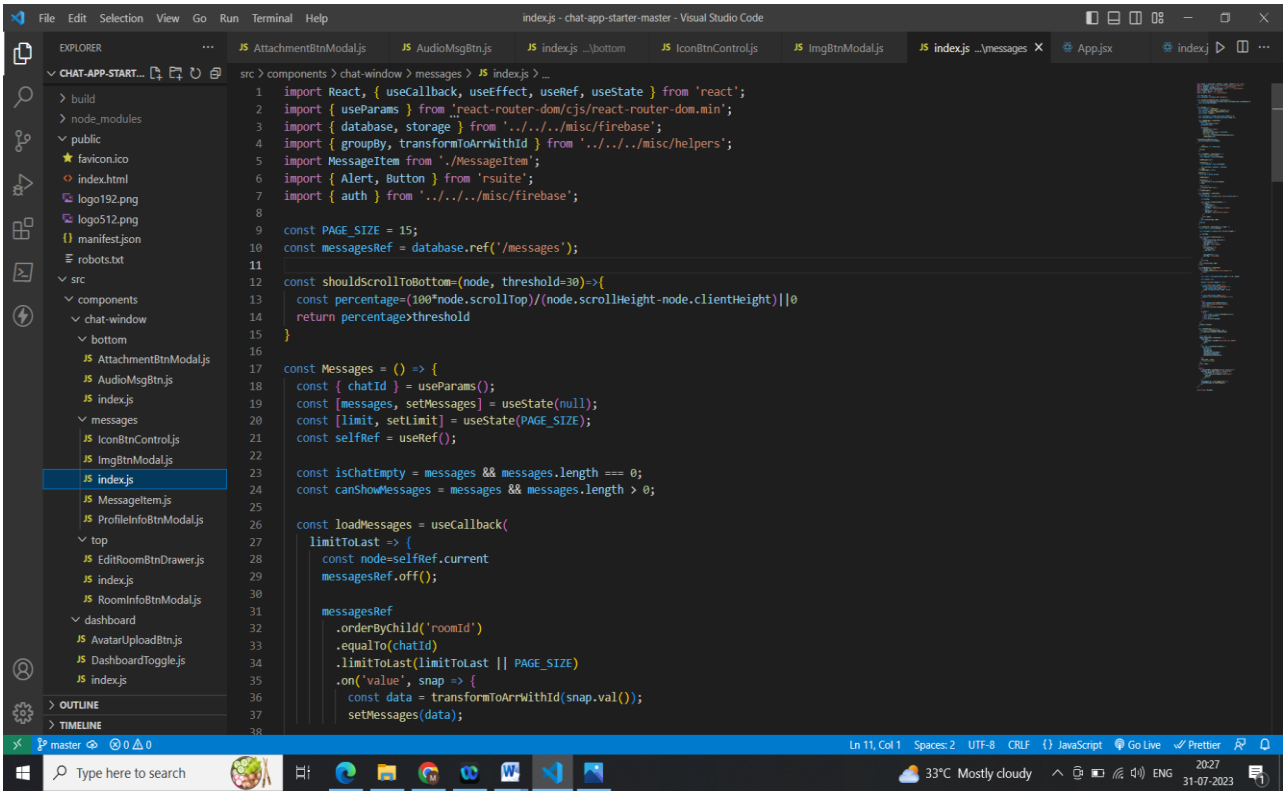

Figure 12. Screenshot of Index.js

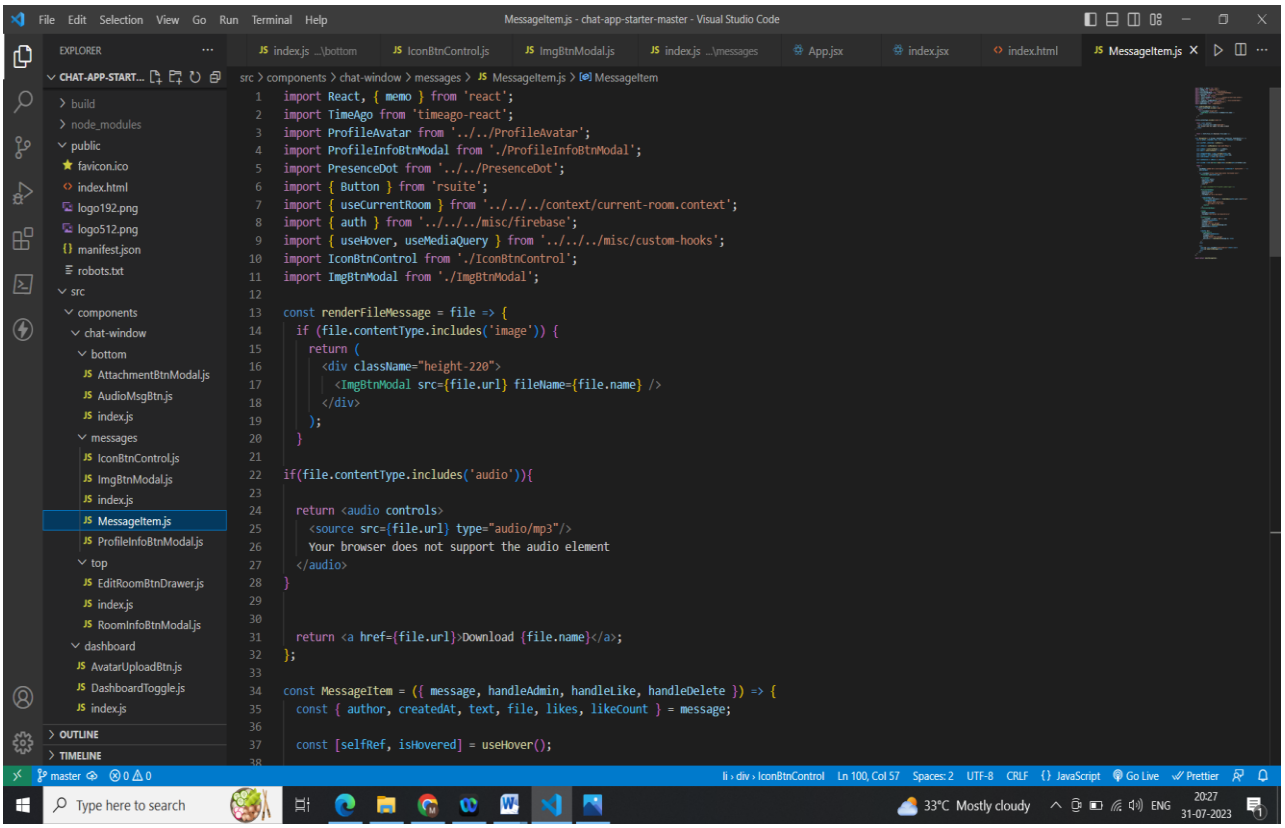

Figure 13. Screenshot of MessageItem.js

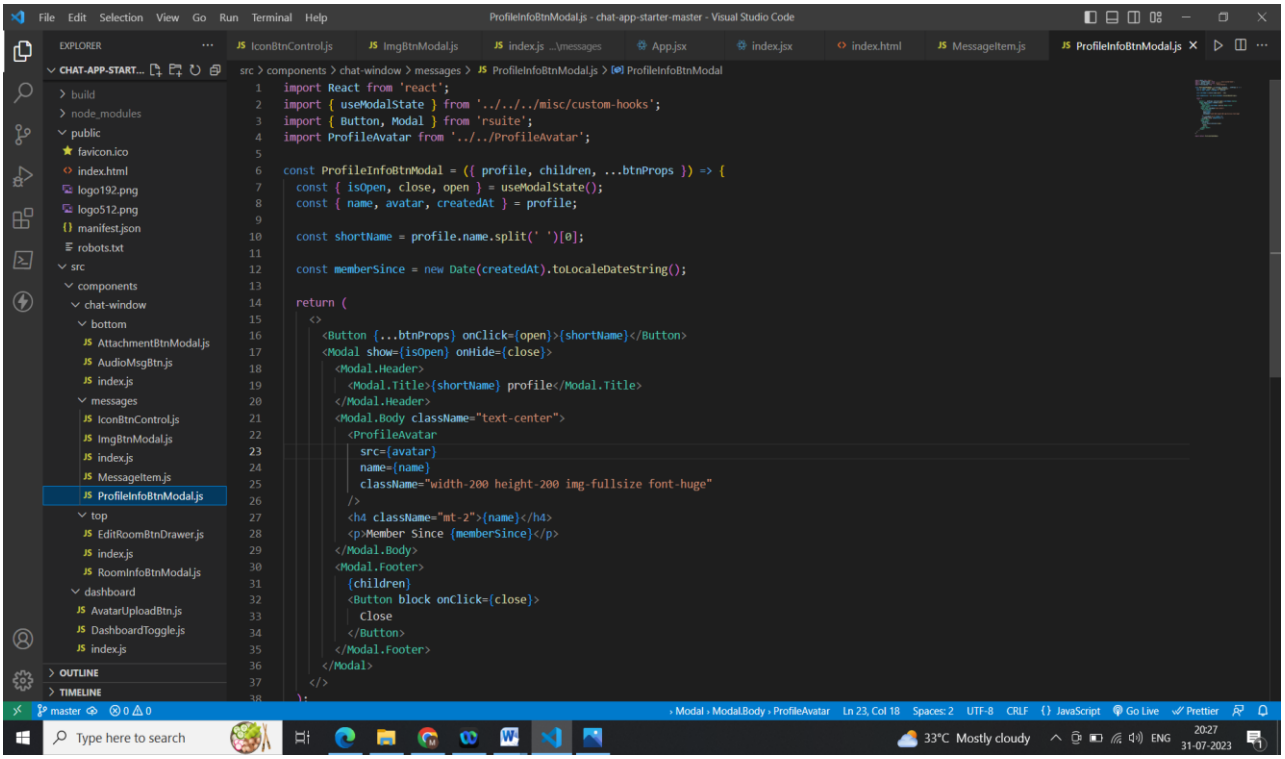

Figure 14. Screenshot of ProfileInfoBtnModal.js

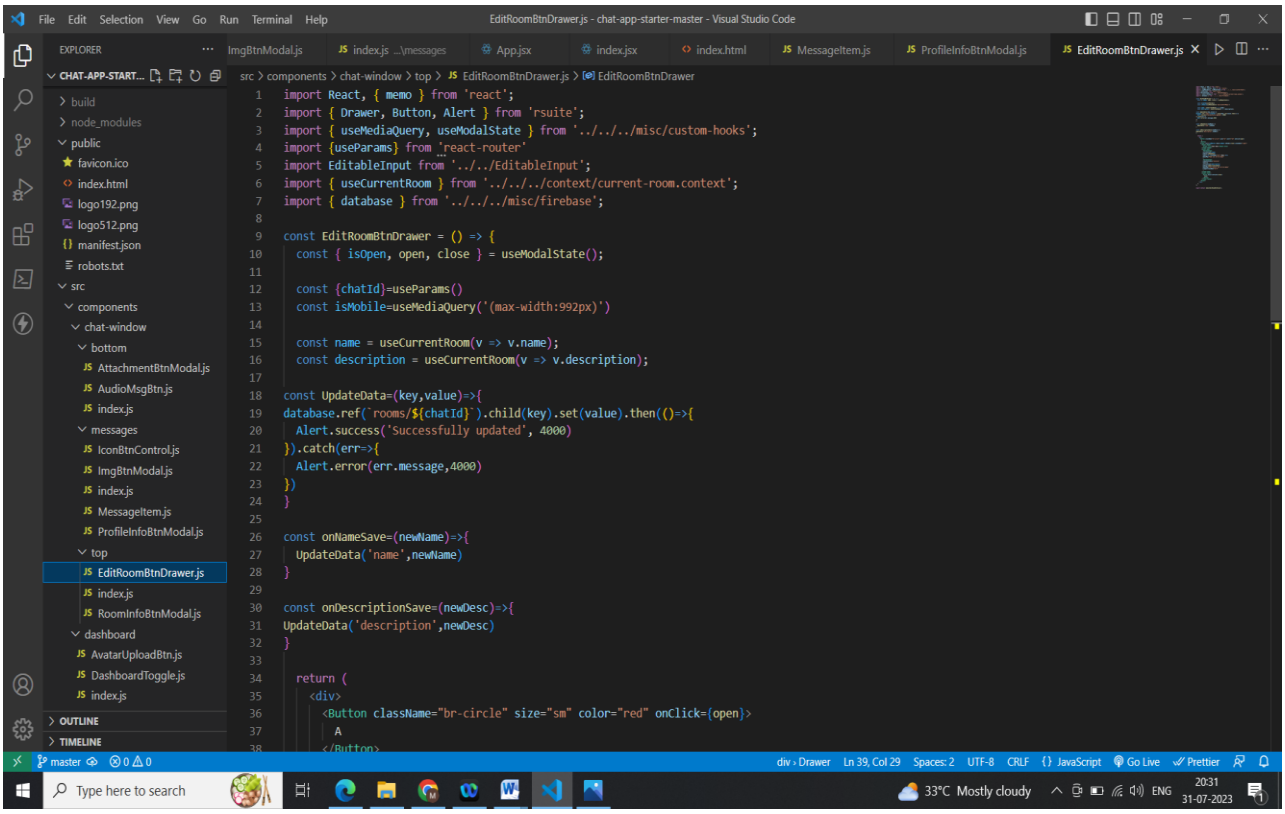

Figure 15. Screenshot of EditRoomBtnDrawer.js

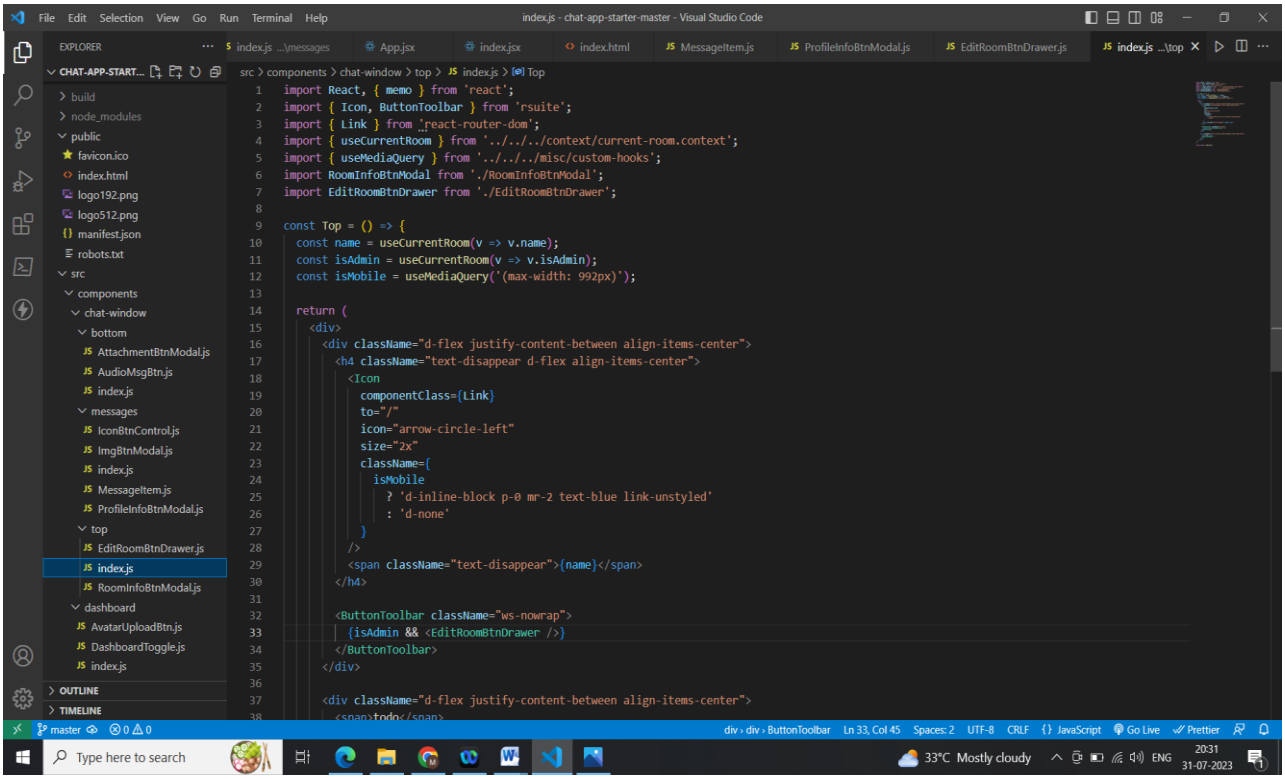

Figure 16. Screenshot of Index.js

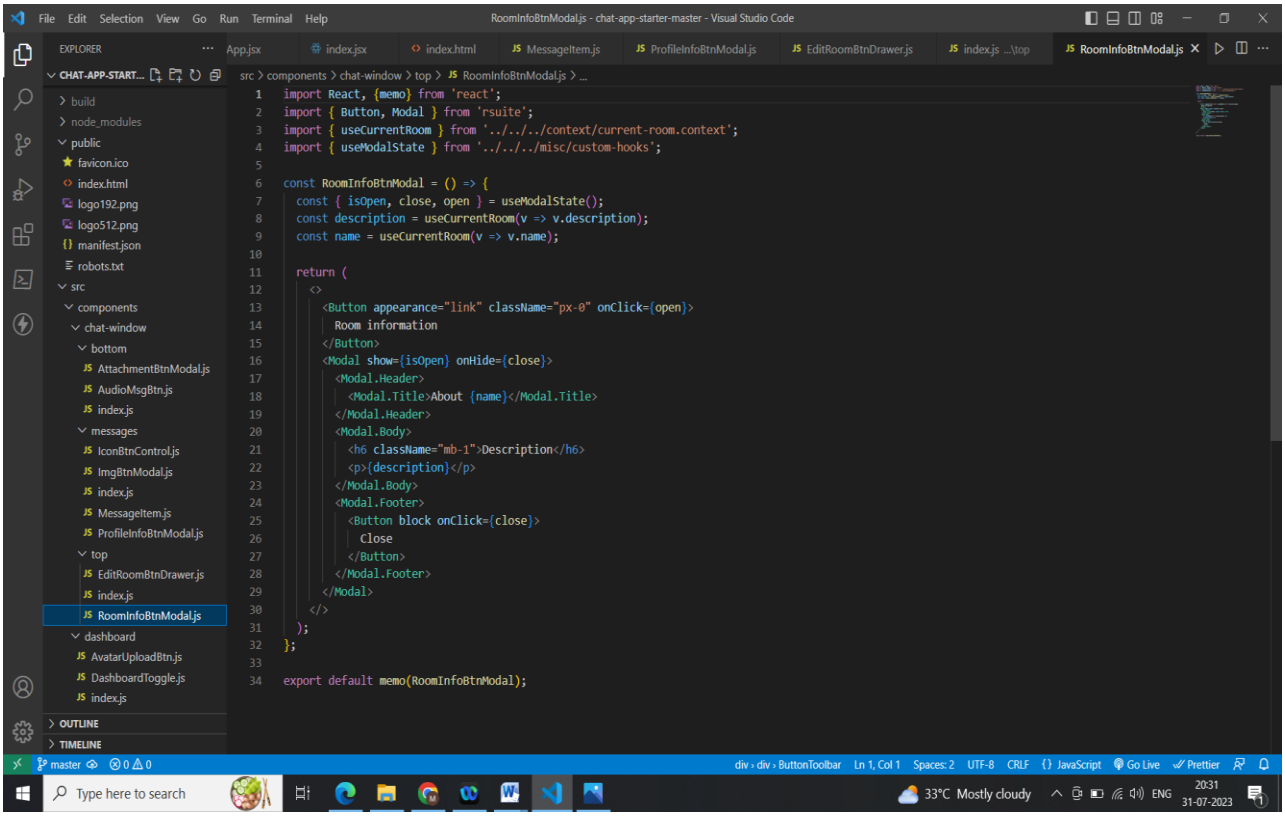

Figure 17. Screenshot of RoomInfoBtnModal.js

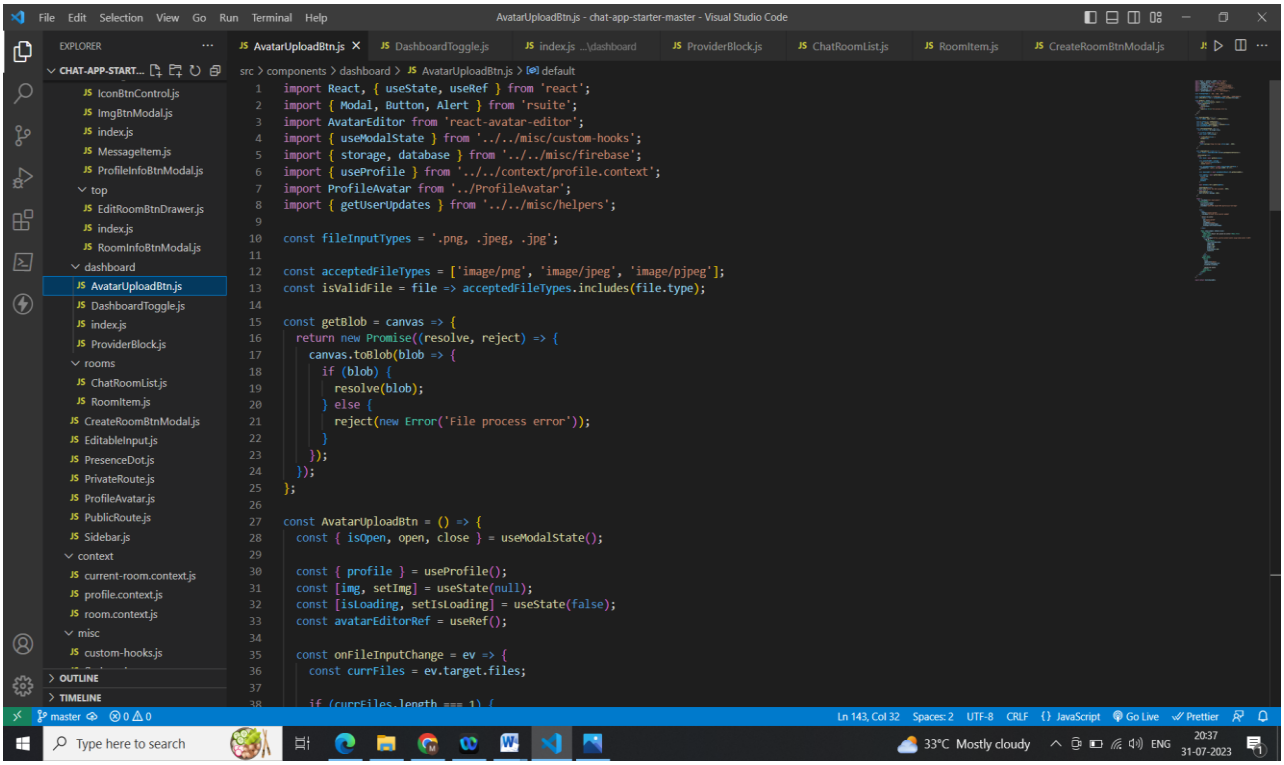

Figure 18. Screenshot of AvatarUploadBtn.js

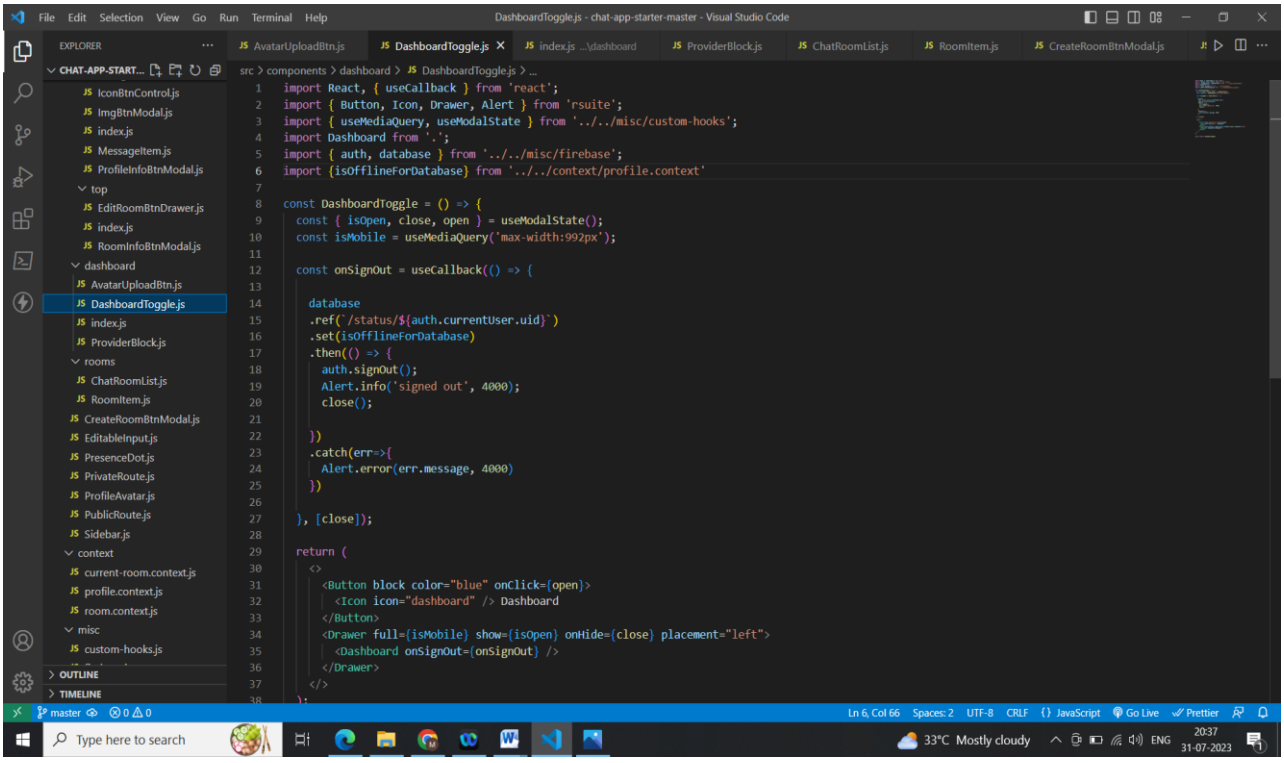

Figure 19. Screenshot of DashboardToggle.js

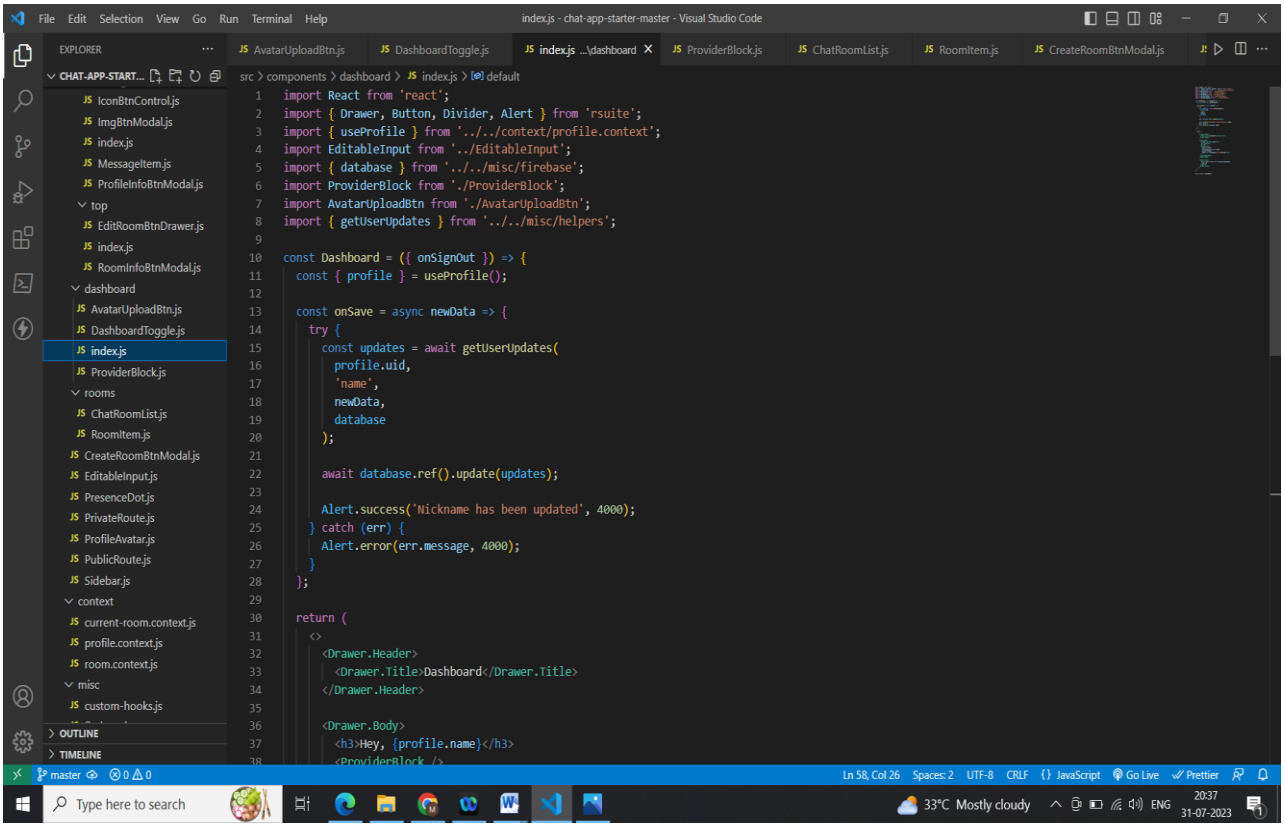

Figure 20. Screenshot of index.js

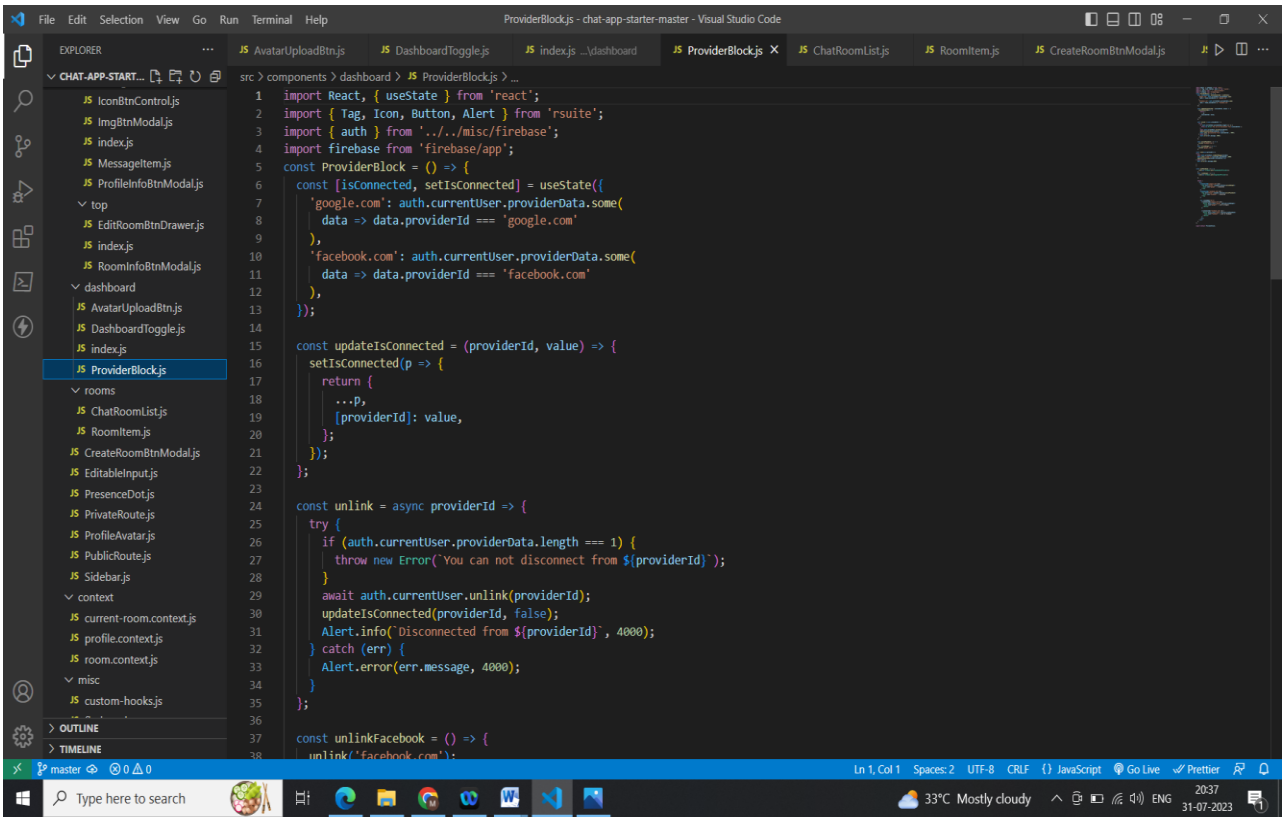

Figure 21. Screenshot of ProviderBlock.js

