# Software Engineering Mini Project Stage-II

## Project Report

Submitted by

**1. Atharva Marathe (111903024)**
**2. Avishkar Andhale (111903025)**

Mrs. Tanuja Pattanshetti
(Instructor)

**Department of Computer Engineering,**
**College of Engineering Pune**

# Table of Contents

## 1. Problem Statement

To provide a web-based collaborative platform for coding.

Building software projects by students, companies working remotely is common these days. Hence collaborative tools for coding are essential while working on group or team-based projects. Lot of Collaborative editing tools are available on the internet, but most of them are Rich Text based editors. CollabPro provides collaborative experience for coders with syntax highlighting, auto-suggestions features. This product makes it easy and convenient for teams working on a common project.

## 2. Objectives

- To provide an online collaborative tool for coding.
- To provide tools for building software projects collaboratively on the web.
- Support all the necessary tools and features that are provided by a code editor
- To give a smooth lag-free collaborative experience for the users.
- Provide necessary security features to the end-users.

## 3. Motivation

- Building software projects by students and by companies by working remotely is common these days. Hence collaborative tools for coding are essential while working on group or team-based projects.
- A Lot of collaborative tools are available on the internet which provide the facility of collaborative coding.
- But most of them do not focus on the features needed for building a project online
- CollabPro application provides solution of this problem.
- Operational transformation is a technology for supporting a range of collaboration functionalities in advanced collaborative software systems.
- By using OT library and socket programming, efficient collaborative environment can be created.

- This tool enables users to do coding effectively within a group / team.

# 4. Summary of SRS

## 4.1 Purpose of SRS

The purpose of this document is to give a detailed description of the requirements for "CollabPro – Collaborative Code Editor". It will illustrate the purpose and complete declaration for the development of system. It will also explain system constraints, interface and interactions with other external applications.

## 4.2 Document Convention

To build this template we followed IEEE standard Software Requirements Specification.

## 4.3 Intended audience Reading Suggestions

This document is made to guide developers and customers so that they can suggest changes in further scope of improvement. Also, this document specifies use case diagram, product scope, product functions, functional and non-functional requirements and technical issues related with the product.

The software requirement specification (SRS) document is written for a more general audience. This document is intended for individuals directly involved in the development of Collaborative Code Editor. It also includes software developers, project consultants, and team managers.

## 4.4 Product Functions

The main function of our product is to provide a platform to users, to code and complete the tasks online, without the need to install apps on the computer. It enables users to easily collaborate with team members in real-time. Users can communicate with each other through chat section.

## 4.5 User Classes and Characteristics

There are mainly two users for this software:

1. Host: The host creates coding sessions, invites other participants / team-members. The host has the rights to use all the features of the application.

2. Participant: The participant joins the session created by the host. The participant can use features of the application depending upon the session privileges granted by the host.

## 4.6 Design and Implementation Constraints

In order to use the application smoothly, a stable internet connection is required. As the number of users increase, the traffic congestion may become severe, leading to decrease in performance. As the files are temporarily stored on the server, the server needs to have sufficient hard-disk space.

## 4.7 User Documentation

With the application separate user manual will be provided that will help the user to get familiar with the platform.

## 4.8 User Interfaces

Various user interfaces for the product could be:

- Login Page
- Home Page
- Session Dashboard
- Code Editor Window

## 4.9 Hardware Requirements

- Processor
- RAM
- Hard Drive
- Internet Connection

## 4.10 Software Requirements

- React: Front-end JS library to be used for frontend work
- MongoDB: To be used to store databases
- Express: Back-end JS library to design web application
- Node.js: Back-end JavaScript runtime environment.
- Web Browser

## 4.11 Communications Interfaces

HTTP connections will be setup between client and server of the application. Socket IO library will be used to implement real-time collaboration.

## 4.12 Performance, Safety and Security Requirements

Real-time code editors require minimum latency, as many users may edit the contents simultaneously. As the number of users increase, the server must be able to handle the traffic congestion efficiently.

User-data should be stored temporarily in the server, in case if the user gets disconnected, and should be able to resume from where it left.

Only authentic and authorized must be admitted to the session. The user data must be kept secure during the session. All the personal information of the user, passwords must be encrypted to prevent any kind of piracy.

## 4.13 References

1] Operational Transformation: https://srijancse.medium.com/how-real-time-collaborative-editing-work-operational-transformationac4902d75682/

2] Socket Programming: https://socket.io/

3] Code-Mirror https://codemirror.net/

## 5. ER Diagram

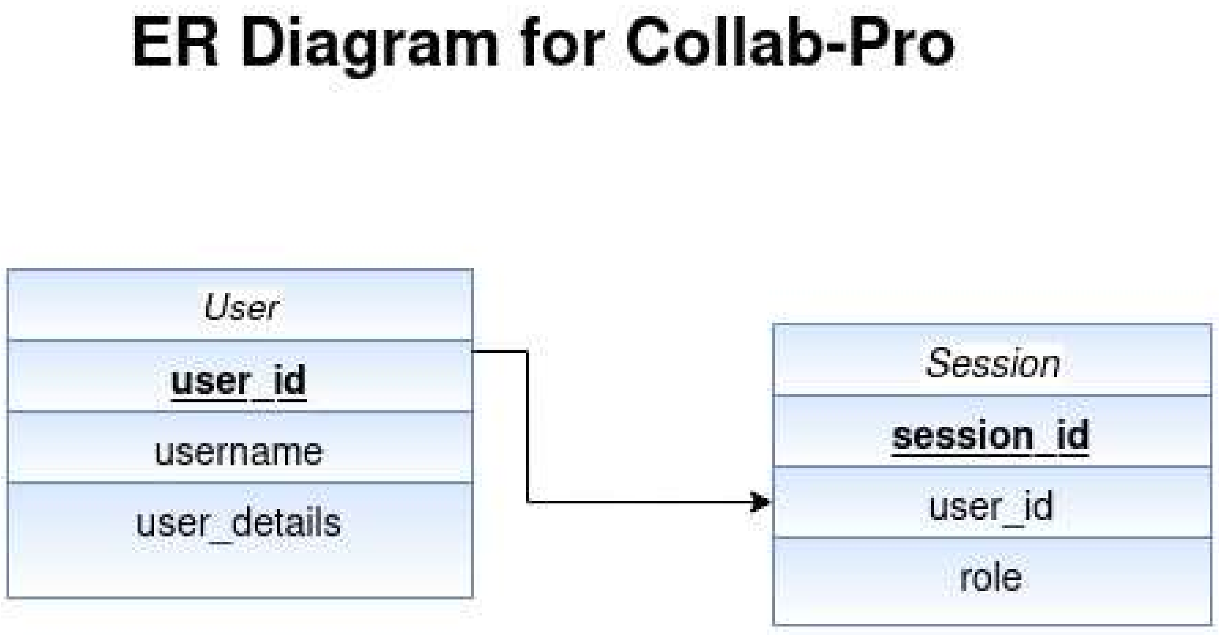
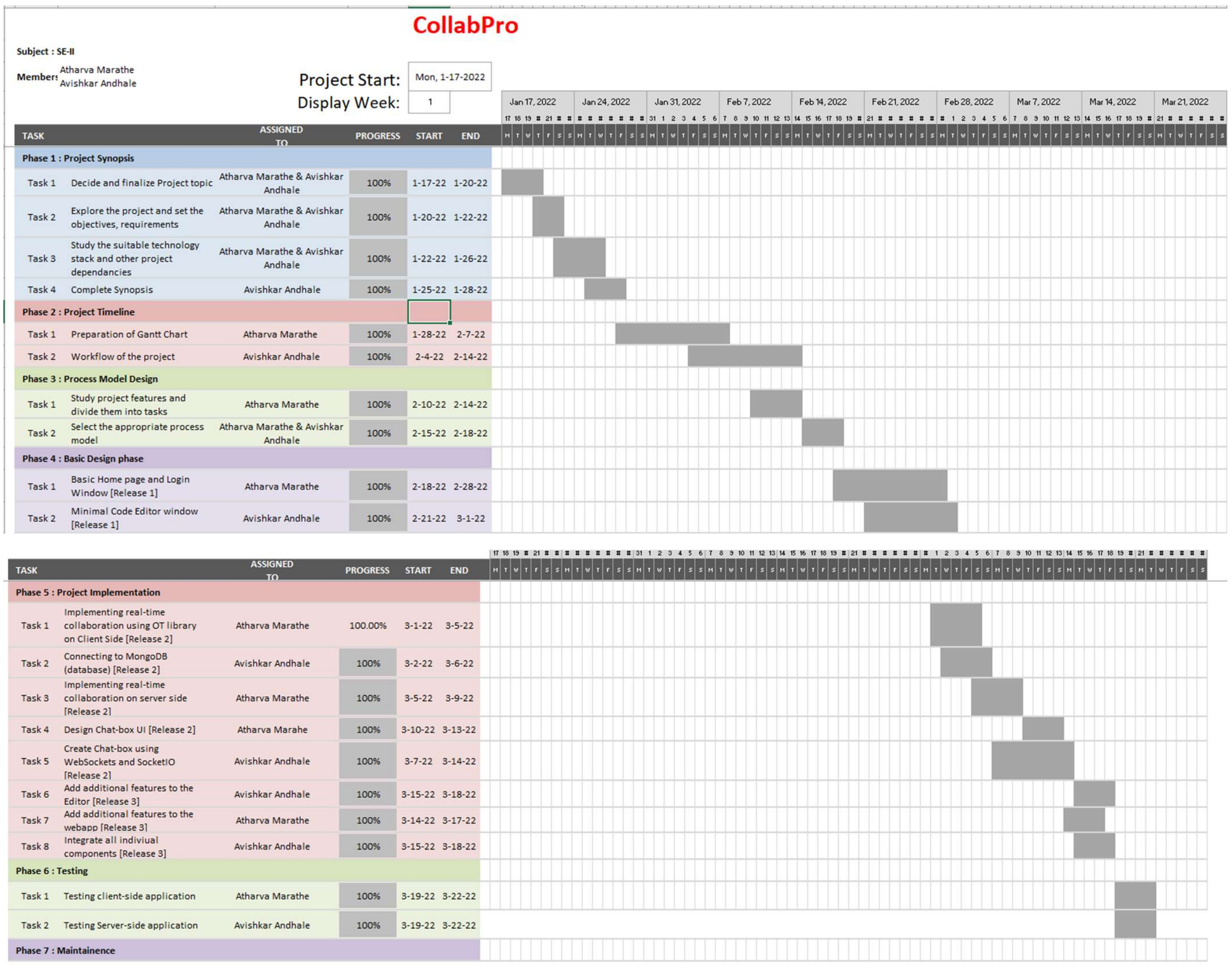


