Microsoft Engage Mentorship Program 2021

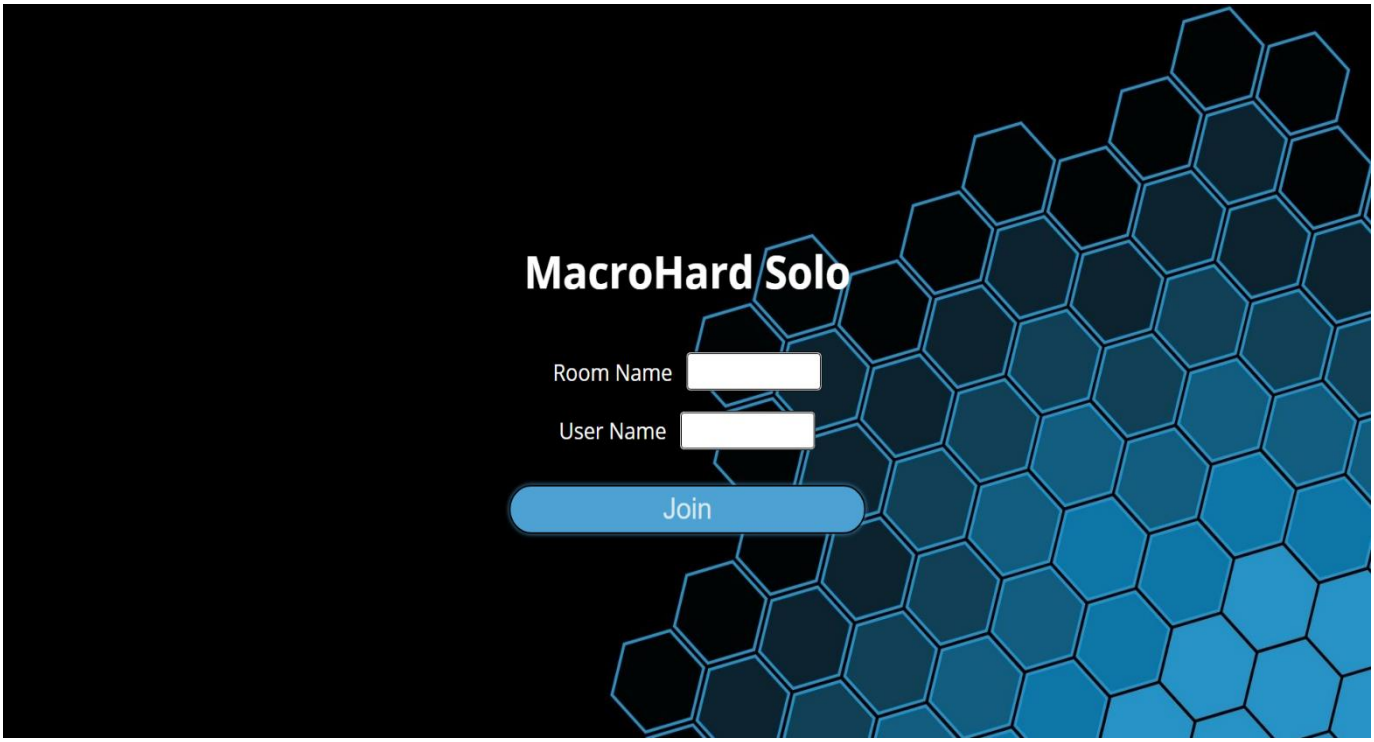
Project Documentation

Microsoft teams clone

MacroHard Solo

By- Aneesha Sharma

July 12th, 2021



MacroHard Solo is a web application that allows users to connect and call with other users in real time, along with offering many other functionalities.

Git Repo: https://github.com/Aneeshass/teams-clone

Live Demo: https://macrohard-solo.herokuapp.com/

1.0. Introduction

1.1. Purpose

The purpose of this document is to present a detailed description of the web application that I made, named “MacroHard Solo”. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and the constraints under which it will not.

1.2. Scope/ Intended use

This software system is a video calling web application which allows a group of users to join the call and interact with each other. It also provides a variety of other features to user for them to have a close Microsoft teams experience.

This app allows you to create a room with your provided room name/id with you can share amongst those whom you want to connect. It also asks for you to provide a username, by which other users will identify you.

Along with the basic functionality of turning on/off your video and audio, more specifically, this app allows users to chat during the call, without disrupting the flow of the call. Not just that, but this application has a chat area where users can chat with those in the call, before the call starts and after the call ends as well. This app also provides a few other features alongside, such as, sharing your

screen with everyone in the call, recording a call and downloading (.webm format) it to go over the meeting after it ends as well. There are also features which provides the user with the number of participants present in the call or in the chat room. The users can also invite more people by sharing the room id using the invite button.