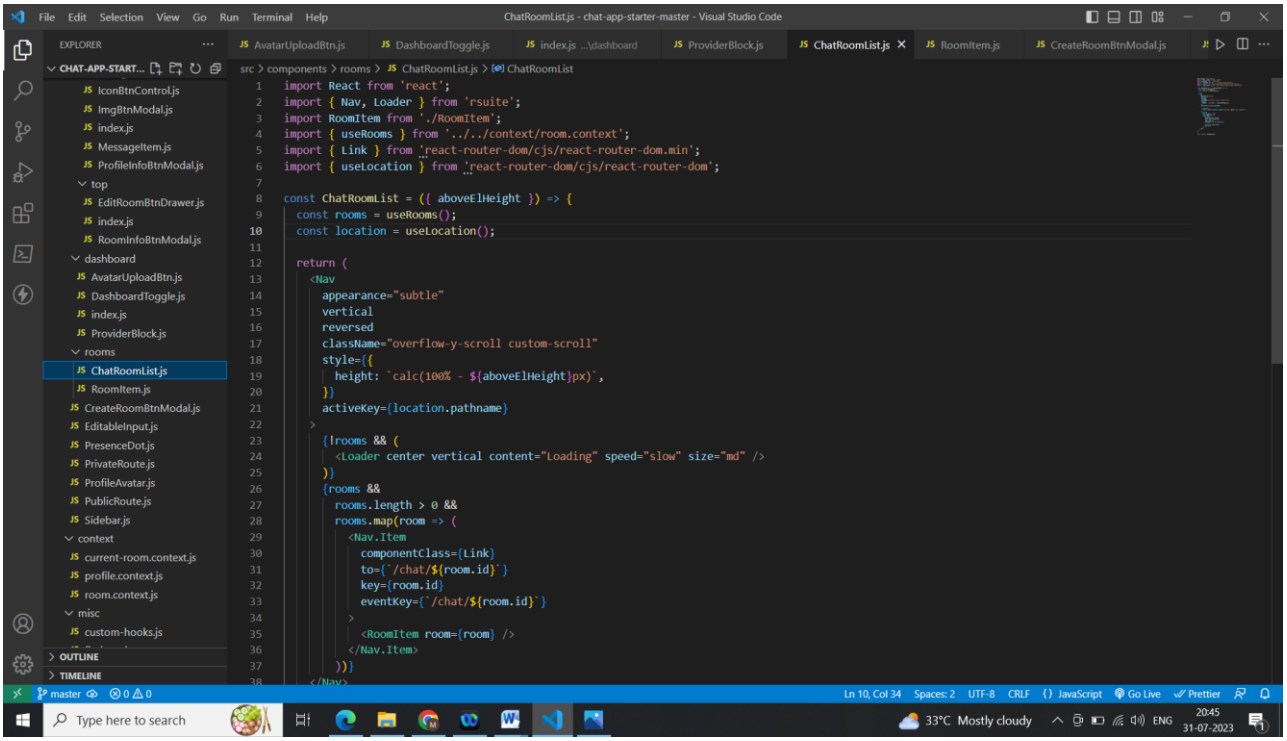

Figure 22. Screenshot of ChatRoomList.js

6.2. Screenshots of Working Project

Login Page

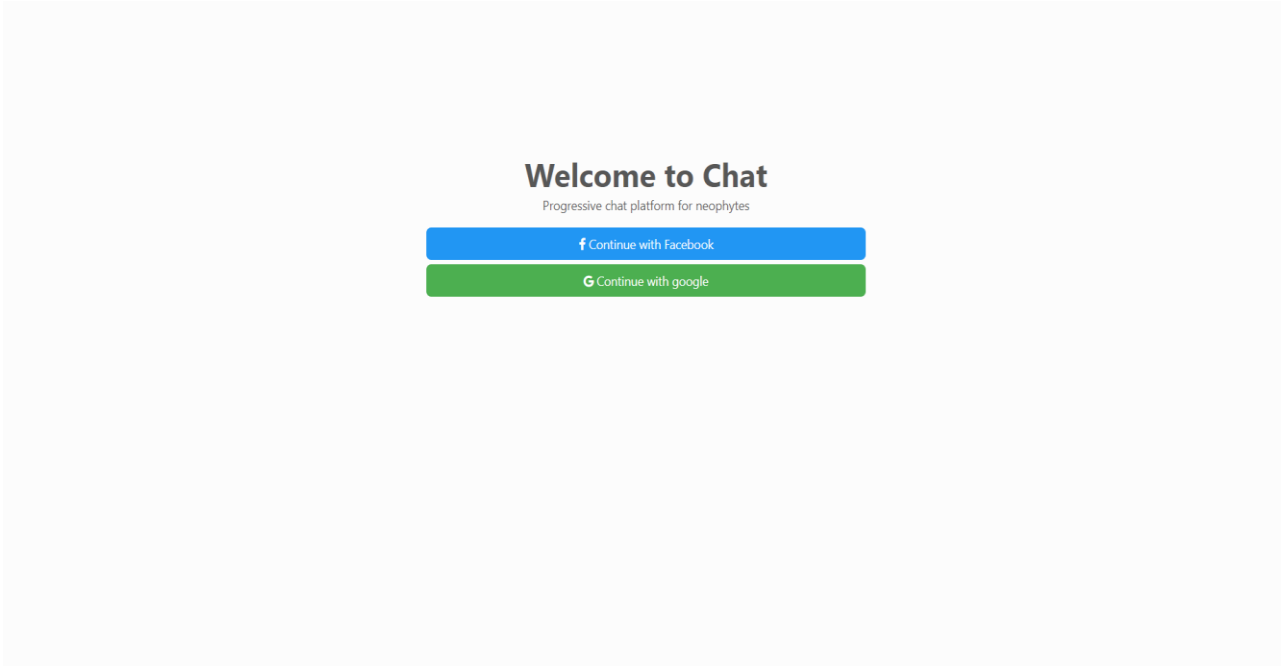

Figure 23. Screenshot of Login Page

Dashboard

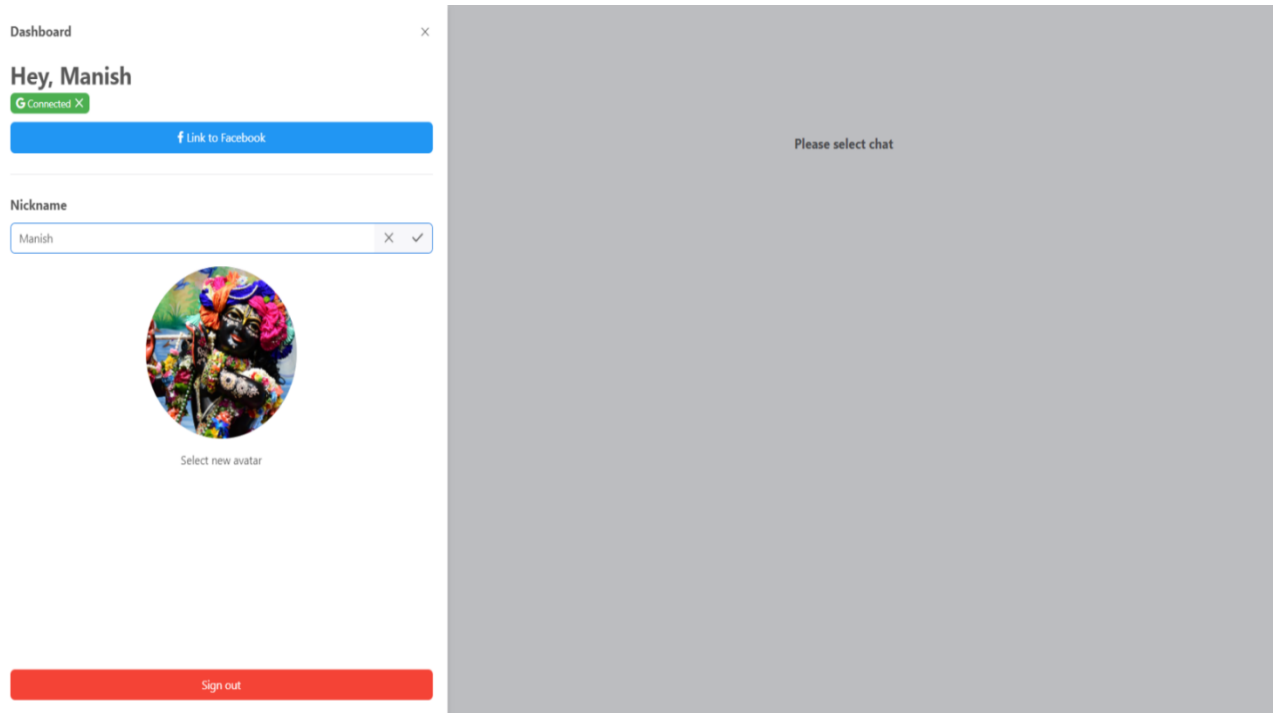

Figure 24. Screenshot of Dashboard

Chat rooms

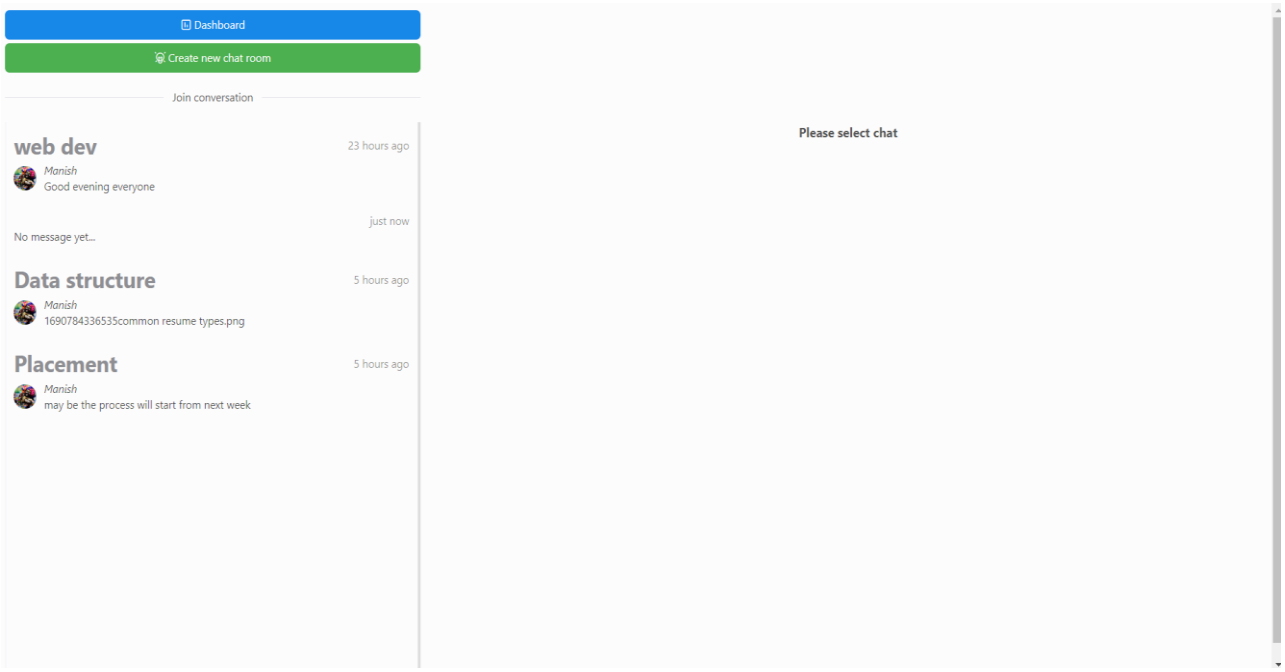

Figure 25. Screenshot of Chat rooms

Create new chat room

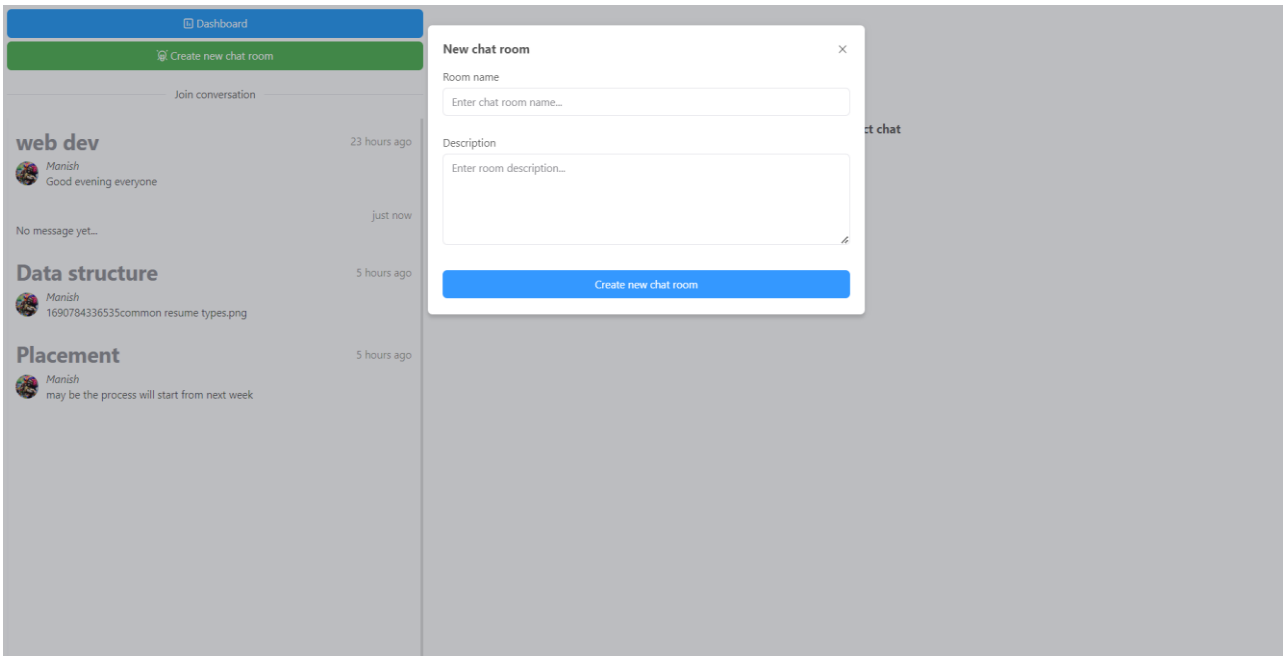

Figure 26. Screenshot of create new chat room

Chats

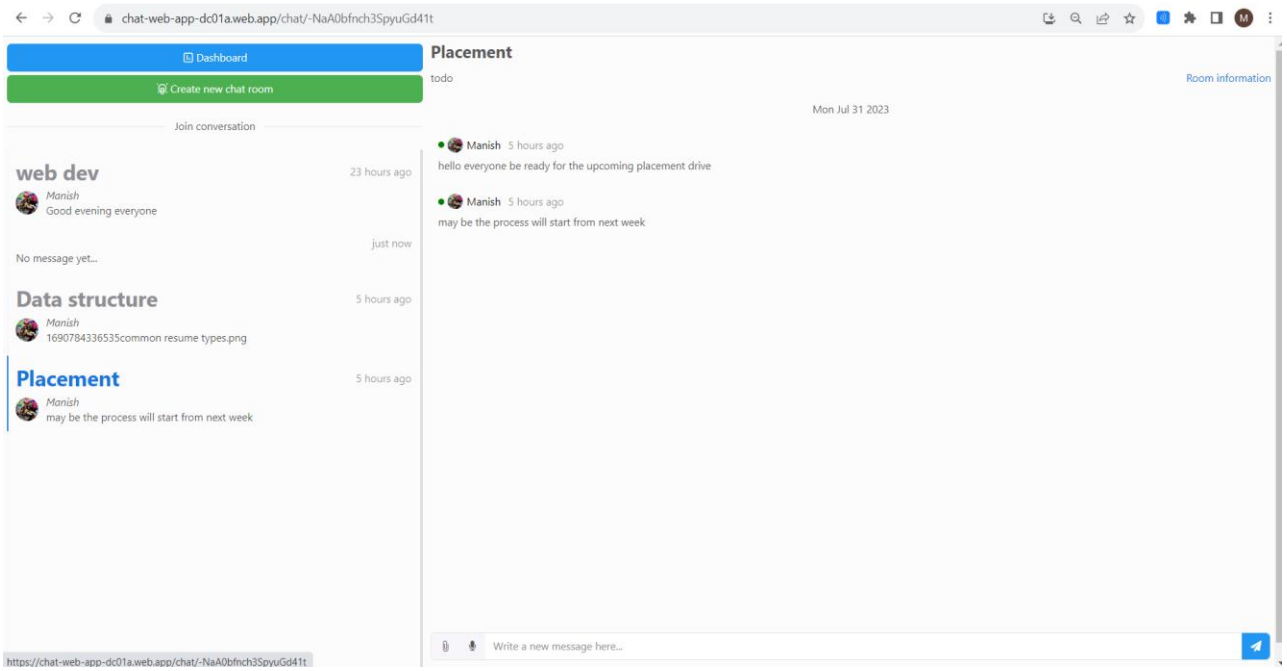

Figure 27. Screenshot of chat

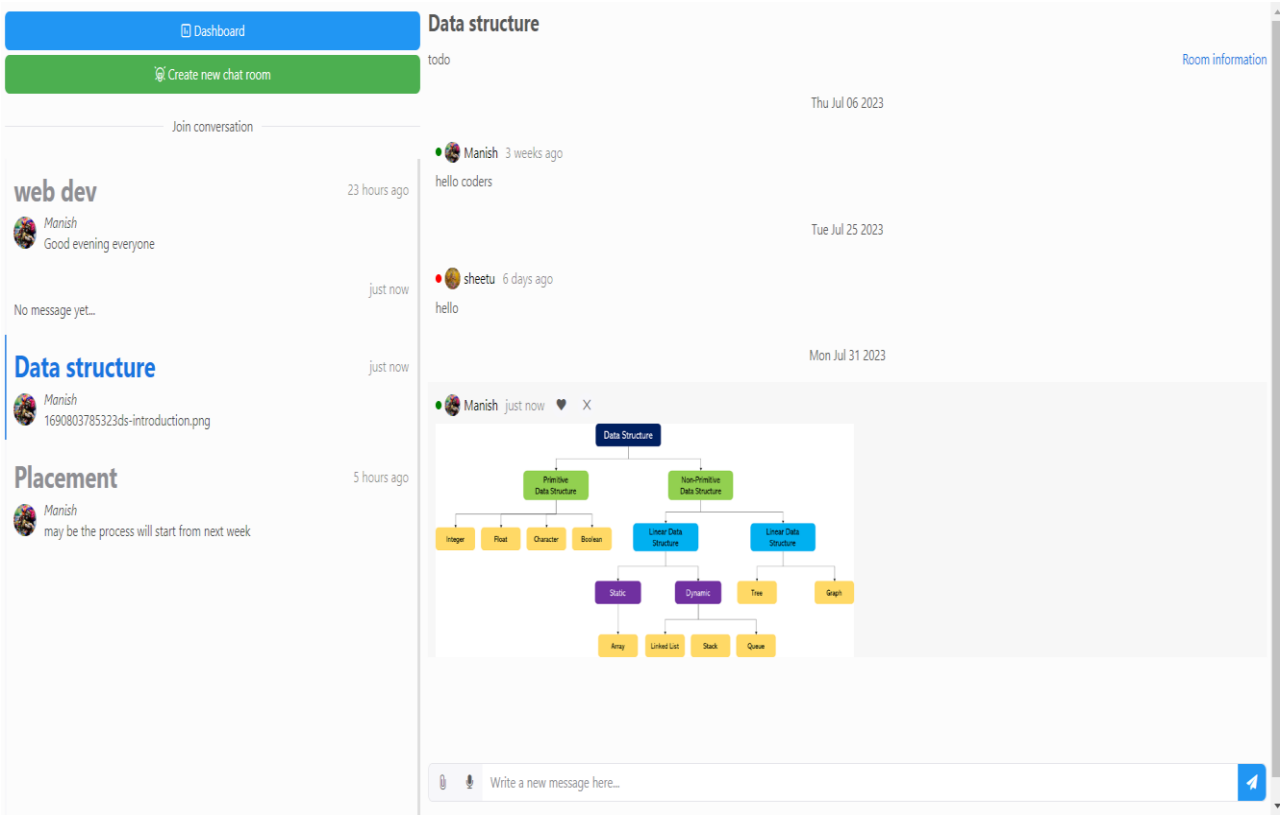

Figure 28. Screenshot of chat

6.3. Testing

The testing phase of the "Chat Application using React and Firebase" was a crucial step in ensuring the application's reliability, functionality, and user