Figure 1.1 ER-DIAGRAM.

## Gantt Chart



CollabPro

Subject : SE-II

Members: Atharva Marathe, Avishkar Andhale

Project Start: Mon, 1-17-2022

Display Week: 1

| TASK | | ASSIGNED TO | PROGRESS | START | END |
|---|---|---|---|---|---|
| **Phase 1 : Project Synopsis** | | | | | |
| Task 1 | Decide and finalize Project topic | Atharva Marathe & Avishkar Andhale | 100% | 1-17-22 | 1-20-22 |
| Task 2 | Explore the project and set the objectives, requirements | Atharva Marathe & Avishkar Andhale | 100% | 1-20-22 | 1-22-22 |
| Task 3 | Study the suitable technology stack and other project dependancies | Atharva Marathe & Avishkar Andhale | 100% | 1-22-22 | 1-26-22 |
| Task 4 | Complete Synopsis | Avishkar Andhale | 100% | 1-25-22 | 1-28-22 |
| **Phase 2 : Project Timeline** | | | | | |
| Task 1 | Preparation of Gantt Chart | Atharva Marathe | 100% | 1-28-22 | 2-7-22 |
| Task 2 | Workflow of the project | Avishkar Andhale | 100% | 2-4-22 | 2-14-22 |
| **Phase 3 : Process Model Design** | | | | | |
| Task 1 | Study project features and divide them into tasks | Atharva Marathe | 100% | 2-10-22 | 2-14-22 |
| Task 2 | Select the appropriate process model | Atharva Marathe & Avishkar Andhale | 100% | 2-15-22 | 2-18-22 |
| **Phase 4 : Basic Design phase** | | | | | |
| Task 1 | Basic Home page and Login Window [Release 1] | Atharva Marathe | 100% | 2-18-22 | 2-28-22 |
| Task 2 | Minimal Code Editor window [Release 1] | Avishkar Andhale | 100% | 2-21-22 | 3-1-22 |

| TASK | | ASSIGNED TO | PROGRESS | START | END |
|---|---|---|---|---|---|
| **Phase 5 : Project Implementation** | | | | | |
| Task 1 | Implementing real-time collaboration using OT library on Client Side [Release 2] | Atharva Marathe | 100.00% | 3-1-22 | 3-5-22 |
| Task 2 | Connecting to MongoDB (database) [Release 2] | Avishkar Andhale | 100% | 3-2-22 | 3-6-22 |
| Task 3 | Implementing real-time collaboration on server side [Release 2] | Atharva Marathe | 100% | 3-5-22 | 3-9-22 |
| Task 4 | Design Chat-box UI [Release 2] | Atharva Marahe | 100% | 3-10-22 | 3-13-22 |
| Task 5 | Create Chat-box using WebSockets and SocketIO [Release 2] | Avishkar Andhale | 100% | 3-7-22 | 3-14-22 |
| Task 6 | Add additional features to the Editor [Release 3] | Avishkar Andhale | 100% | 3-15-22 | 3-18-22 |
| Task 7 | Add additional features to the webapp [Release 3] | Atharva Marathe | 100% | 3-14-22 | 3-17-22 |
| Task 8 | Integrate all indiviual components [Release 3] | Avishkar Andhale | 100% | 3-15-22 | 3-18-22 |
| **Phase 6 : Testing** | | | | | |
| Task 1 | Testing client-side application | Atharva Marathe | 100% | 3-19-22 | 3-22-22 |
| Task 2 | Testing Server-side application | Avishkar Andhale | 100% | 3-19-22 | 3-22-22 |
| **Phase 7 : Maintainence** | | | | | |

Figure 1.2 Process Workflow

# 6. UML and Explanation

## 6.1 Use Case Diagram

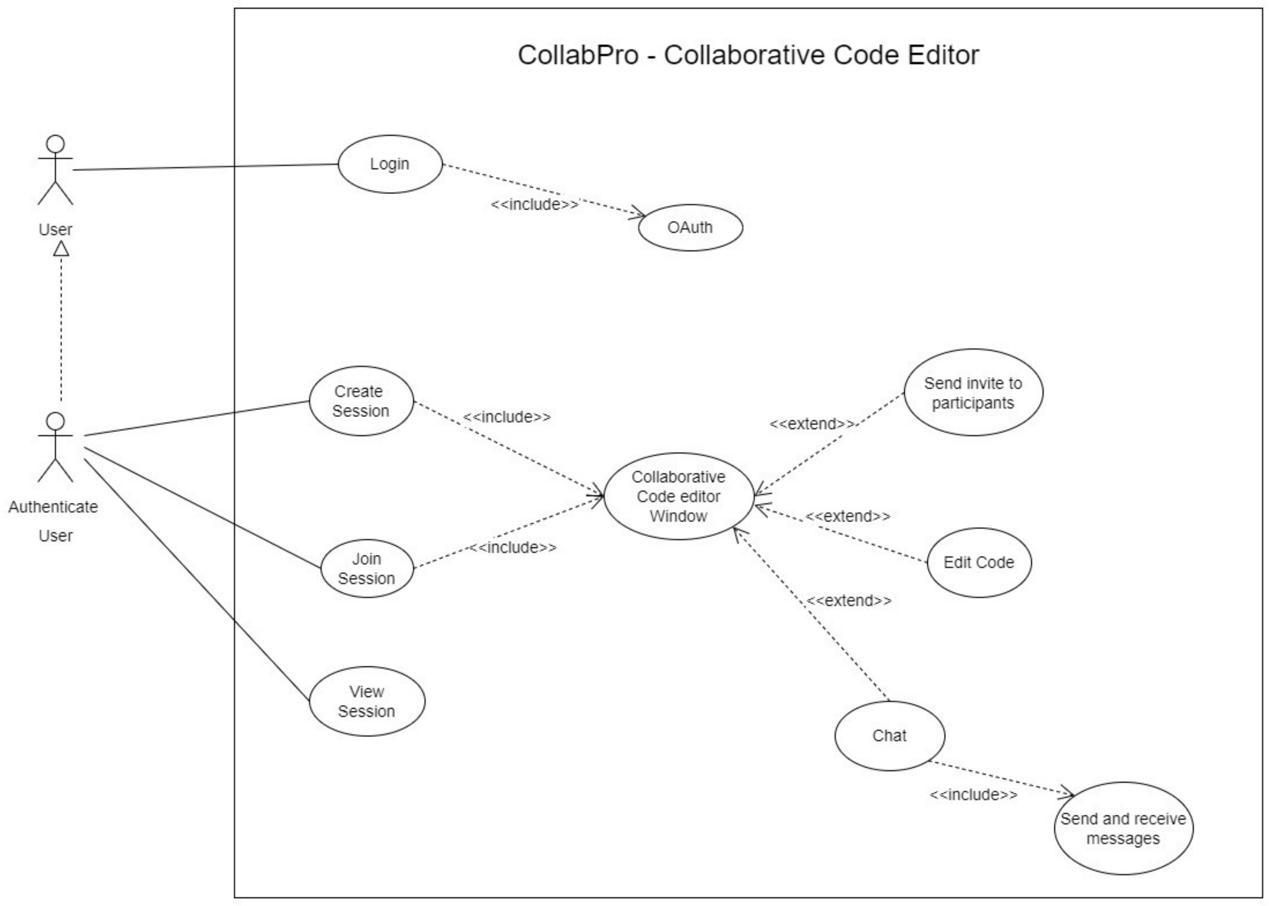

CollabPro - Collaborative Code Editor
User
Authenticate User
Login
OAuth
<<include>>
Create Session
Join Session
View Session
Collaborative Code editor Window
Send invite to participants
Edit Code
Chat
Send and receive messages
<<extend>>