1.3. Intended Audience

The intended audience of this document are JavaScript developers and anyone who wish to know more about the application before using. Readers are not required to know all the details; however, some knowledge of basic concepts such as such of Node.js modules, would be advantageous for reading of this document.

2.0. Overall Description

2.1. Technologies and APIs used

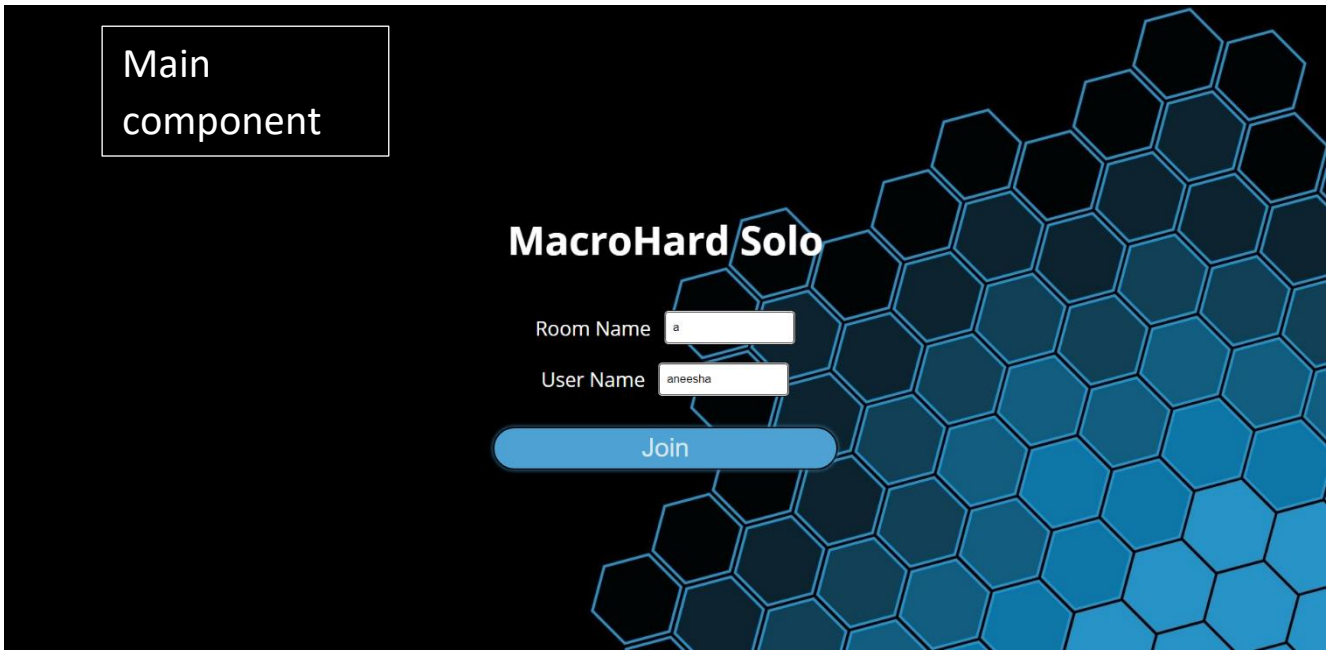
- React- React is a free and open-source front-end JavaScript library for building user interfaces or UI components.
- Node.js- Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser.
- Express- Express is a back end web application framework for Node.js designed for building web applications and APIs.