satisfaction. Rigorous testing methodologies were employed to identify and rectify any bugs, errors, or usability issues, leading to a polished and robust final product. Below are the details of the testing process**:**

1. Sign Up new User

- **Test Step**

Go to the Chat App page and Sign Up with a valid email and a six-character password then click “Sign Up”

- **Expected Result**

It should allow the user and enter the chat panel

• Actual Outcome

A new user is entered in the chat panel.

2. Check New Created User in Firebase database

• Test Step

Go to the Firebase console project with Gmail and password. Check the Firebase´s

Authentication users

• Expected Result

Newly Created User email, created date and unique uid in Firebase

• Actual Outcome.

A new Created User is available in Firebase Authentication user list.

3. User messages with name and time

• Test Step

Logged in user writes the message in Send Message Form

• Expected Result

Users write any message and receiver views the message with sender´s name and time

• Actual Outcome

Sender´s name with time is viewed by receiver.

4. Logged in User chat data in Firebase

• Test Step

Go to the Firebase console project with Gmail and password. Check the Firebase´s real-time database data

• Expected Result

User´s chat messages data available in the real-time database

• Actual Outcome

Chat messages data of user are available in Firebase database.

5. Logged in User able to create New Chat Room

• Test Step

User assigns a name to New Chat Room in empty field under the Rooms

• Expected Result

User creates a Room

• Actual Outcome

New Room is created by user.

