WhatsApp Web Clone

Project Description

We all have used WhatsApp web on our PCs. Ever thought of making it yourself? The interface that we will be making for our project will look alike. Firebase's real-time database will give you a seamless messaging feature.

Author

Sheetal Singh

Collaborator(s)

Ayush Kumar Shaw, Kiran Suresh

Project Language(s)

JavaScript

Difficulty

Advanced

Duration

60 hours

Prerequisite(s)

HTML, CSS, JavaScript

Skills to be learned

React Basics, React Hooks, Firebase Firestore Integration, React Context API

Overview

Objective

"There is no better way to understand a framework, library, and its features, than by making projects." You will be making the following features :

- WhatsApp Web UI

- Create rooms or groups
- Sign in with a Google account
- Last seen for groups
- Messaging boxes [green for own and white for the replies of others]
- Send message in the group with timestamp

Learning Scope

This project will give you a hands-on experience with the React library. Fullstack development currently has many different stacks and technologies to learn. It is very easy to get overwhelmed and get distracted. React is one of the most popularly used Front-end libraries used by companies such as Facebook, Pinterest, Uber, Instagram, and many more. Beginner tutorials and mini-projects are good for beginners but a full-fledged project can make you understand concepts with real-world applications. You will learn not just about UI designing but also integrating it with the backend. You will be learning about :

- Basics of React
  - Functional Components
  - useState
  - useEffect
  - Props
- Context API
- Material- UI (Design)
- Firebase
  - Authentication (using `Sign In With Google`)
  - Firestore(DB)

[Note: We're using `Sign In With Google` instead of the traditional way of WhatsApp to keep it simple.]

Project Stages

We can break down the project in the following stages: 1. Front-end - Sidebar Component - Chat Component 2. Back-end - Authentication - Database(Firestore)

High-Level Approach

- Identify the UI components and segregate them into smaller components
- Start with building individual components with static data which will be replaced by real-time data later
- CSS styling and Material-UI icons implementation

- Setup project on Firebase
- Configure and install Firebase in your React project
- Design the database for your requirement
- Authentication - Add authentication in your project to access the chat section only after authenticating with a google account.
- Integration of data with UI

Project demo: https://www.youtube.com/embed/HiU5JkQ9dS4

Applications

- Building an efficient and real-time text-based communication solution for the web

Credits

- "Clever Programmer organization"

Task 1

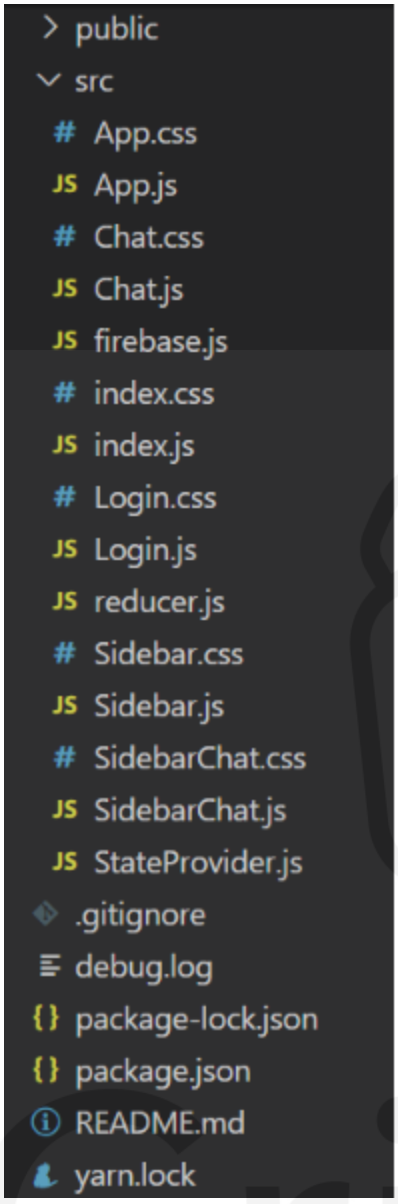
File Structure and Dependencies installation

Before you get started with building the components it is better to plan your components. This will give you a clear idea of what goes where.

Requirements

- Create your React app using `create-react-app`.
- Your file structure should look similar to the image below.