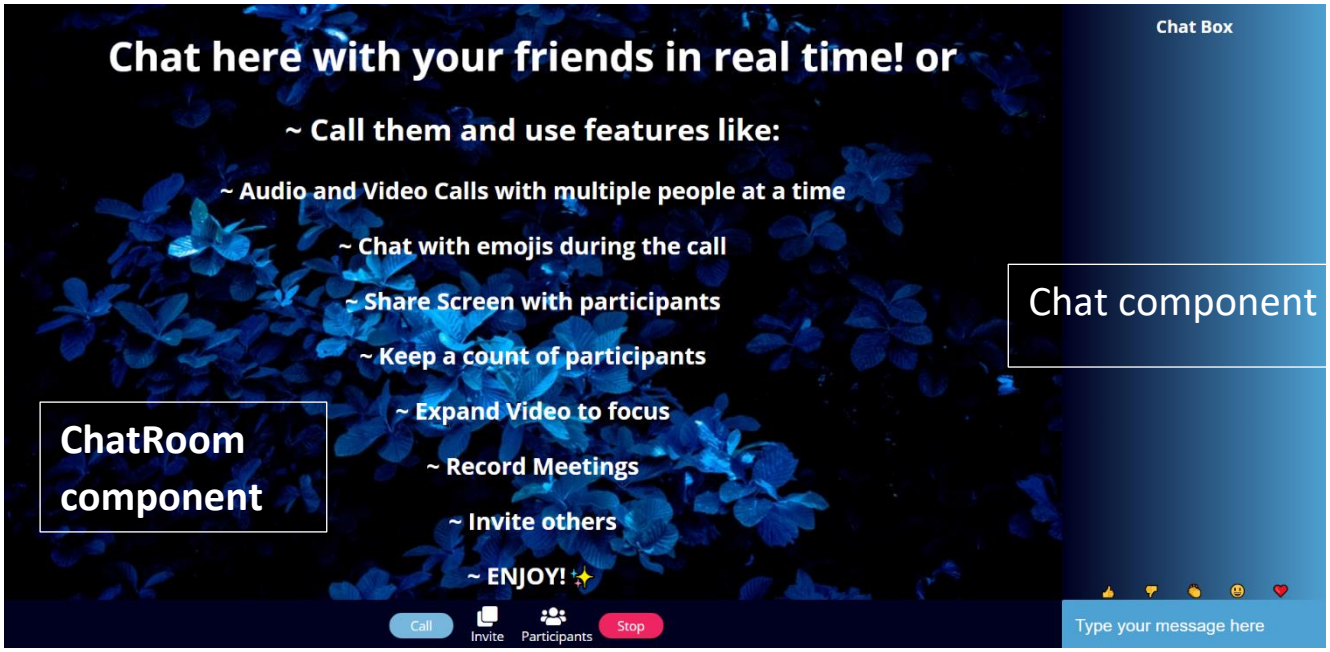
- WebRTC- WebRTC is a free and open-source project providing web browsers and mobile applications with real-time communication via simple application programming interfaces.
- Socket.IO- Socket.IO is a JavaScript library for real-time web applications. It enables real-time, bi-directional communication between web clients and servers.
- Styled-components- Utilising tagged template literals and the power of CSS, styled-components allows you to write actual CSS code to style your components without the mapping between components and styles.
- Simple-peer- It provides simple one-to-one WebRTC video/voice and data channels. It simplifies WebRTC peer-to-peer data, video, and audio calls.
- Font awesome- Font awesome lets you place awesome font icons just about anywhere using a style prefix and the icon's name.
- Google fonts- Google Fonts is a library of more than a thousand free and open source font families, an interactive web directory for browsing the library, and APIs.