Figure 1.4 Use Case Diagram

## 6.2 Class diagram.

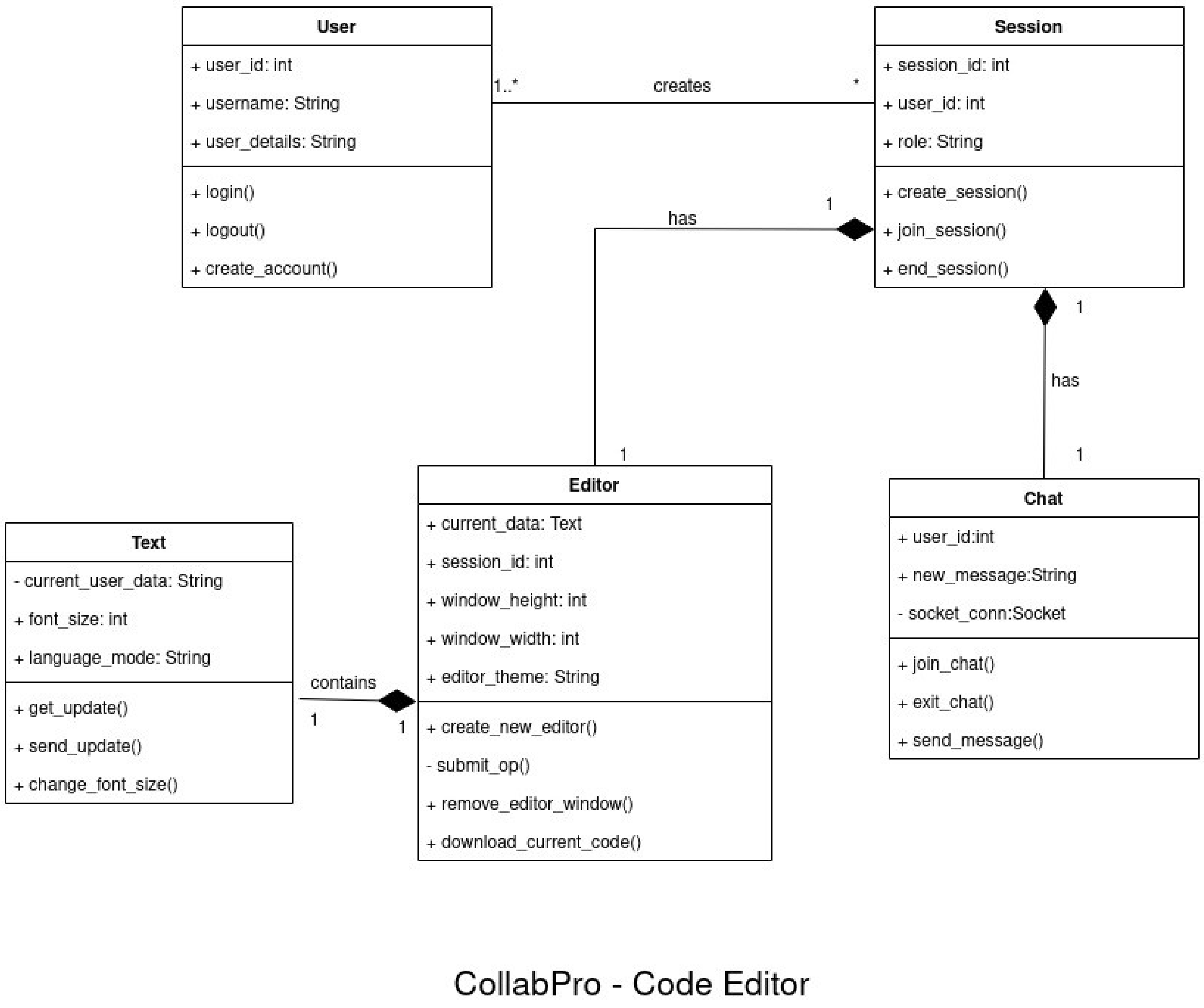


Figure 1.5 Class Diagram

## 6.3 State Diagram

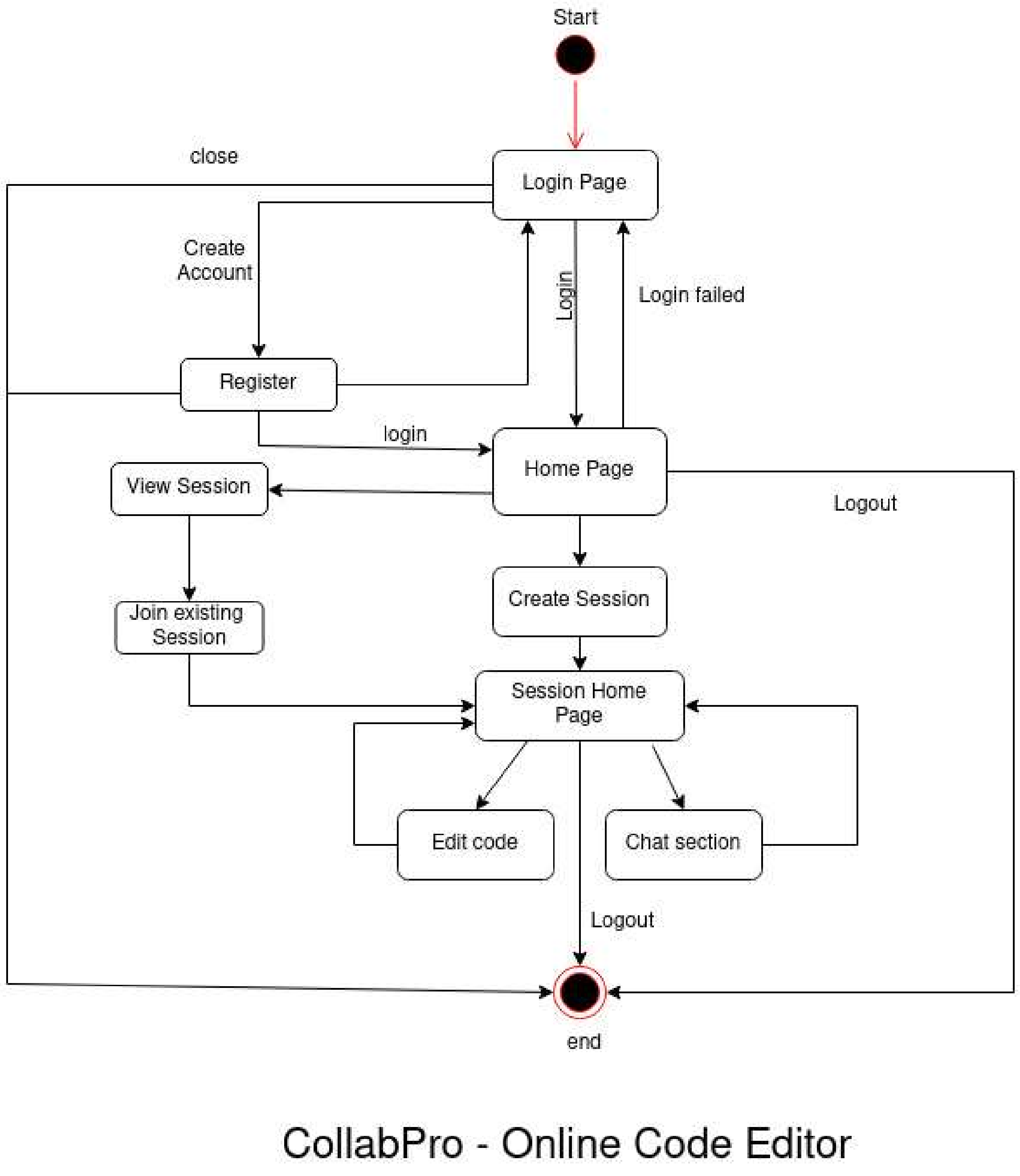


Figure 1.6 State Diagram

## 6.4 Sequence Diagram

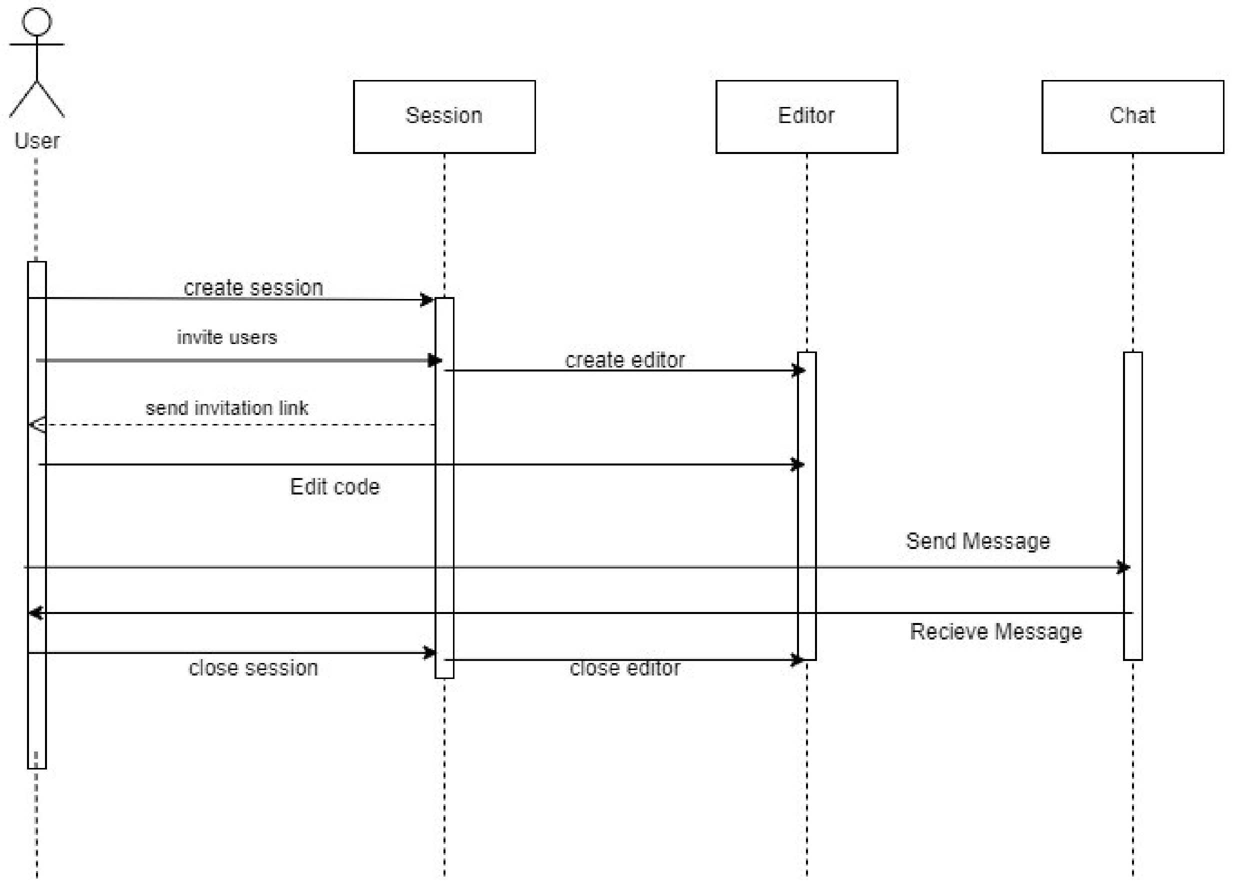


Figure 1.7 Sequence Diagram

## 6.5 Activity Diagram

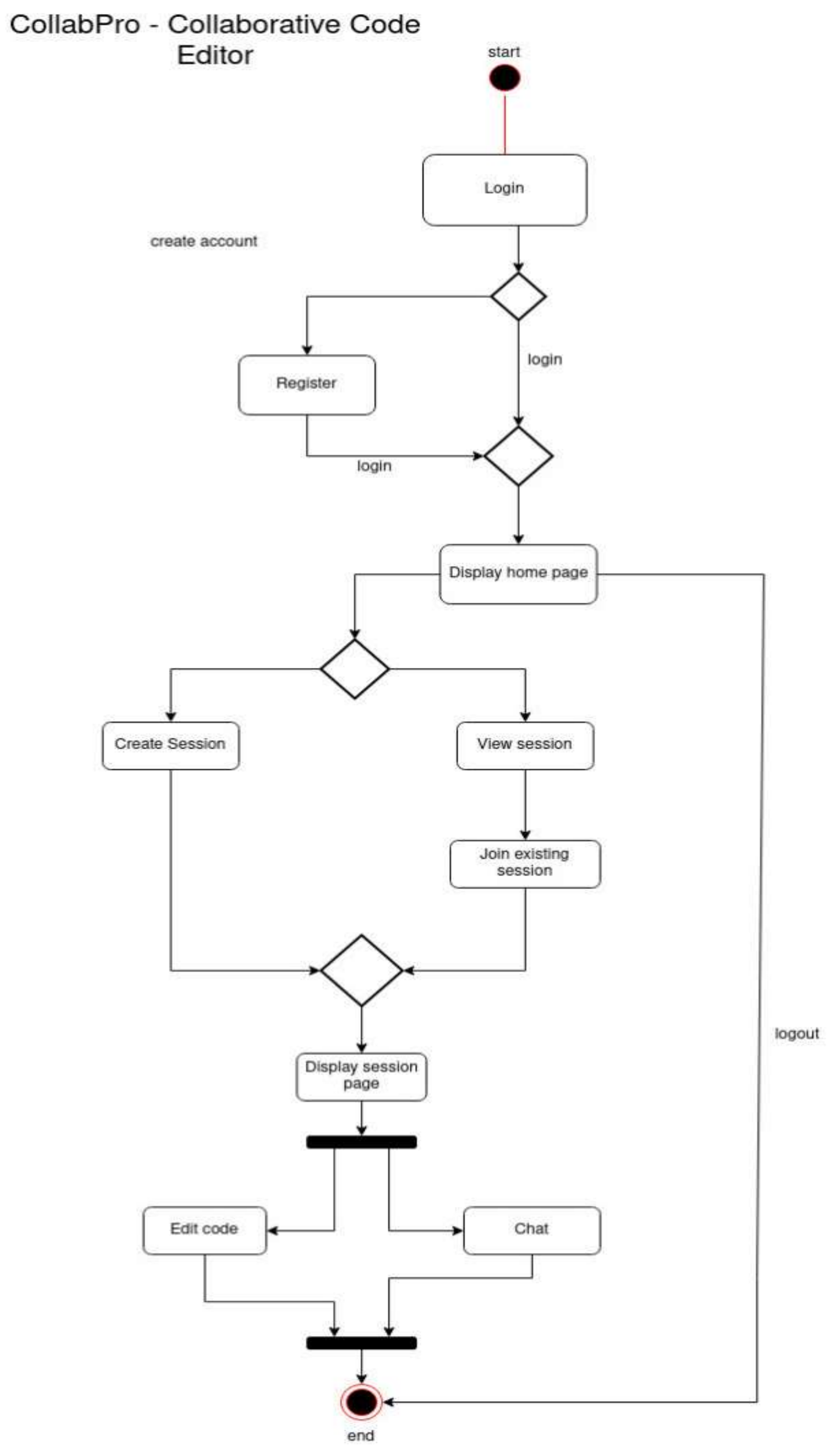


Figure 1.8 Activity Diagram

## 6.6 Component Diagram

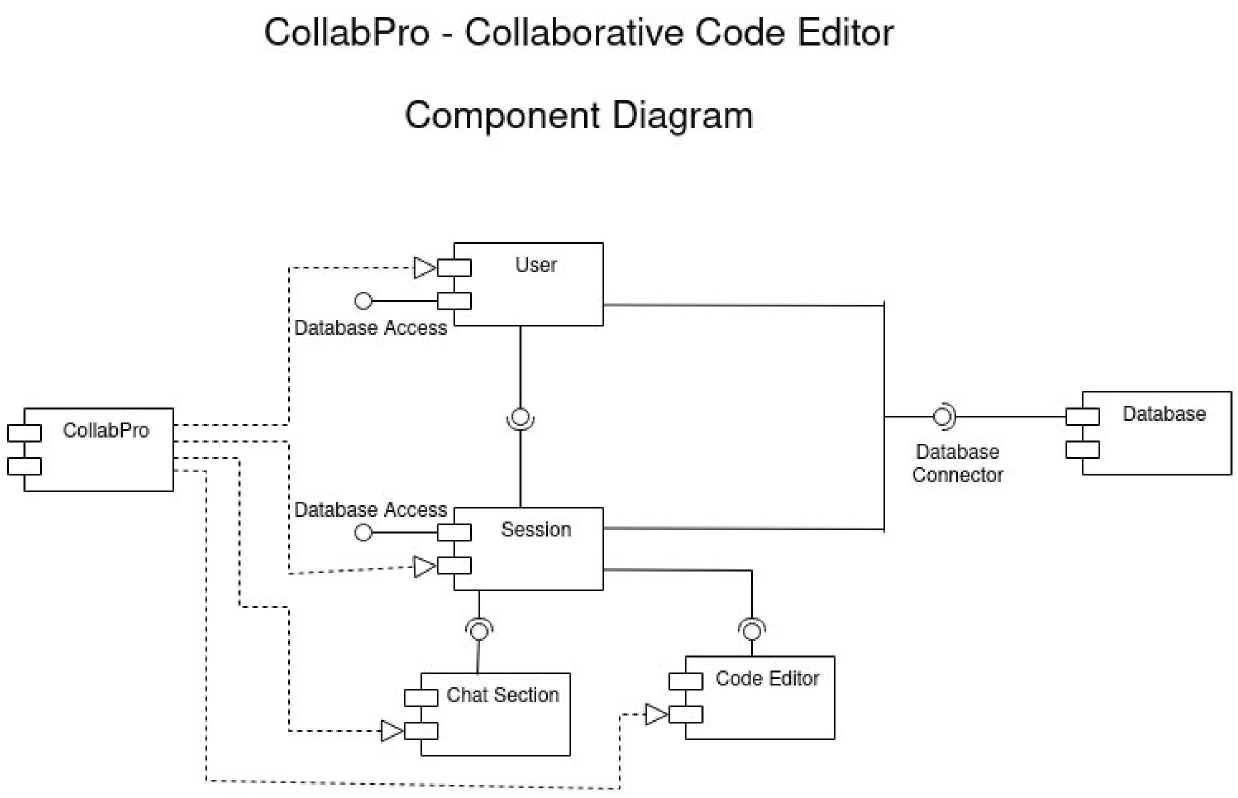


Figure 1.9 Component Diagram

## 7. Snapshots with Result

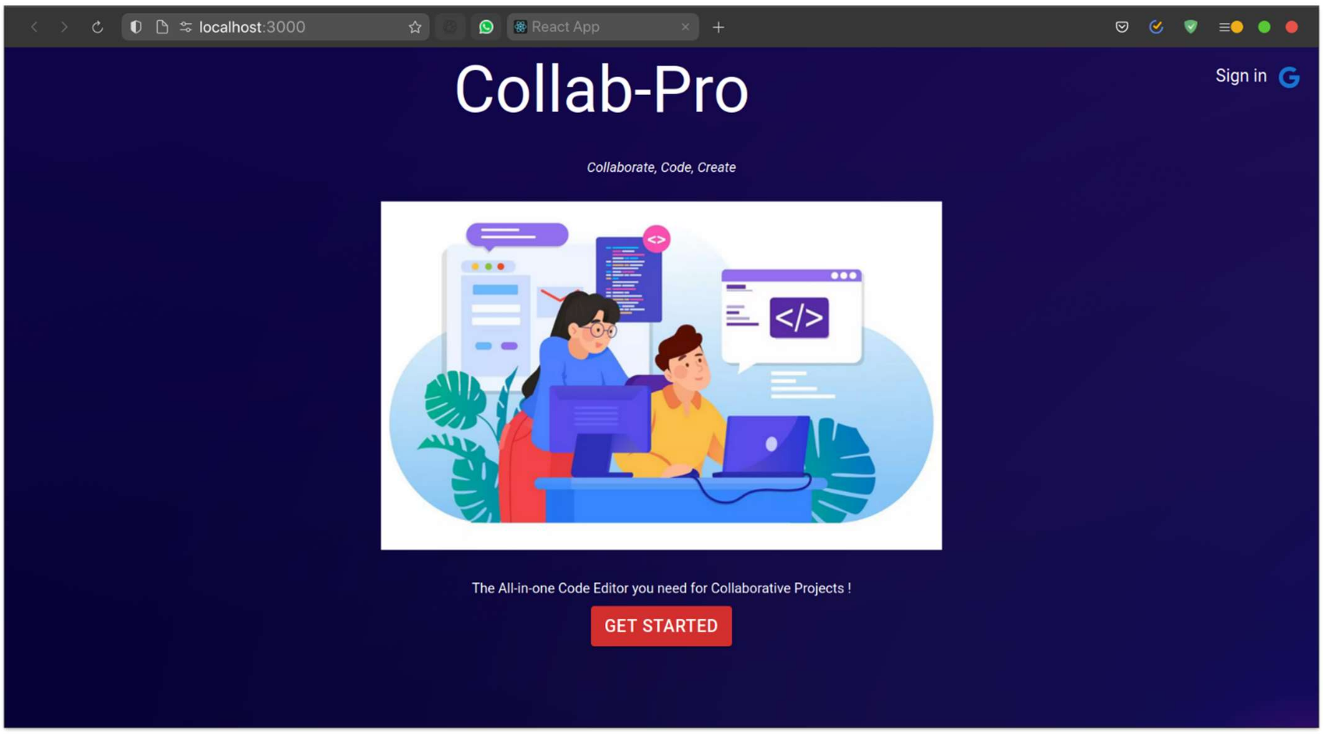
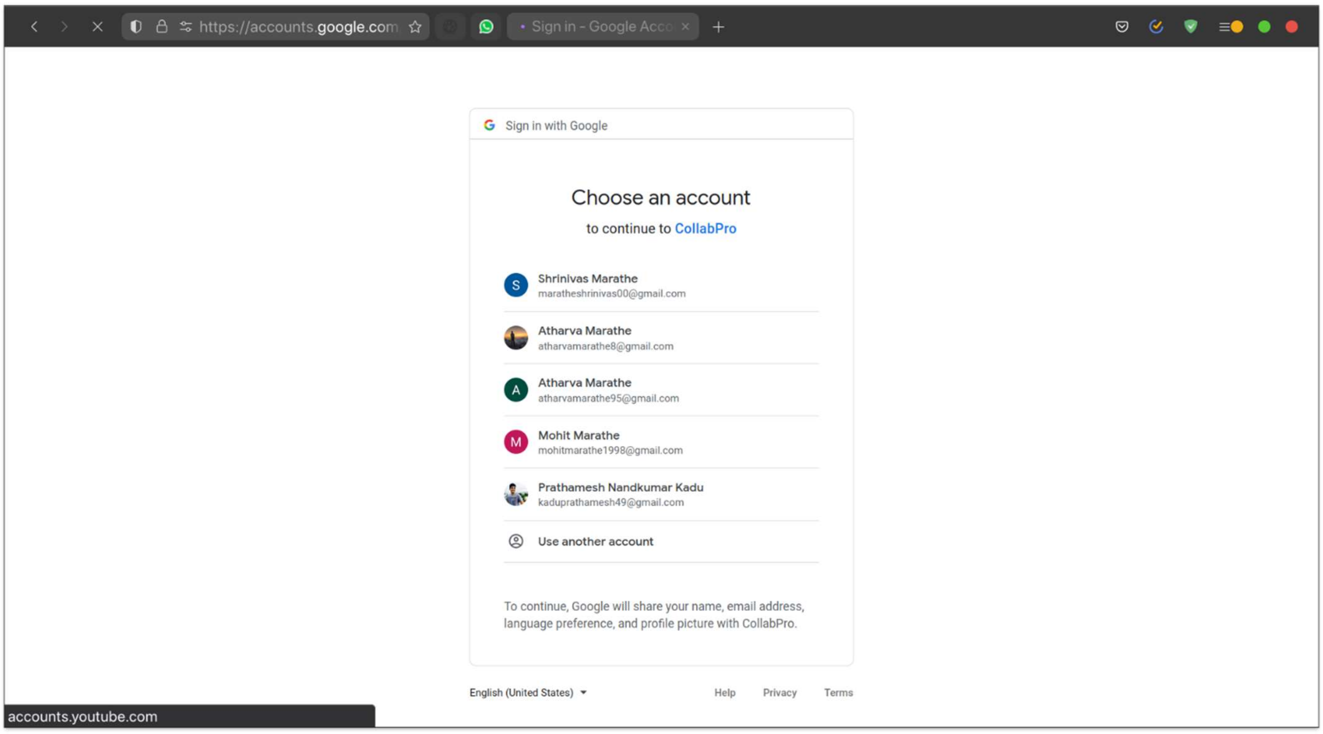

localhost:3000
React App
Collab-Pro
Sign in
Collaborate, Code, Create
The All-in-one Code Editor you need for Collaborative Projects !
GET STARTED

Figure 1.10 Login Page



Sign in with Google

Choose an account

to continue to CollabPro

Shrinivas Marathe
maratheshrinivas00@gmail.com

Atharva Marathe
atharvamarathe8@gmail.com

Atharva Marathe
atharvamarathe95@gmail.com

Mohit Marathe
mohitmarathe1998@gmail.com

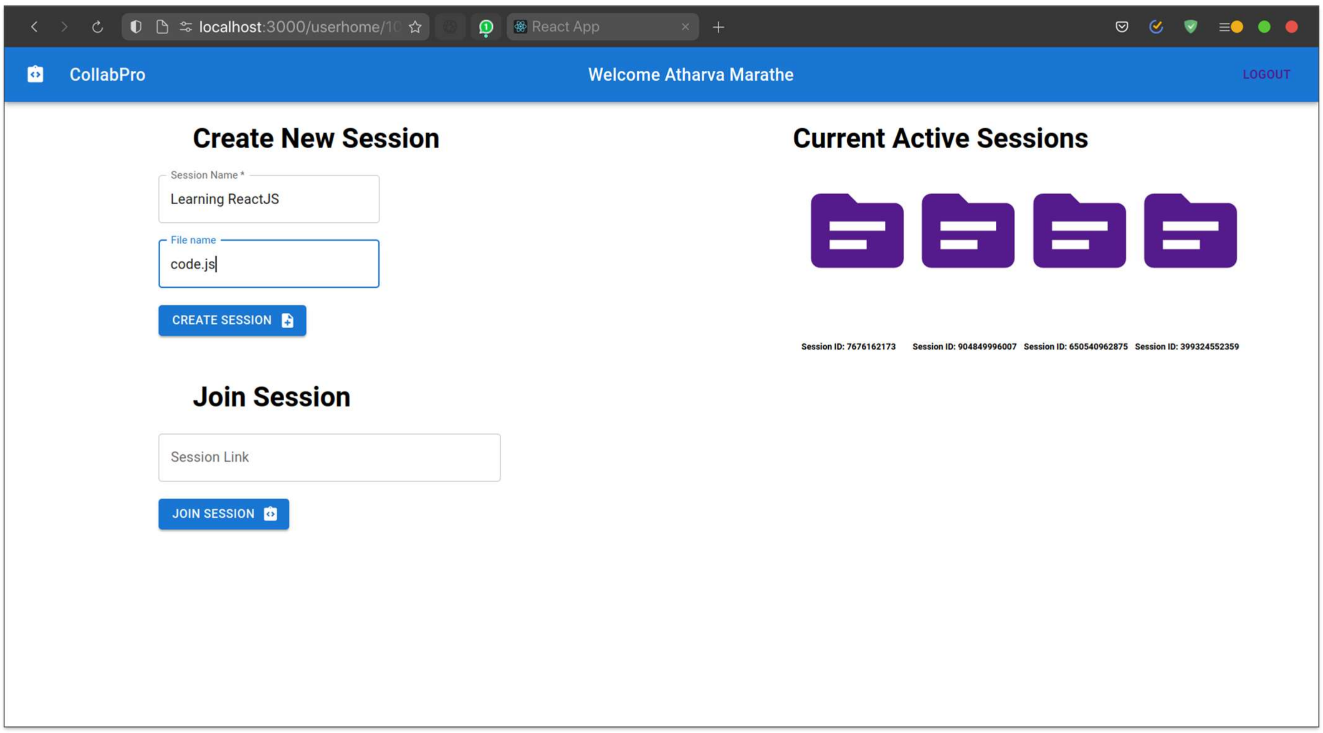
Prathamesh Nandkumar Kadu
kaduprathamesh49@gmail.com

Use another account

To continue, Google will share your name, email address, language preference, and profile picture with CollabPro.

English (United States) Help Privacy Terms

Figure 1.11 OAuth Login Page


localhost:3000/userhome/1
React App

CollabPro
Welcome Atharva Marathe
LOGOUT

Create New Session

Session Name *: Learning ReactJS
File name: code.js

CREATE SESSION

Join Session

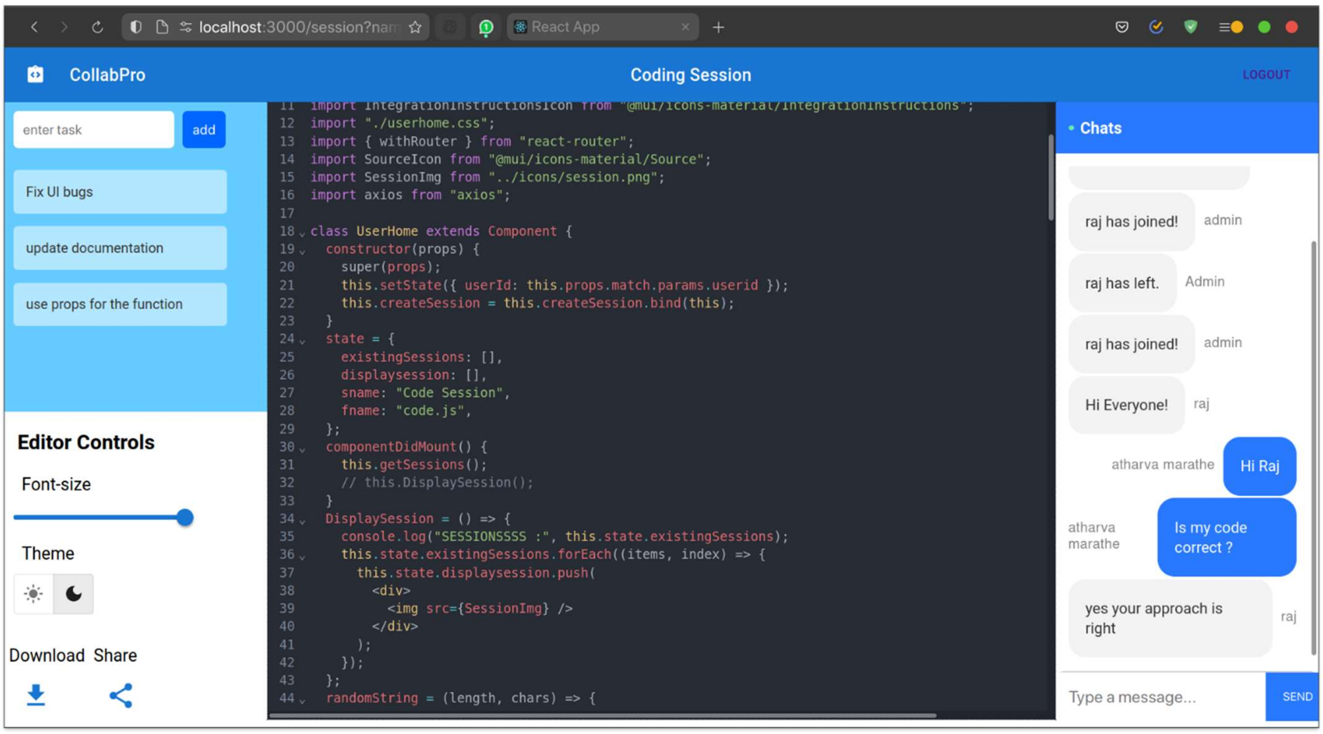
Session Link

JOIN SESSION

Current Active Sessions

Session ID: 7676162173
Session ID: 904849996007
Session ID: 650540962875
Session ID: 399324552359

Figure 1.12 Home Page


localhost:3000/session?nam
React App

CollabPro
Coding Session
LOGOUT

enter task
add

Fix UI bugs
update documentation
use props for the function

Editor Controls
Font-size
Theme
Download Share

```
import IntegrationInstructionsIcon from "@mui/icons-material/IntegrationInstructions";
import "./userhome.css";
import { withRouter } from "react-router";
import SourceIcon from "@mui/icons-material/Source";
import SessionImg from "../icons/session.png";
import axios from "axios";

class UserHome extends Component {
  constructor(props) {
    super(props);
    this.setState({ userId: this.props.match.params.userid });
    this.createSession = this.createSession.bind(this);
  }
  state = {
    existingSessions: [],
    displaysession: [],
    sname: "Code Session",
    fname: "code.js",
  };
  componentDidMount() {
    this.getSessions();
    // this.DisplaySession();
  }
  DisplaySession = () => {
    console.log("SESSIONSSSS :", this.state.existingSessions);
    this.state.existingSessions.forEach((items, index) => {
      this.state.displaysession.push(
        <div>
          <img src={SessionImg} />
        </div>
      );
    });
  };
  randomString = (length, chars) => {
```

Chats

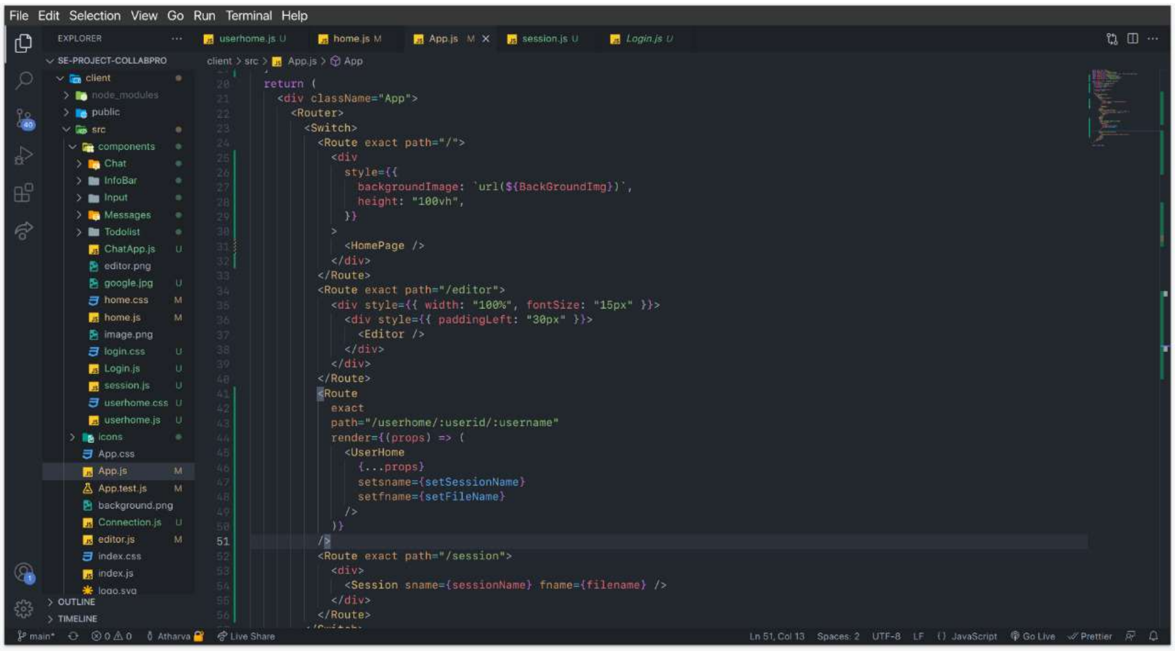
raj has joined! admin
raj has left. Admin
raj has joined! admin
Hi Everyone! raj
atharva marathe Hi Raj
atharva marathe Is my code correct ?
yes your approach is right raj

Type a message...
SEND

Figure 1.12 Session Page

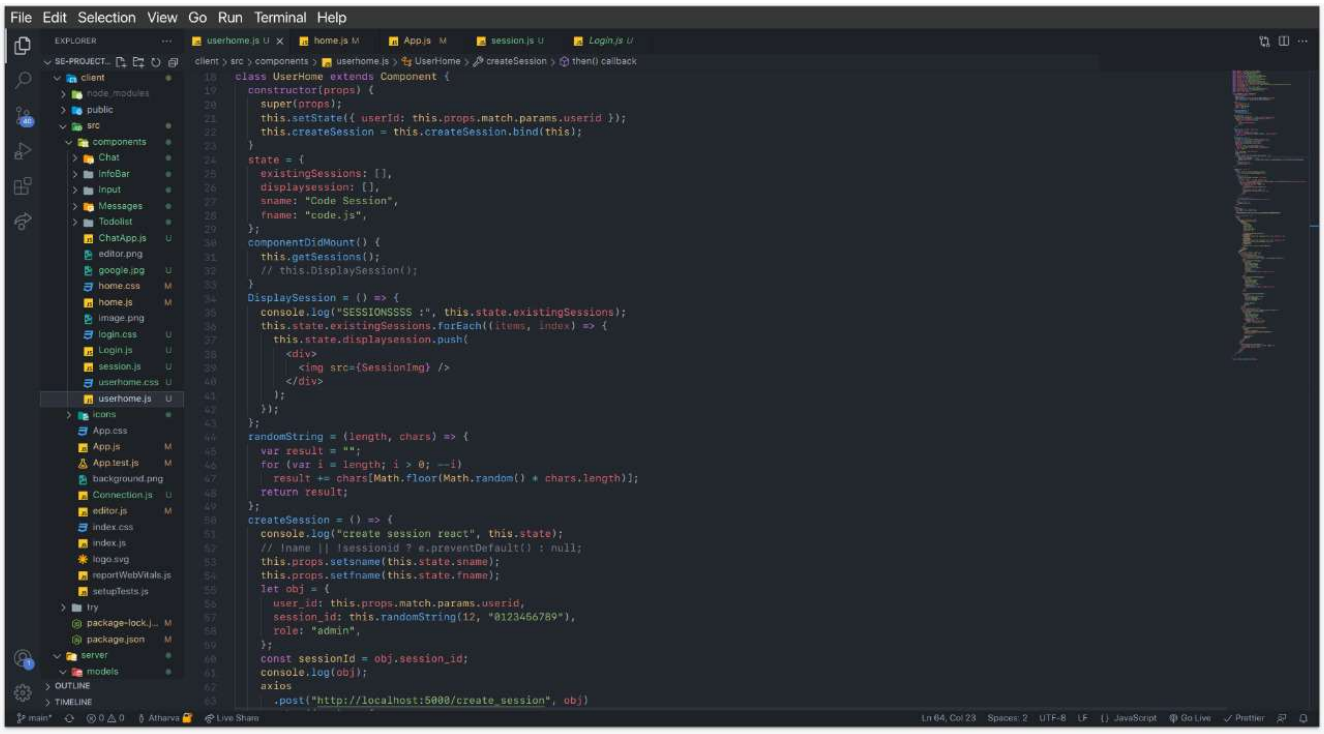
## Coding Snippets:

### 1.Main App component



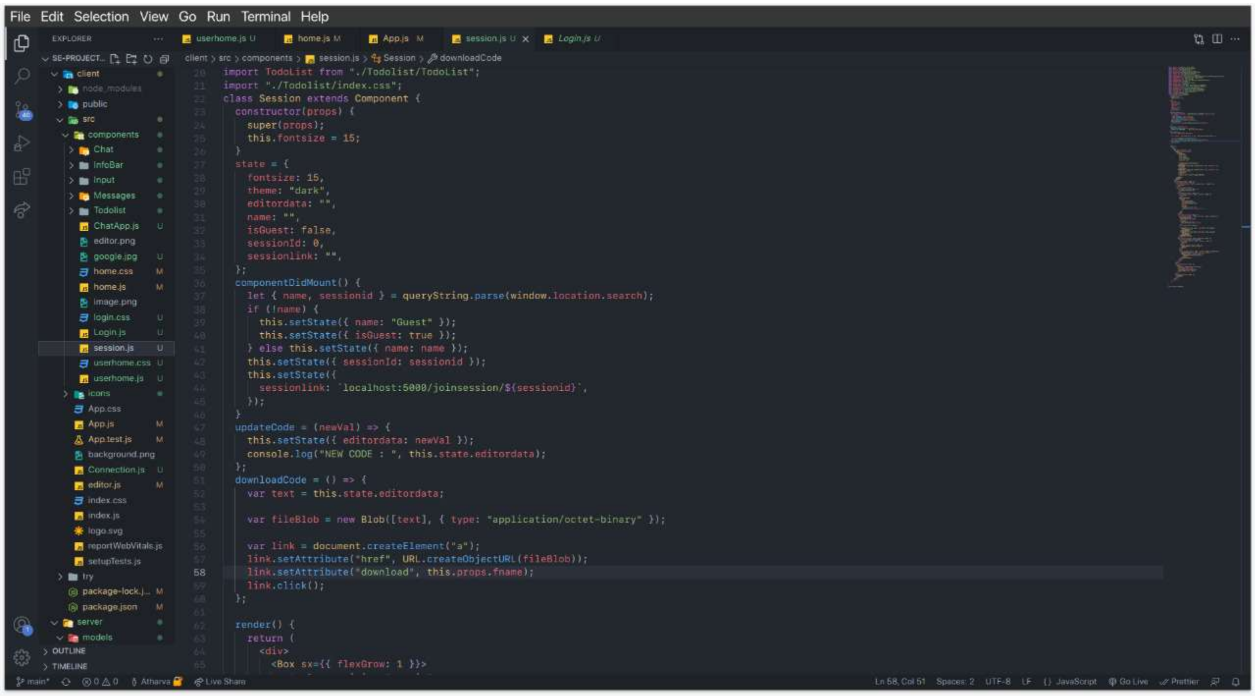
```
    return (
      <div className="App">
        <Router>
          <Switch>
            <Route exact path="/">
              <div
                style={{
                  backgroundImage: `url(${BackGroundImg})`,
                  height: "100vh",
                }}
              >
                <HomePage />
              </div>
            </Route>
            <Route exact path="/editor">
              <div style={{ width: "100%", fontSize: "15px" }}>
                <div style={{ paddingLeft: "30px" }}>
                  <Editor />
                </div>
              </div>
            </Route>
            <Route
              exact
              path="/userhome/:userid/:username"
              render={(props) => (
                <UserHome
                  {...props}
                  setsname={setSessionName}
                  setfname={setFileName}
                />
              )}
            />
            <Route exact path="/session">
              <div>
                <Session sname={sessionName} fname={filename} />
              </div>
            </Route>
```

### 2.Home Page



```
class UserHome extends Component {
  constructor(props) {
    super(props);
    this.setState({ userId: this.props.match.params.userid });
    this.createSession = this.createSession.bind(this);
  }
  state = {
    existingSessions: [],
    displaysession: [],
    sname: "Code Session",
    fname: "code.js",
  };
  componentDidMount() {
    this.getSessions();
    // this.DisplaySession();
  }
  DisplaySession = () => {
    console.log("SESSIONSSSS :", this.state.existingSessions);
    this.state.existingSessions.forEach((items, index) => {
      this.state.displaysession.push(
        <div>
          <img src={SessionImg} />
        </div>
      );
    });
  };
  randomString = (length, chars) => {
    var result = "";
    for (var i = length; i > 0; --i)
      result += chars[Math.floor(Math.random() * chars.length)];
    return result;
  };
  createSession = () => {
    console.log("create session react", this.state);
    // !name || !sessionid ? e.preventDefault() : null;
    this.props.setsname(this.state.sname);
    this.props.setfname(this.state.fname);
    let obj = {
      user_id: this.props.match.params.userid,
      session_id: this.randomString(12, "0123456789"),
      role: "admin",
    };
    const sessionId = obj.session_id;
    console.log(obj);
    axios
      .post("http://localhost:5000/create_session", obj)
```

## 3. Session



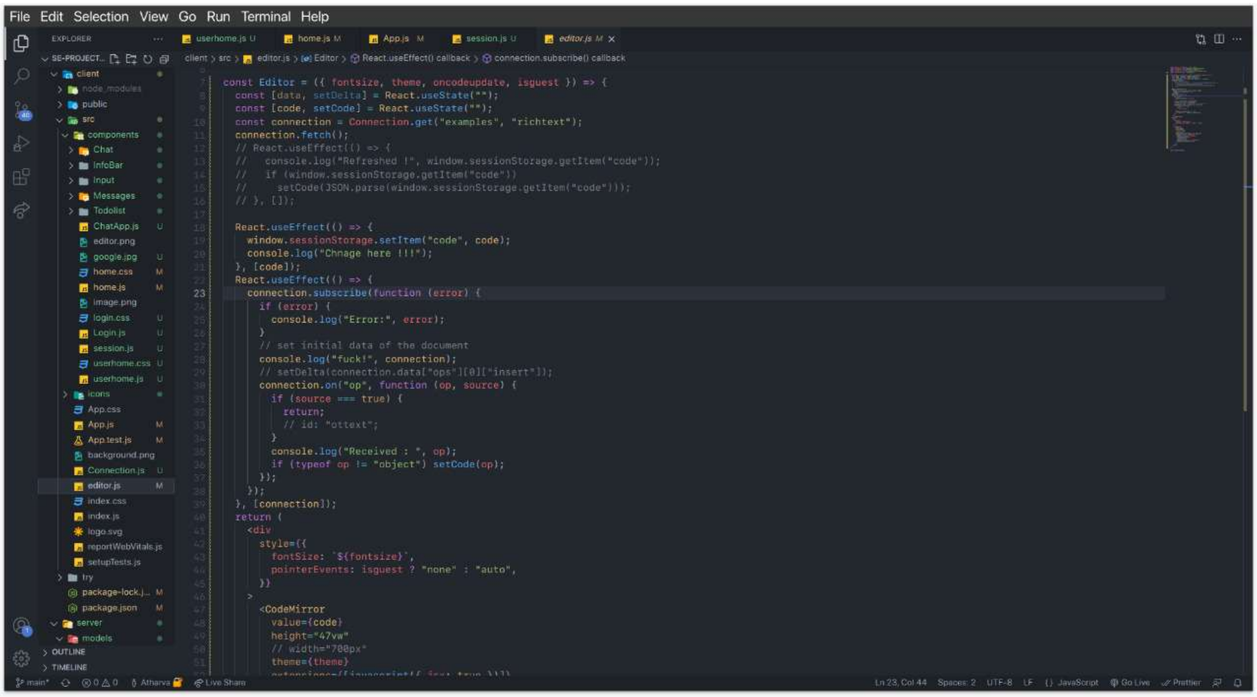
```
import TodoList from "./Todolist/TodoList";
import "./Todolist/index.css";
class Session extends Component {
  constructor(props) {
    super(props);
    this.fontsize = 15;
  }
  state = {
    fontsize: 15,
    theme: "dark",
    editordata: "",
    name: "",
    isGuest: false,
    sessionId: 0,
    sessionlink: "",
  };
  componentDidMount() {
    let { name, sessionid } = queryString.parse(window.location.search);
    if (!name) {
      this.setState({ name: "Guest" });
      this.setState({ isGuest: true });
    } else this.setState({ name: name });
    this.setState({ sessionId: sessionid });
    this.setState({
      sessionlink: `localhost:5000/joinsession/${sessionid}`,
    });
  }
  updateCode = (newVal) => {
    this.setState({ editordata: newVal });
    console.log("NEW CODE : ", this.state.editordata);
  };
  downloadCode = () => {
    var text = this.state.editordata;

    var fileBlob = new Blob([text], { type: "application/octet-binary" });

    var link = document.createElement("a");
    link.setAttribute("href", URL.createObjectURL(fileBlob));
    link.setAttribute("download", this.props.fname);
    link.click();
  };

  render() {
    return (
      <div>
        <Box sx={{ flexGrow: 1 }}>
```

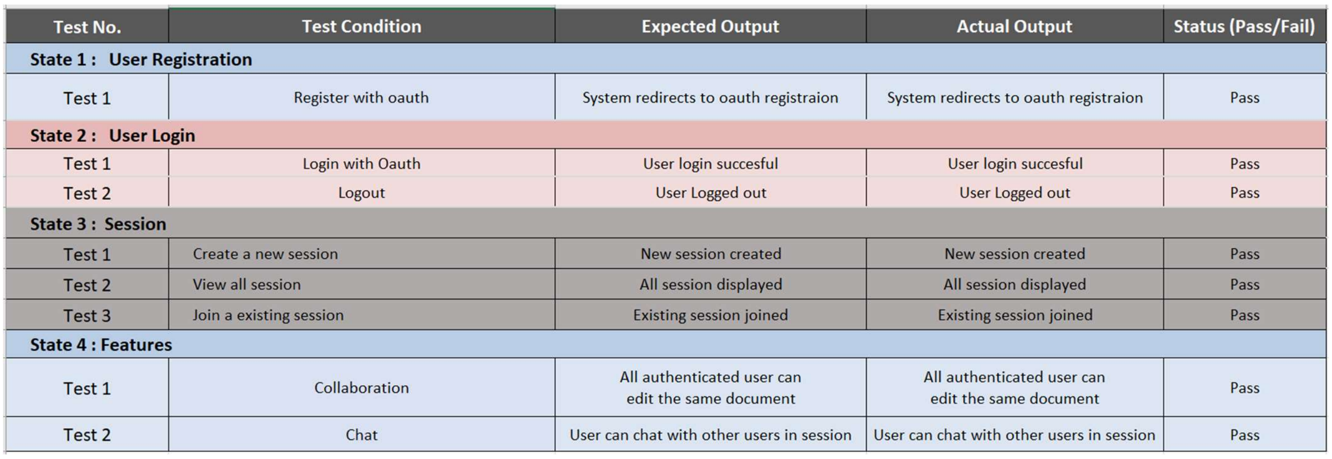
## 4.Editor



```
const Editor = ({ fontsize, theme, oncodeupdate, isguest }) => {
  const [data, setDelta] = React.useState("");
  const [code, setCode] = React.useState("");
  const connection = Connection.get("examples", "richtext");
  connection.fetch();
  // React.useEffect(() => {
  //   console.log("Refreshed !", window.sessionStorage.getItem("code"));
  //   if (window.sessionStorage.getItem("code"))
  //     setCode(JSON.parse(window.sessionStorage.getItem("code")));
  // }, []);

  React.useEffect(() => {
    window.sessionStorage.setItem("code", code);
    console.log("Chnage here !!!");
  }, [code]);
  React.useEffect(() => {
    connection.subscribe(function (error) {
      if (error) {
        console.log("Error:", error);
      }
      // set initial data of the document
      console.log("fuck!", connection);
      // setDelta(connection.data["ops"][0]["insert"]);
      connection.on("op", function (op, source) {
        if (source === true) {
          return;
          // id: "ottext";
        }
        console.log("Received : ", op);
        if (typeof op != "object") setCode(op);
      });
    });
  }, [connection]);
  return (
    <div
      style={{
        fontSize: `${fontsize}`,
        pointerEvents: isguest ? "none" : "auto",
      }}
    >
      <CodeMirror
        value={code}
        height="47vw"
        // width="700px"
        theme={theme}
```

## Testing



| Test No. | Test Condition | Expected Output | Actual Output | Status (Pass/Fail) |
|---|---|---|---|---|
| **State 1 : User Registration** | | | | |
| Test 1 | Register with oauth | System redirects to oauth registraion | System redirects to oauth registraion | Pass |
| **State 2 : User Login** | | | | |
| Test 1 | Login with Oauth | User login succesful | User login succesful | Pass |
| Test 2 | Logout | User Logged out | User Logged out | Pass |
| **State 3 : Session** | | | | |
| Test 1 | Create a new session | New session created | New session created | Pass |
| Test 2 | View all session | All session displayed | All session displayed | Pass |
| Test 3 | Join a existing session | Existing session joined | Existing session joined | Pass |
| **State 4 : Features** | | | | |
| Test 1 | Collaboration | All authenticated user can edit the same document | All authenticated user can edit the same document | Pass |
| Test 2 | Chat | User can chat with other users in session | User can chat with other users in session | Pass |

## 8. Conclusion

- Thus, the CollabPro provides a collaborative coding experience with features like real-time coding, chat-section, syntax highlighting, auto-suggestions etc.
- It provides a simple intuitive experience to the users bundled with useful coding features.
- With the use of OT library, fast and efficient real-time collaboration is assured.
- Google's OAuth provides simple, fast and secure registration and login facility.
- This enables developers to work on collaborative projects on the web with ease.

## 9. Future Scope

- Large projects consist of complex directory structure. So editor supporting this can improve the useability of the product.
- CollabPro provides chat section to communicate within the users. With the facility of video calling, the collaborative experience will become more useful.
- Support for online compiler and a terminal will ease the developers' jobs for testing and running the code.
- Adding Version Control System like git will further simplify the job of the developers to commit, fork to their repositories.

## 10. GitHub Project URL

https://github.com/atharvamarathe/SE-Project-CollabPro

## Appendix A: Glossary

React : Front-end JS library to be used for frontend work
MongoDB : To be used to store databases
Express : Back-end JS library to design web application
Nodejs : Back-end JavaScript runtime environment.