Crio.Do



- There are multiple dependencies that you need to install to get started with your project.

```
"dependencies": {
    "@material-ui/core": "^4.11.0",
    "@material-ui/icons": "^4.9.1",
    "@testing-library/jest-dom": "^5.11.4",
    "@testing-library/react": "^11.1.0",
    "@testing-library/user-event": "^12.1.10",
    "firebase": "^8.2.1",
    "react": "^17.0.1",
    "react-dom": "^17.0.1",
    "react-router-dom": "^5.2.0",
```

```
    "react-scripts": "4.0.0",
    "web-vitals": "^0.2.4"
  },
```

References

- Create react app
- npm install
- React components, JSX, props for beginners
- Material-UI

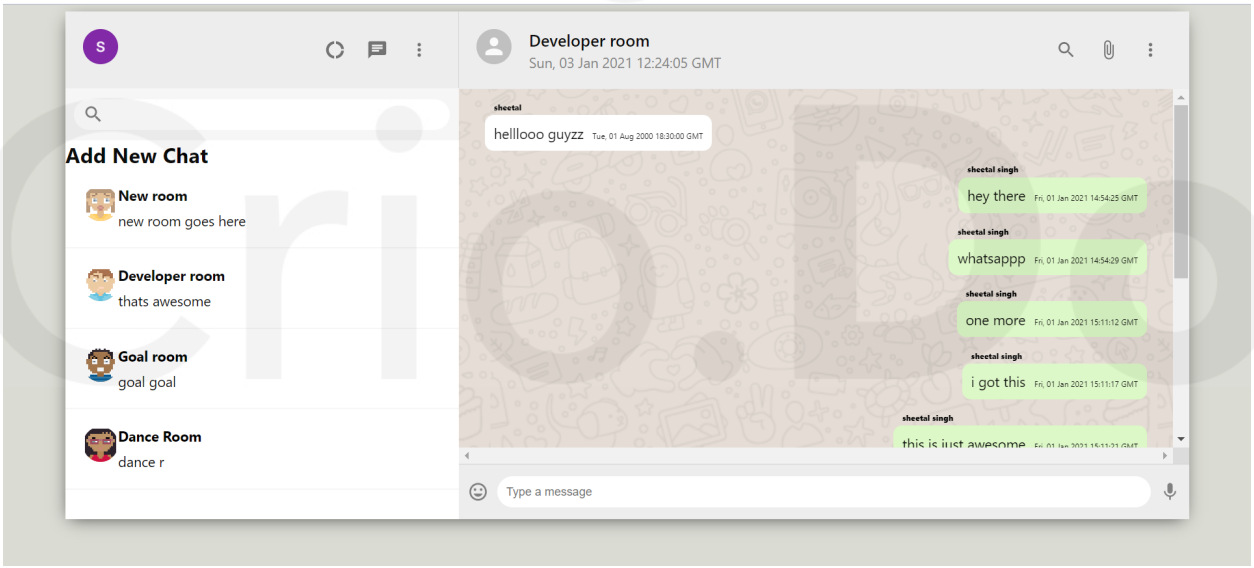
Expected Outcome

You should be able to successfully install the React application with all the required dependencies. You can check the `package.json` file for verification.

Task 2

App Components

Now you can start designing your UI components. For UI components you can refer to the below image. It basically has two major components: - Sidebar Component - Sidebar header [with account avatar and icons] - Sidebar Search - Sidebar Chat List - Chat section Component - Chat header - Chat Body - Chat Footer



Requirements

- Design the components to make them similar to the above screenshot.
- Additional requirements:

- The header icons should be clickable.
- There should be an input section for the search and chatbox.
- The page should be responsive.

References

- Material-UI icons
- Basic Concepts of Flexbox
- How to pick colors from the image
- React component composition

Bonus

The Material-UI icons used for this WhatsApp clone are

- SearchIcon
- MoreVertIcon
- AttachFileIcon
- MicIcon
- InsertEmoticonIcon
- DonutLargeIcon
- ChatIcon

Bring it On!

The Icons you would be using won't be clickable or show click animation initially. Find out a way to make seem like a button though they are icons.

Expected Outcome

After making each of the components individually, you should be able to assemble them in the `app.js` file and should have a UI look somewhat like the image given above.

Task 3

Firebase

Firebase is a cloud platform provided by Google to enable mobile and web application development. We will be using Firebase for Google authentication and Firebase Firestore for the NoSQL database for our chat application.

Requirements

- Create a Firebase project for your application.
- Create a Firestore database.

- Find a method to ensure that your Firestore is connected to your React app once the configuration file `firebase.js` is in place.