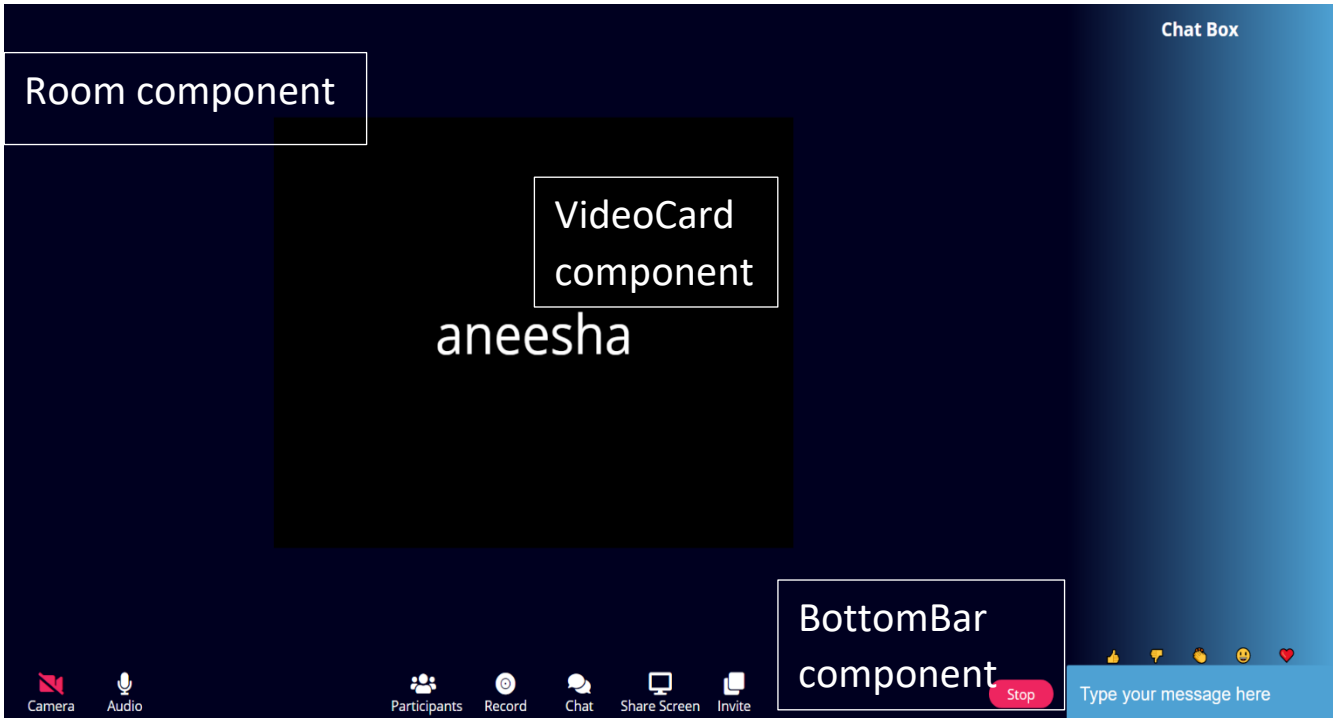
2.2. Components

This application uses the basic server and client-side files to segregate the code. There are 2 index.js files in both the sides with functionalities to create a

connection with the server and managing the DOM elements of the app with the server respectively. Apart from these, below the components used and re-used too, keeping in mind the agile methodology and easy and pleasing user interface.







1. Main- This component creates the landing page of the application where room id and username to initialize all other functions are taken in from the user.
2. VideoCard- This component created an area and video stream for any new user that connects.
3. Chat- This component allows users to chat with each other in real time. I have added some emojis as well to make the chat interesting.
4. BottomBar- This component creates a bottom bar which has buttons for all the functionalities this application has to offer such as button for the above mentioned chat component, leaving the call, recording, screen sharing etc.

5. RoomChat- This component creates a chatting area where those who have joined the room with room id can share with each other. It also has the above mentioned BottomBar component.
6. Room- This component creates a calling area where users can video call with each other alongside using the above mentioned BottomBar component's functionalities.

2.3. User classes & Functions

- **React**
  - I have used react hooks useEffect(), useState() and useRef() when and wherever needed. I used useState() for the use of buttons in the application to start or stop the function by pressing the same button again. I have used useRef() to store the user and its peers ref, their video refs, and even their screens refs to find and deal with each component when needed. I have also used useEffect() to tell what each components needs to after render.
  - To change routes within the application, I have used react-router-dom using BrowserRouter, Route, Switch.
  - I have also used ReactDOM that provides DOM specific methods that can be used at the top level of a web app to enable an efficient way of managing DOM elements of the web page.

- I have used render() method of ReactDOM to wrap various components of the application in a div element.

- **Socket.io**
  - I have used socket.io-client that tries to establish a WebSocket (server and browser) connection if possible.
    - And with that I have, in a constructor, established a connection to the server's IP and port running the SocketIO server. This constructor is used wherever a connection needs to made or destroyed.
- **Simple-peer**
  - I have used Peer from simple-peer to create and then that add peer in the stream wherever a connection is asked to be made.
- **WebRTC**
  - I used mediaDevices of getDisplayMedia() method to select and grant permission to capture the contents of a display for screensharing.
  - I used MediaRecorder which provided me the functionality to record media, provides by the mediaDevices.
- **Styled components**
  - To do the CSS for each component I have imported styled from styled-components to write the CSS with creating separate files and making a div for them in the same file.

2.4. Agile Methodology

Agile is the ability to create and respond to change. It is a way of dealing with, and ultimately succeeding in, an uncertain and turbulent environment. "Agile" represents the adaptiveness and response to change. During our mentorship, we were given a surprise feature according to which we had adapt and had to implement in our application. As our surprise feature, we were given the task for implement chat feature which not only works during the call but also before and after the call as well. To implement this feature in the limited period of time, I re-used the same chat component that I had made earlier to chat during the call. This saved me a lot of time as well as did not make my code unnecessary complicated and long. I also used the shifted the code to join the call to join the chat area before making a connection so that people in the call can chat with those in the chat room and vice versa. Overall, I can say that I understood the concept of agile to a greater depth when I got under pressure and how important it is to adapt and deliver.

2.5. Operating Environment

Operating environment for the airline management system is as listed below.

- client/server system.
- Operating system: Windows.
- Works with any browser which supports JavaScript, react and html.
- Hosted on Heroku. (may respond slow sometimes)

2.6. Path

The flow of the application is as shown below: