Database Mini Project Report

on

“REALTIME CODE COLLABORATION PLATFORM”

Submitted by

33207 Atharv Chavan

33215 Pratik Daigavane

33209 Raman Bhandari

33211 Nihal Bora

(TE-10)

Under the guidance of

Mr Swapnil Mane

Department of Information Technology

Pune Institute of Computer Technology College of Engineering

Sr. No 27, Pune-Satara Road, Dhankawadi, Pune - 411 043.

A.Y. 2020-2021

CERTIFICATE

This is to certify that the mini project report entitled **"REALTIME CODE COLLABORATION PLATFORM"** being submitted by**, Atharv Chavan(33207), Raman Bhandari (33209), Nihal Bora (33211), Pratik Daigavane (33215)** is a record of bonafide work carried out by them under the supervision and guidance of **Mr Swapnil Mane** in partial fulfilment of the requirement for **TE (Information Technology Engineering) – 2015 course** of Savitribai Phule Pune University, Pune in the academic year 2020-2021.

Date: 29-11-2020

Place: Pune

Guide	Subject Coordinator	Head of Department
Mr Swapnil Mane	Prof Ravi Murumkar	Dr. A. M. Bagade

Principal

Dr P. T. Kulkarni

CONTENT

4. System Design
    - ER Model
    - Schema Description
    - Tables Description
    - System Flow chart / Activity diagram
    - User Interface Design
    - Validations

5. System Implementation
    - Hardware and Software Platform description
    - Tools used
    - Future work / Extension
    - Conclusion References

Abstract:

This pandemic has hit us hard, but with time everything is starting to come online so is the case with our colleges. All Coding Practicals are being conducted online, mostly through PPTs and screen sharing, while this way is relevant for theory lectures but it becomes hard to coupe up with online Practicals. On our platform teacher will create a room, will share invite link. After the room is created a Linux container will be created and all users will be able to access that Linux environment. Students will join the room, which will put them and the teacher in the same coding environment. Now they can create files and start coding in realtime using our online code editor, including syntax highlighting and all other features.

Acknowledgement:

Firstly, we would like to thank our teacher and guide
Mr Swapnil Mane who gave us his valuable suggestions and ideas when we needed them. He encouraged us to work on this project.

We are also grateful to our college for allowing us to work with them and providing us with the necessary resources for the project. We would also like to thank all of them who helped us to complete this project. We are immensely grateful to all involved in this project as without their inspiration and valuable suggestions it would not have been possible to develop the project within the prescribed time.

Introduction:

1.1 Purpose:

- All the students do not have the same coding environment and some of them don't even have Linux installed.
- If the student wants to share code or even if the teacher wants to share a code, they need to send a file which then needs to be run locally, hence consuming more time.
- Debugging and doubt solving in a code is cumbersome and the teacher cannot help the student in realtime.
- A poor network connection is the biggest hurdle of all due to which quality of video share may be bad and code may not be visible clearly.

1.2 Scope:

- Efficient Code Sharing In real-time
- Access to the same Linux environment.
- Dedicated Linux terminal
- Communicate through text and voice chat.
- Start coding in realtime using the online code editor
- Communications through web sockets
- Less bandwidth consumption

1.3 Definition, Acronym, and Abbreviations:

- DBMS: Database management system
- NoSQL: Non-SQL
- DB: Database
- JS: JavaScript

1.4 References:

- https://repl.it/
- https://www.sciencedirect.com/science/article/pii/S1877050915020608
- https://webdesign.tutsplus.com/articles/real-time-code-collaboration-tools-for-developers--cms-30494

1.5 Developers' Responsibilities:

An Overview:

- Organizing Data
- Processing Queries
- Concurrency control
- Security
- Managing Docker Container

General Description:

2.1 Product Function Perspective:

Execute-IT is a web application that provides workspace to writing, perform, display the results of the code through the terminal, and collaborate with other users in real-time. The application main features are providing workspace to make, execute and build the source code, real-time collaboration, chat, and build the terminal. This application supports C, C++, JS, Python and many more programming languages.

2.2 User Characteristics:

There are 2 types of users in the rooms.

- **Normal Participant:** A user in a room which is not created by him but by another user, therefore not a Host of the room.

- **Host User:** A user in a room which is created by him where other users can join in to collaborate on projects/

2.3 General Constraints:

- If we see the performance constraints, as we are using the React.js and MongoDB, the performance will be on top. Again, the web interface is very user friendly and the user can easily manipulate it.

2.4 Assumptions and Dependencies:

- It is assumed that this software will be available on every device and it will be platform-independent.

- User has a Gmail account to login into the system.

Specific Requirements:

3.1 Inputs and Outputs:

The interface of Execute It is very user friendly. A user can join a room or create a room. In the workspace area, user and create, modify and delete files. Those files can also be executed and the output is shown there itself.
The Voice and text chat functionality is provided for communication between the members of the room.

3.2 Functional Requirements:

On our platform, the functional requirements are OAuth login system, Room and Linux Environment creation, Realtime editing of files by multiple users, voice chat, text chat and support for multiple programming languages. If any new user comes then the system will check whether that user already exists or not otherwise the system will add it in the database.

3.3 Functional Interface Requirements:

The website interface is very user friendly.
A user can create a room and easily add more users to the room using the invite code. Any user can create a file and also all users can edit the data in the file at the same time. They can also communicate through voice and text channel.

3.4 Performance Constraints:

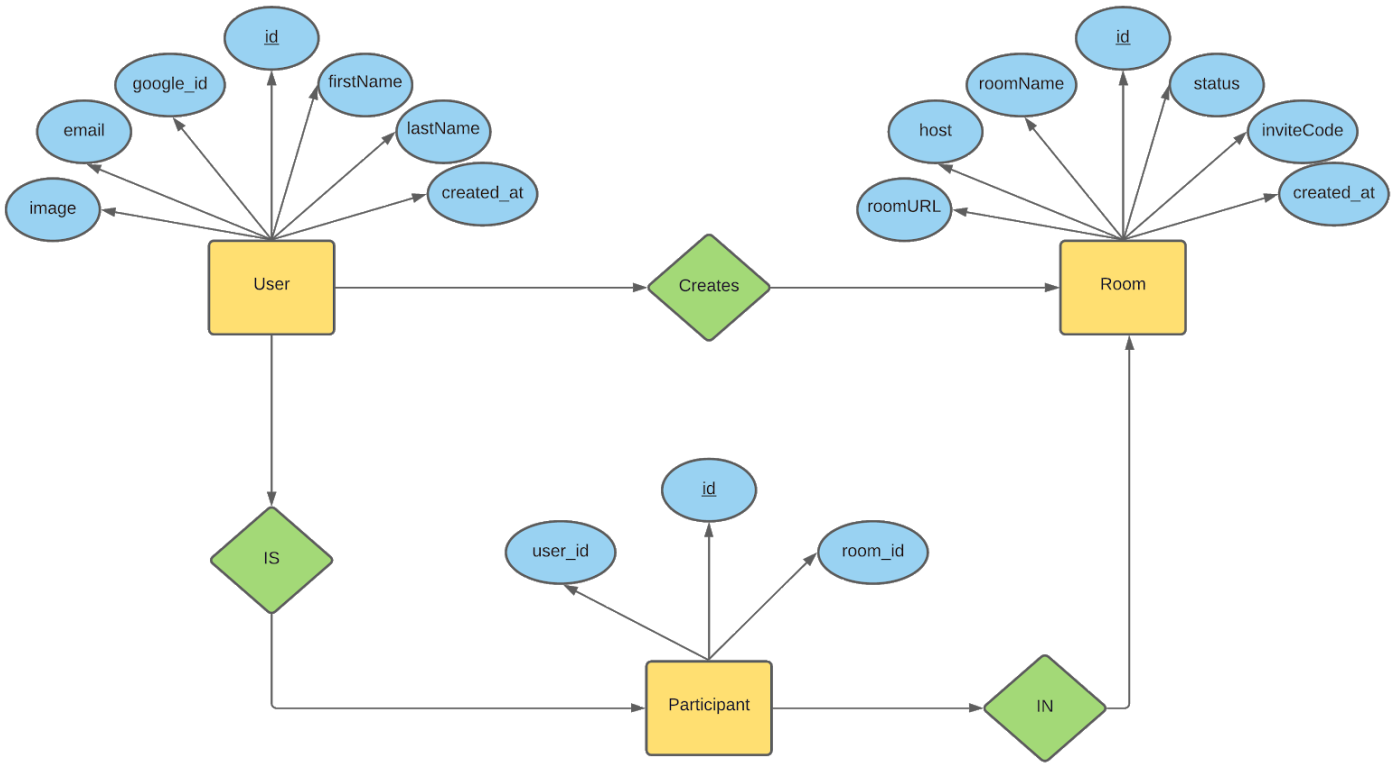
On this site, we have used React.js and MongoDB. MongoDB's document model allows virtually any kind of data structure to be modelled and manipulated easily. MongoDB's BSON data format, inspired by JSON, allows you to have objects in one collection have different sets of fields. Due to these immense advantages of MongoDB and React.js is an open-source JavaScript library that is used for building user interfaces specifically for single-page applications. The architecture of our platform is highly robust and scalable, hence it can perform exceptionally even with low specifications hardware

3.5 Acceptance criteria:

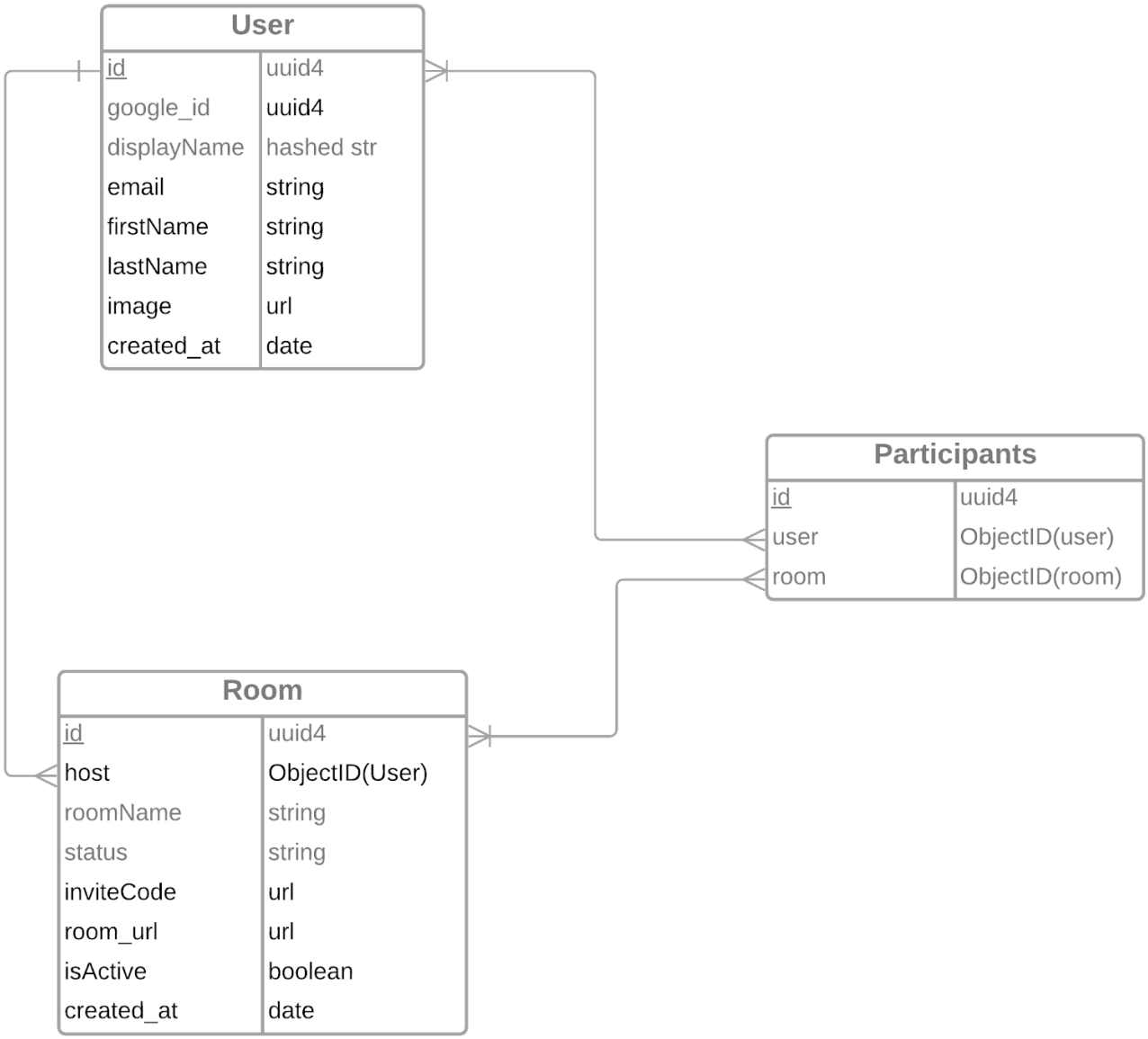
A user cannot enter a room without signing in.
Also, the user should have a Gmail account to sign in. Also if the user wants to enter a room he/she should enter invite code to join the room.

System Design:

4.1 ER Model:



4.2 Schema Description:



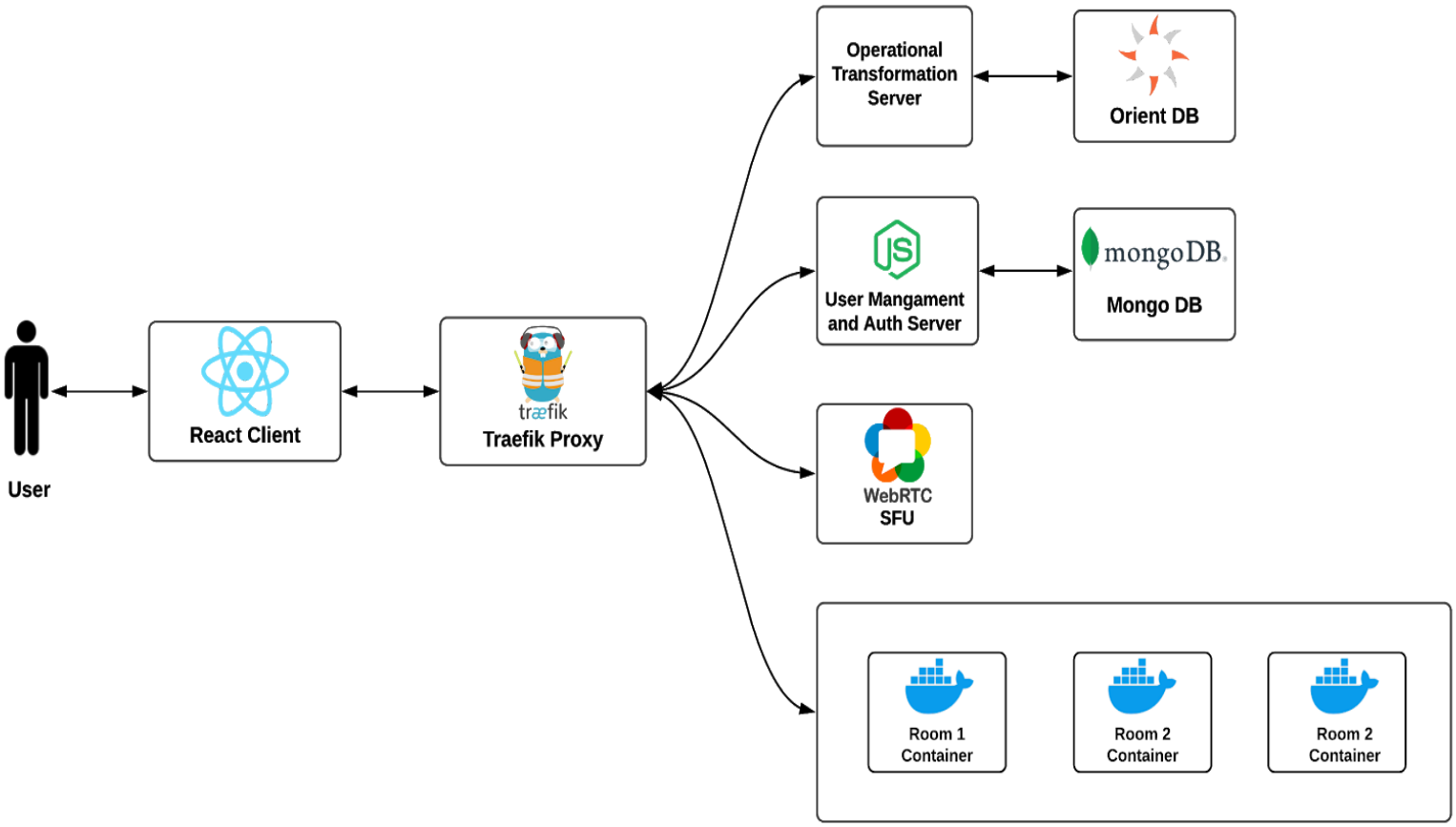
4.3 Collection Description

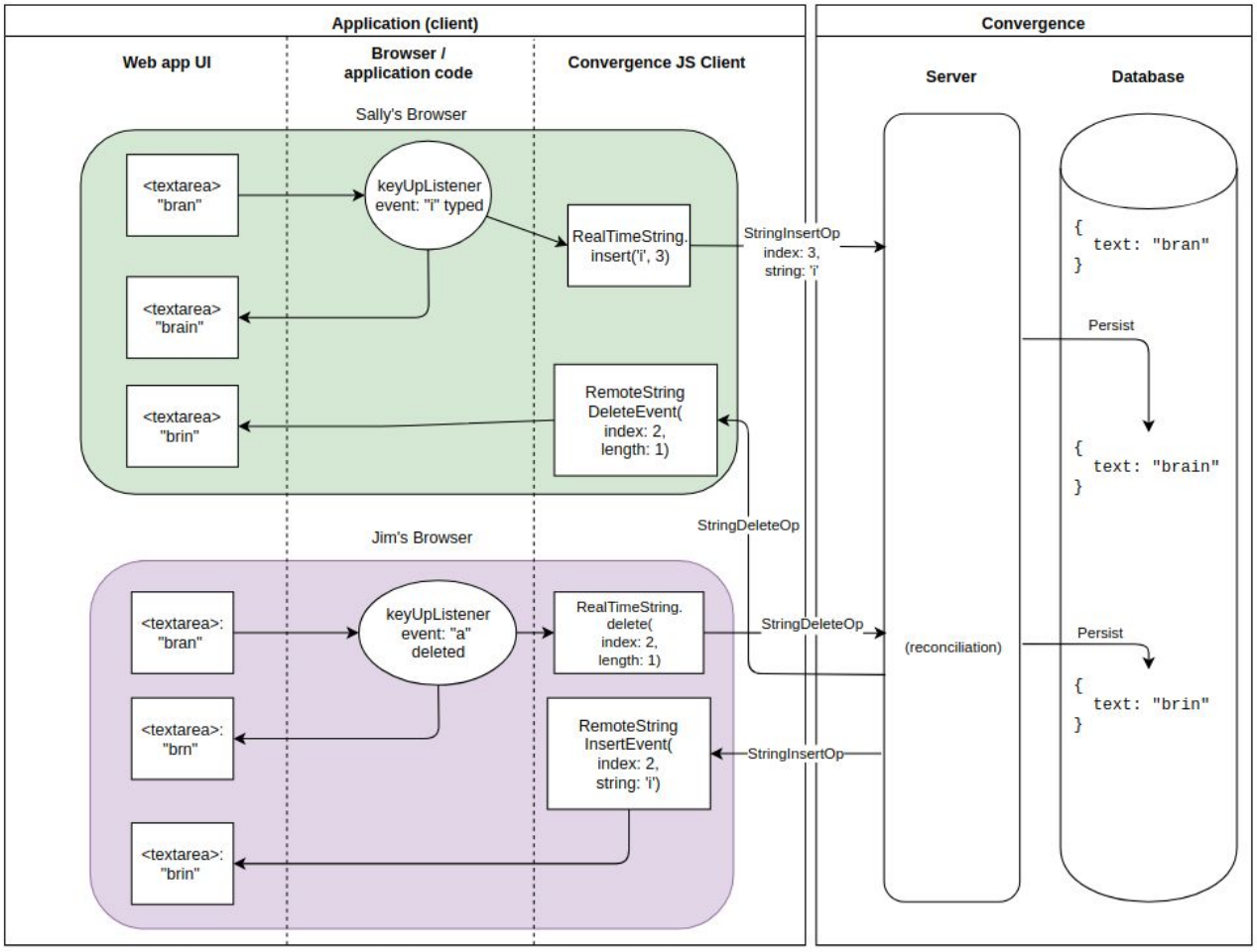
```
let RoomSchema = new mongoose.Schema({
    _id: {
        type: String,
        default: function genUUID() {
            return uuidv4()
        },
        primaryKey: true

    },
    roomName: {
        type: String,
        required: true,
        unique: true
    },
    status: {
        type: String,
        default: 'room_created'
    },
    inviteCode: {
        type: String,
        default: function() {
            return uuidv4()
        },
        unique: true
    },
    host: {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'User',
        required: true,
    },
    createdAt: {
        type: Date,
        default: Date.now()
    },
    roomURL: {
        type: String
    },
    isActive: {
        type: Boolean,
        default: true
    }
})
```

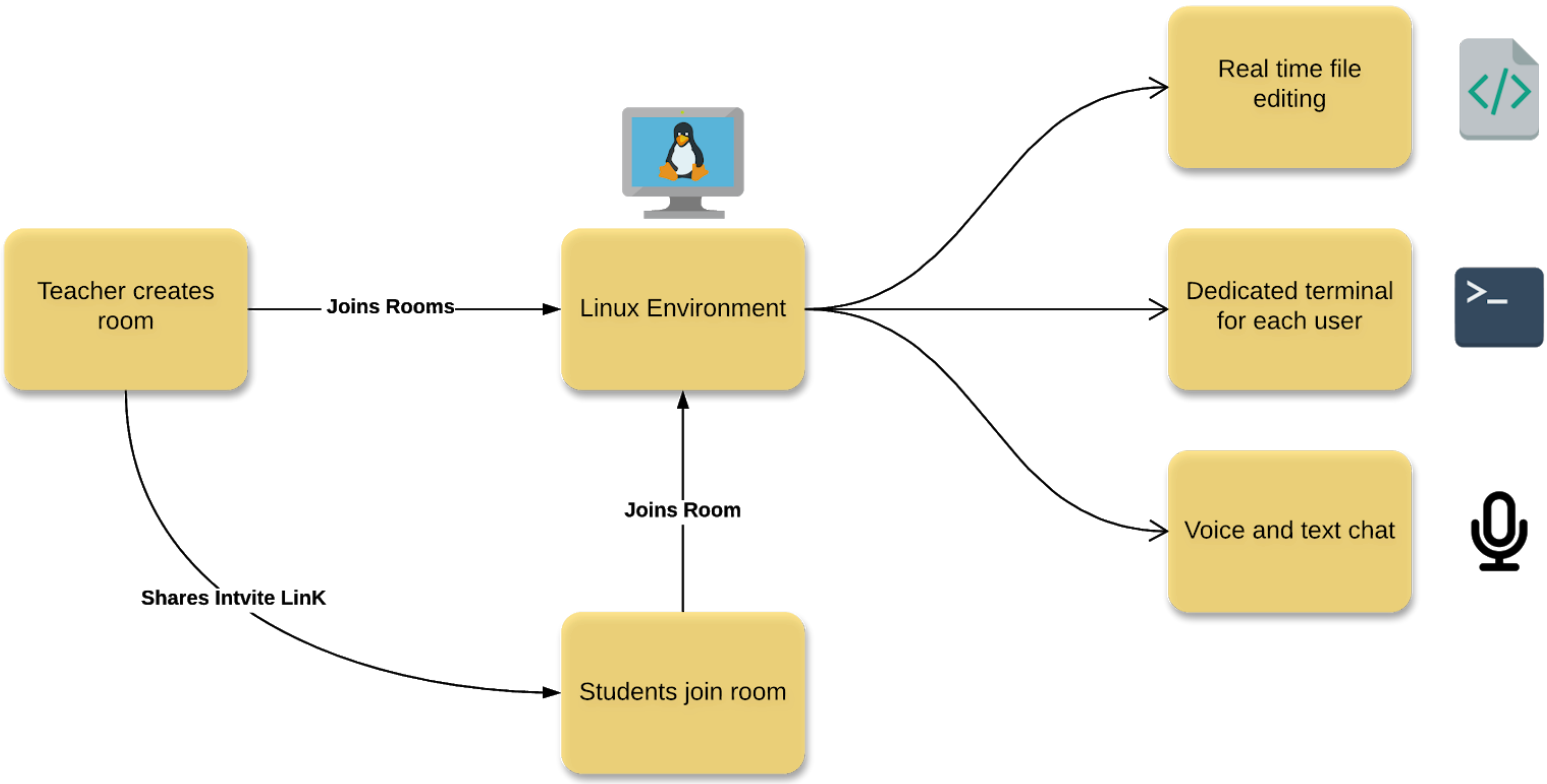
```
let UserSchema = new mongoose.Schema({
    googleId: {
      type: String,
      required: true
    },
    displayName: {
        type: String,
        required: true
    },
    firstName: {
        type: String,
        required: true
    },
    lastName: {
        type: String,
        required: true
    },
    image: {
        type: String,
    },
    email: {
        type: String,
        required: true,
        unique: true,
        lowercase: true,
        validate: (value) => {
            return validator.isEmail(value)
        }
    },
    createdAt: {
        type: Date,
        default: Date.now()
    }
})
```

4.4 System Architecture/Activity diagram:



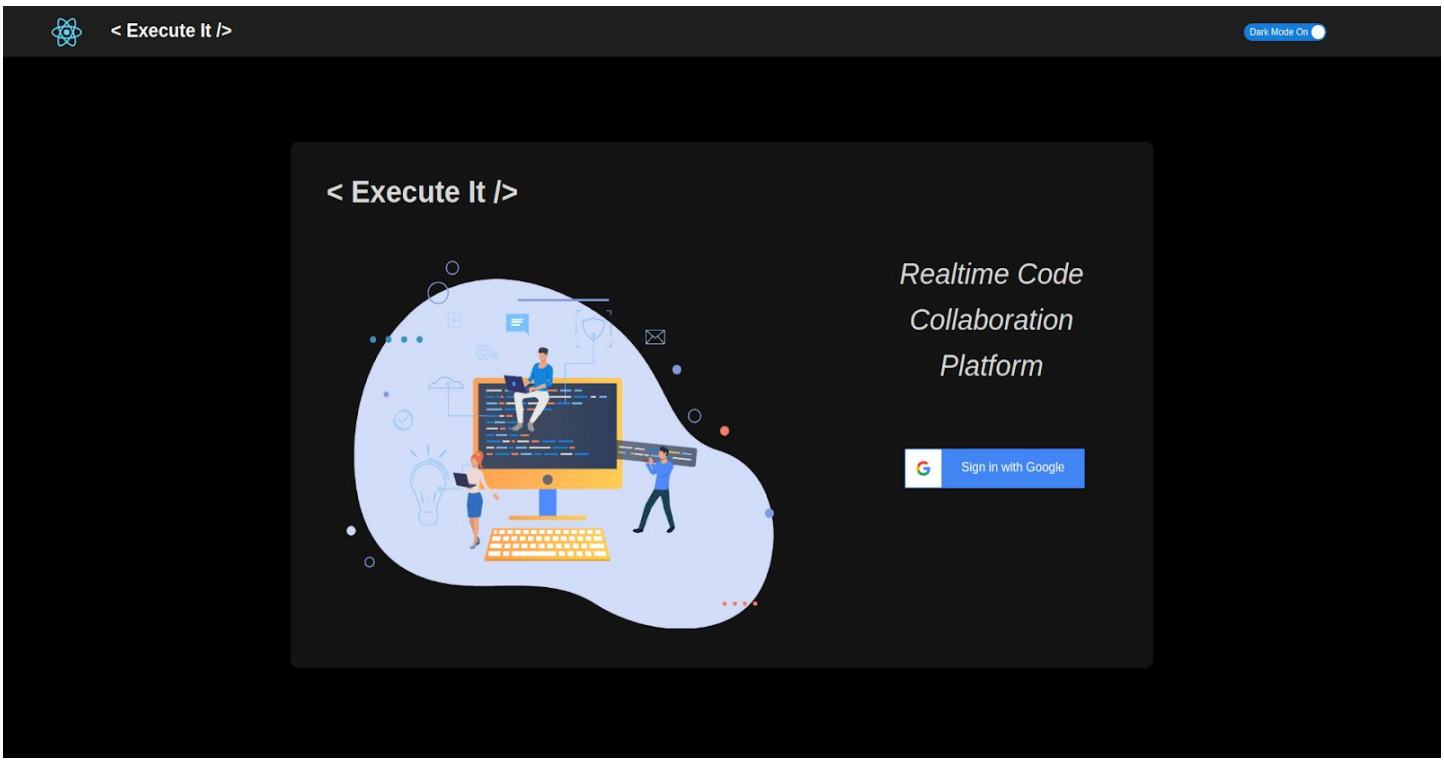

Application (client)
Convergence
Web app UI
Browser / application code
Convergence JS Client
Server
Database
Sally's Browser
<textarea> "bran"
keyUpListener event: "i" typed
RealTimeString. insert('i', 3)
StringInsertOp index: 3, string: 'i'
<textarea> "brain"
<textarea> "brin"
RemoteString DeleteEvent(index: 2, length: 1)
{ text: "bran" }
Persist
{ text: "brain" }
StringDeleteOp
Jim's Browser
<textarea>: "bran"
keyUpListener event: "a" deleted
RealTimeString. delete(index: 2, length: 1)
StringDeleteOp
(reconciliation)
Persist
{ text: "brin" }
<textarea>: "brn"
RemoteString InsertEvent(index: 2, string: 'i')
StringInsertOp
<textarea>: "brin"

4.5 Proposed Solution:

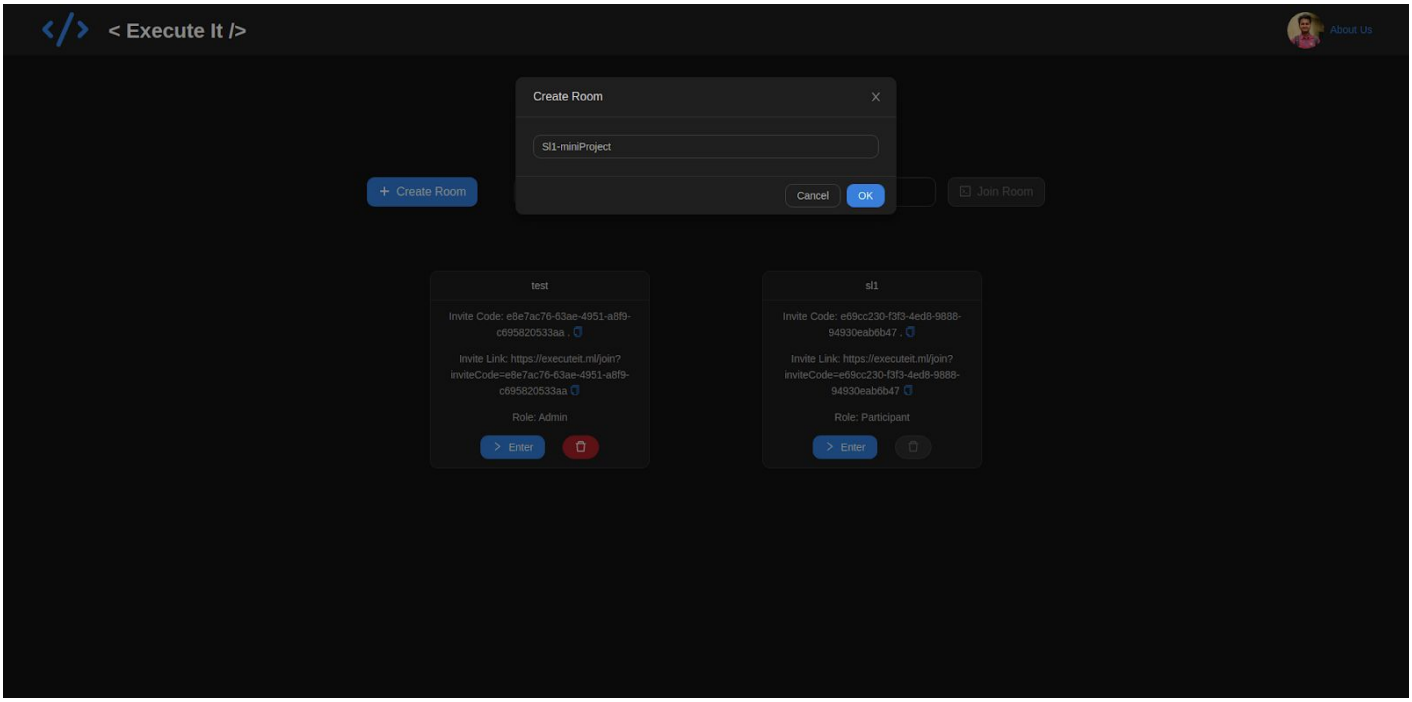


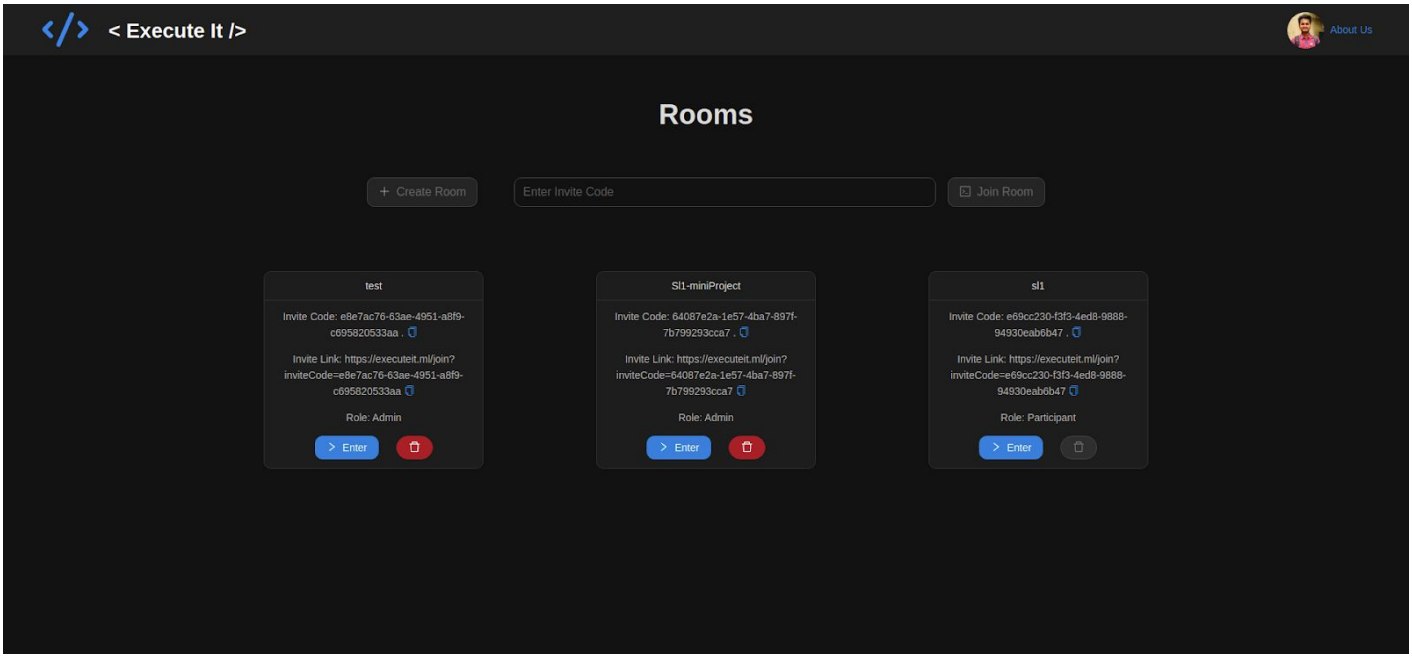
4.6 User Interface Design:

- **Home page:**

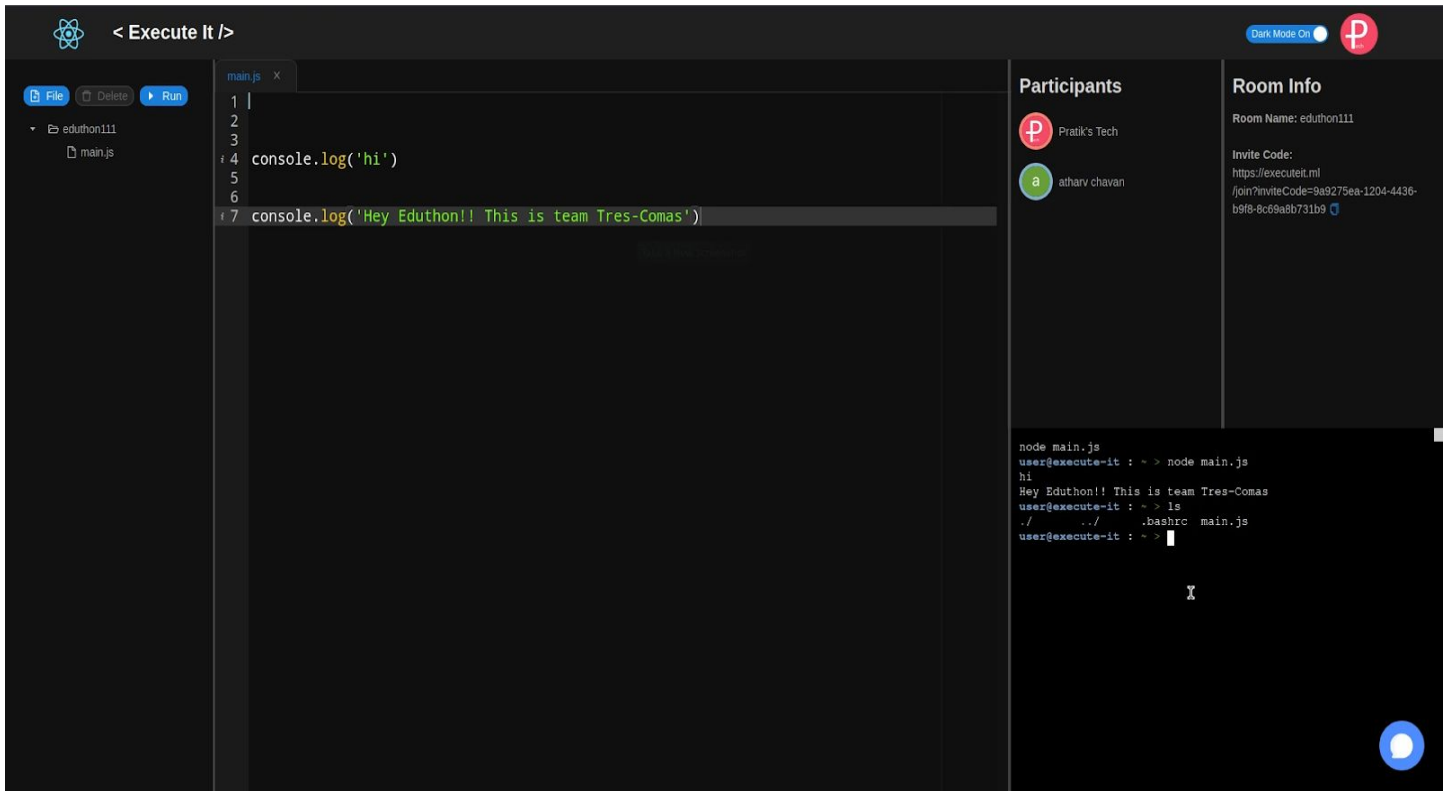


- **Create Room Page:**

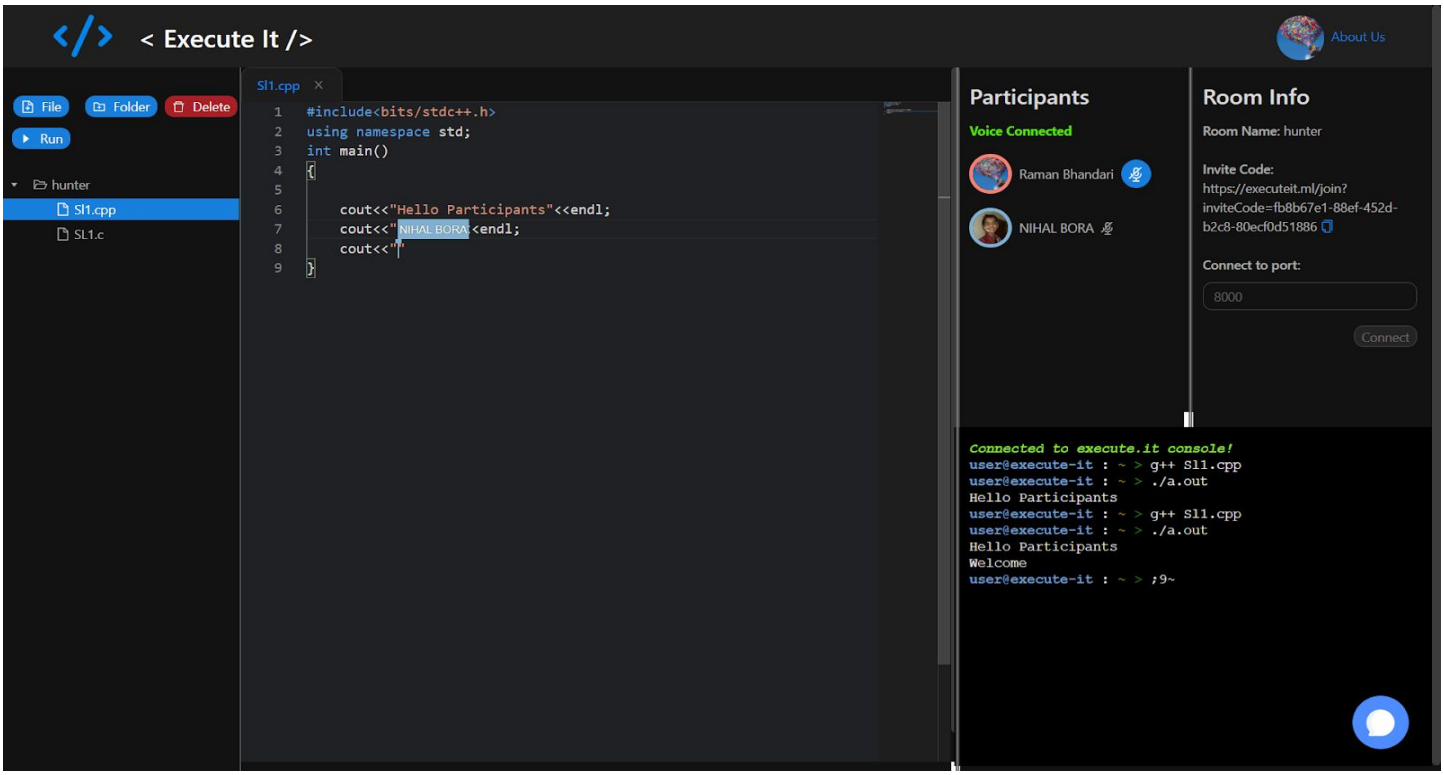




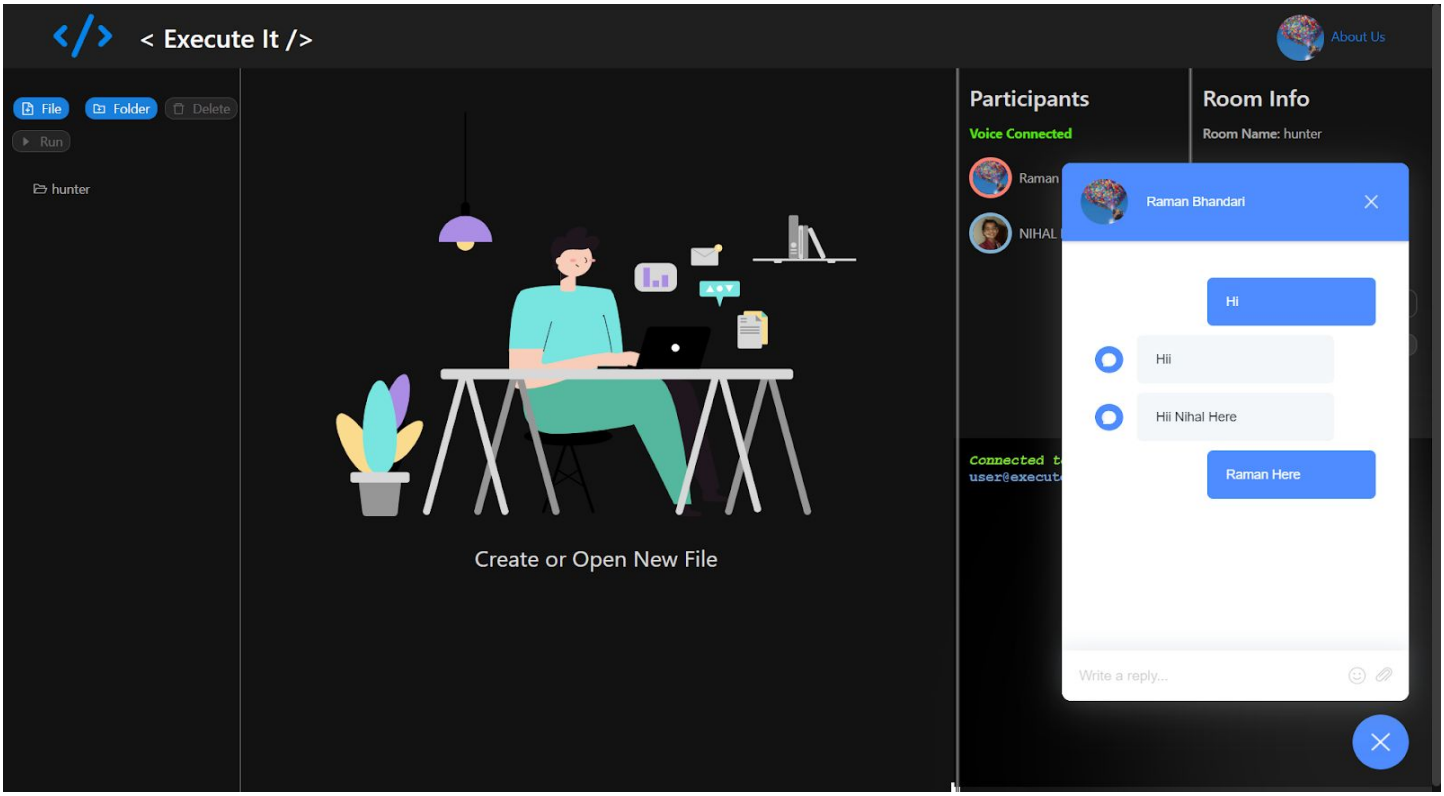
- **Workspace Page 1:**



- **Workspace Page 2:**

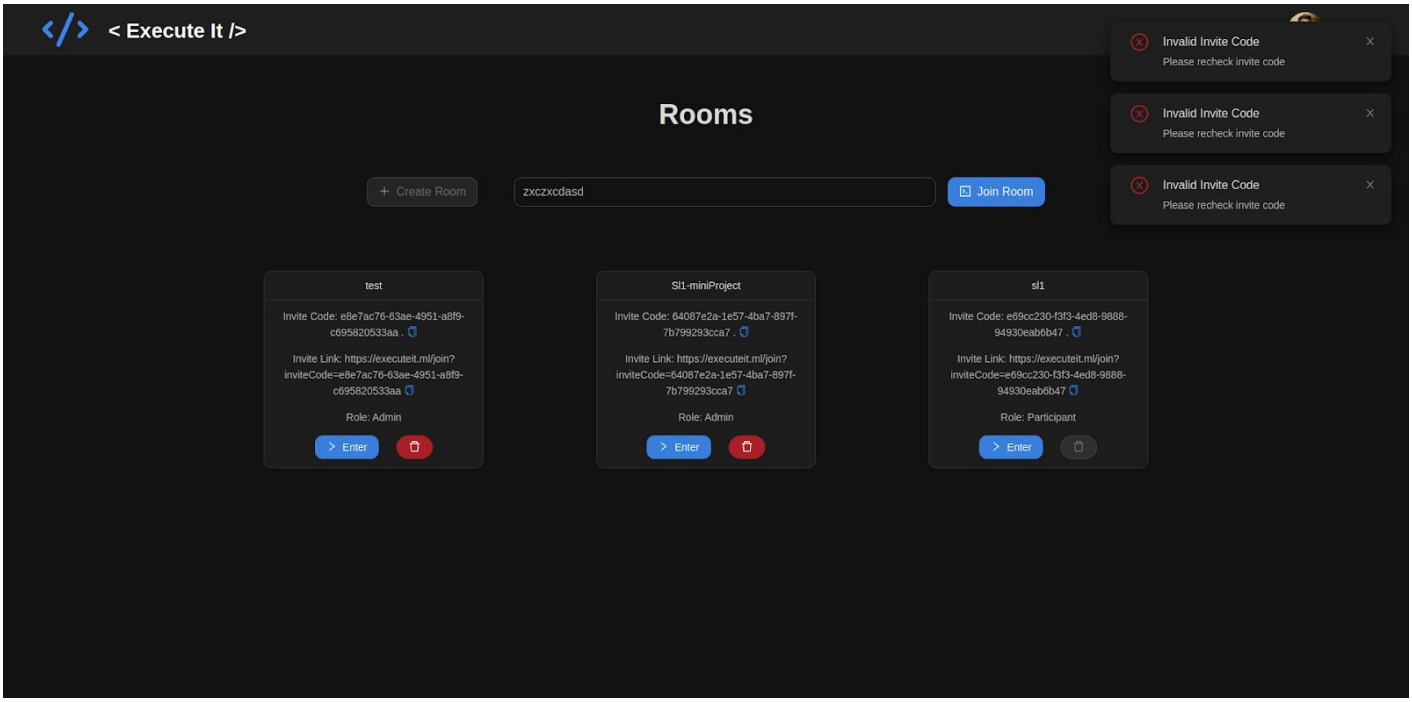


- **Chat Box Component:**



4.7 Validations:

- The names of rooms are unique, hence while creating a new room it is taken into account.
- In the Real-Time Code Sharing Platform, when a new user arrives then might navigate to any window. So, to avoid that, he/she must log in through google account and he/she navigates through tabs.
- On the same, it will also deliver the message that the user is editing the code.



System Implementation:

5.1 Hardware and Software Platform description:

- Hardware: PC
- Software:
    - MongoDB,
    - ReactJS,
    - NodeJS,
    - Docker,
    - Digital Ocean

5.2 Tools Used:

- Node JS
- React JS
- Ant D
- Xterm JS
- MongoDB
- Traefik
- Convergence
- Docker
- Github Actions
- Digital Ocean
- Netlify

5.3 Future Work/ Extension:

- **Our platform overcomes problems with the current way of conducting online practicals and classes.**
- Students and the teachers will now be in the same coding environment.
- Consumes less bandwidth than online video calls.
- In the near future, it can be extended to newer versions after performance improvement.
- Data science platform can also be integrated into our system.

5.4 Conclusion:

In this mini-project, we have successfully designed and implemented the Real-Time Code Sharing Platform using React.js as a front-end and MongoDB as a backend.