6. Delete the user chat data in Firebase database

• Test Step

Go to the Firebase console project with Gmail and password. Check the Firebase´s real-time database data. Delete the chat history under the selected room

• Expected Result

Administrator allows deleting user chat messages or data that will be also deleted in the selected room of the chat panel

• Actual Outcome

The messages or data is lost by deleting in the Firebase real-time database.

7. Logged in user Refreshes chat automatically

- **Test Step**

Logged in user writes the message in Send Message Form

- **Expected Result**

Chat panel windows Refreshes automatically and is also observed by user

- **Actual Outcome**

Window is refreshed in the chat panel automatically.

7. CONCLUSION

The development of the "Chat Application using React and Firebase" was a fulfilling journey that resulted in the creation of a dynamic and interactive platform for real-time communication. The project successfully achieved its objectives of providing users with a seamless and user-friendly chat experience. Throughout the development process, various technologies and methodologies were employed to ensure a robust and reliable application.

The integration of React on the front-end allowed for the creation of modular and reusable components, enhancing the user interface and interactions. SCSS styling streamlined the application's design and maintenance, while Firebase services provided a scalable and efficient back-end solution. The real-time messaging feature, powered by Firebase Realtime Database, facilitated instant message delivery, promoting fluid conversations among users.

The project's success was further enhanced by the thorough testing phase, which identified and addressed potential issues, ensuring a smooth and error-free application. User feedback from User Acceptance Testing played a pivotal role in making valuable refinements and improvements to meet user expectations.

In conclusion, the "Chat Application using React and Firebase" has proven to be a well-crafted solution for real-time communication needs. The project demonstrated the effective implementation of cutting-edge technologies and methodologies to create a user-centric application. As a result, the application stands as a testament to the team's dedication, creativity, and commitment to delivering a feature-rich and user-friendly chat platform. Moving forward, the project serves as a solid foundation for potential enhancements and opens up opportunities for further exploration in the realm of real-time applications.

8. BIBLIOGRAPHY

1. HTML - https://www.educba.com/advantages-of-html/

2. HTML - https://www.quora.com/What-is-an-advantage-of-HTML

3. SCSS - www.sass-lang.com

4. React - https://www.peerbits.com/blog/reasons-to-choose-reactjs-for-your-web-development-project.html

5. DOM - https://www.w3schools.com/js/js_htmldom.asp

6. DOM - https://www.educative.io/answers/what-is-dom

7. Visual Studio Code - https://code.visualstudio.com/

8. Visual Studio Code - https://en.wikipedia.org/wiki/Visual_Studio_Code

9. Life cycle of React JS - https://www.freecodecamp.org/news/how-to-understand-a-components-lifecycle-methods-in-reactjs-e1a609840630/

10. Firebase - https://en.wikipedia.org/wiki/Firebase

11. Firebase - https://www.altexsoft.com/blog/firebase-review-pros-cons-alternatives/