References

- Setup and configure Firestore with a React application
- Beginner's Guide to hooks
- Understanding Firestore

Expected Outcome

- Your configuration file should look like

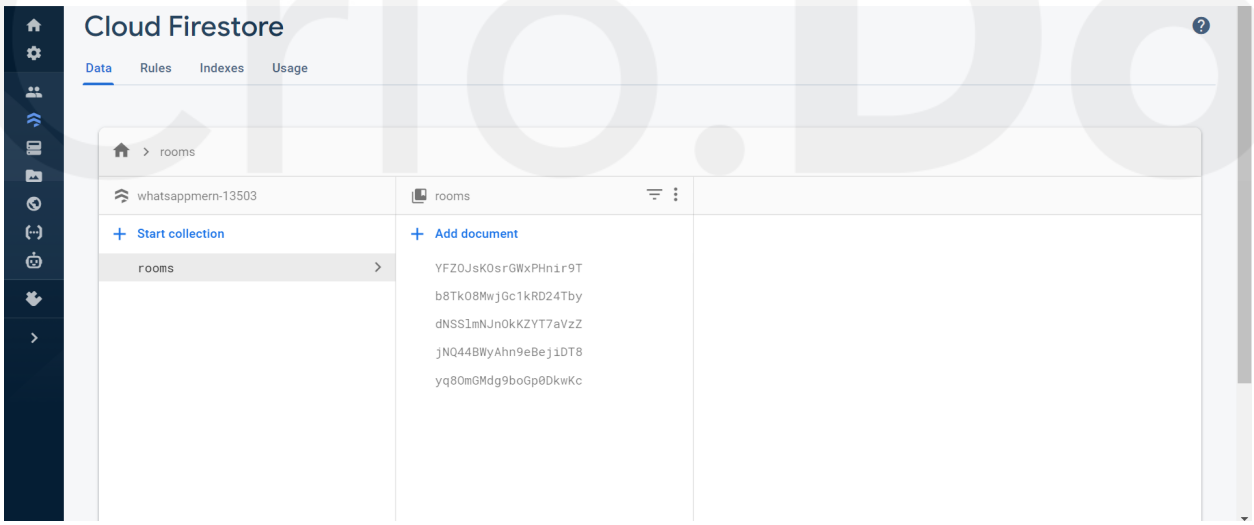
```
import firebase from 'firebase'

const firebaseConfig = {
    apiKey: "<apiKey>",
    authDomain: "whatsapp-13503.firebaseapp.com",
    databaseURL: "https://whatsapp-13503.firebaseio.com",
    projectId: "whatsapp-13503",
    storageBucket: "whatsapp-13503.appspot.com",
    messagingSenderId: "<messagingSenderId>",
    appId: "<appId>"
  };

  const firebaseApp = firebase.initializeApp(firebaseConfig);
  const db = firebase.firestore();
  const auth = firebase.auth();
  const provider = new firebase.auth.GoogleAuthProvider();

  export {auth,provider} ;
  export default db;
```

- You should have access to your firestore database:



Task 4

Component

After the UI gets ready, now is the time to fetch the data from the database. You need to make different states to manage different components. Those states will be responsible to push data to your Firestore database and get data and display them in real-time.

Requirements

- Design your Firestore collection according to the requirement of the chat application. for example, you will make a collection of rooms that hold room names and id. Those individual rooms hold a collection of chats with chat messages, sender names, and timestamps (for last seen and message time).
- Fetch the required data from the configured Firestore DB.
- Replace the static data with the database data.

References

- Understanding React hooks
- How to use useState and useEffect with example
- React hooks API - useState, useEffect
- Fetching data from Firestore
- Consume data from Firebase Firestore in a React app

Tip

- You can add images for your avatars using the state from dice bear avatars. This can be a little add-on from boring plain avatars. Use a random function for different avatars for different rooms.

Expected Outcome

- You should be able to display data from the database.
- You should be able to add rooms with a prompt to the database.
- The chatbox should take the input and display it as a chat in real-time.

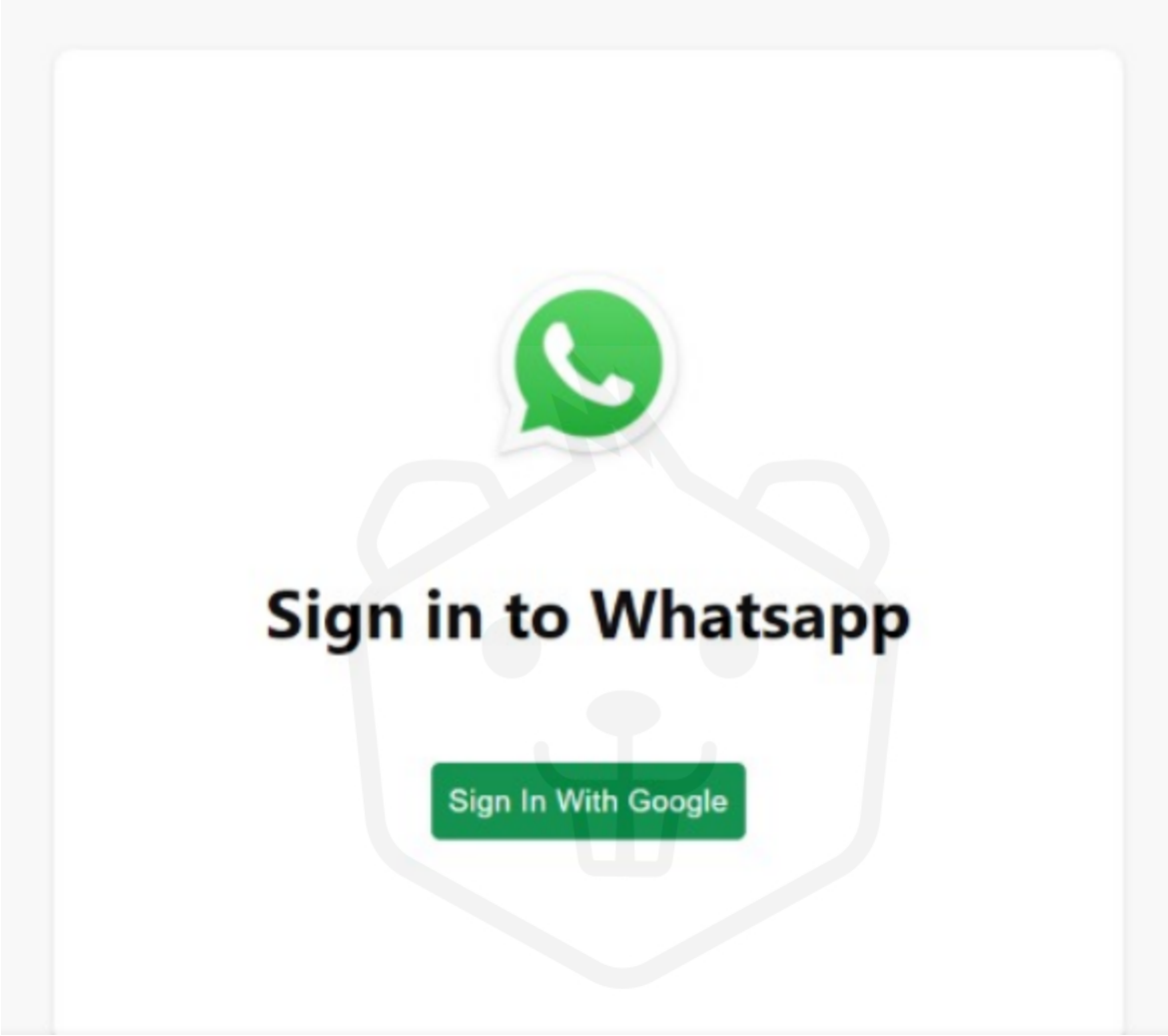
Task 5

Add Authentication

As the final stage, we'll be adding a form of authentication to our application. For keeping it simple, we'll be integrating `Sign In With Google` with the React application.

Requirements

- Login page to redirect to Google authentication prompt.



- Chat section with Google ID avatar on successful login. (Refer to the image of the UI)
- Use conditional rendering to add a class of sender (green box) and others (white box) as the chats, as we have in WhatsApp originally. You can use the user display name and Google auth id name to add the condition.

References

- Firebase Google sign in with React
- Firebase Google Authentication
- React Context API

Tip

React states are local. Authentication uses context API to successfully log in and interact with other components. Context API needs to be understood to get started learning about global state management using libraries such as Redux.

Bring it On!

Deploy the application using Firebase.

Expected Outcome

This would be the final stage of your project. You should be able to sign in with your Google account, have your chats saved, and able to send and receive chats.

Crio.Do