



**CSE471: System Analysis and Design
Project Report**

Project Title: SoulSpeak : An Emotional Support Platform

Group No: 02, CSE471 Lab Section: 04, Fall 2024	
ID	Name
22101090	Saumik Das Turja
21201339	Shaiera Sultana Oishe
21301275	Mohammed Raqin Rahman
22101428	Sibgatullah Tasnim

Submission Date: 7.1.2025

Table of Contents

1. System Request	3
Business need:	3
Business requirements:	3
Business value:	3
Special issues or constraints:	3
2. Functional Requirements	3
3. Technology (Framework, Languages)	3
4. Backend Development	3
5. User Interface Design	3
6. Frontend Development	3
7. User Manual	3
8. Performance and Network Analysis	3
9. Github Repo [Public] Link	4
10. Link of Deployed Project	4
11. Individual Contribution	4
12. References	5

1. System Request

Business need:

In today's rapidly evolving digital landscape, there exists a critical market gap for a comprehensive emotional support platform that effectively addresses the growing mental health challenges faced by individuals worldwide. There is a growing demand for an accessible, empathetic platform that provides emotional support, fosters community interaction, and ensures secure communication for individuals seeking mental well-being. Current solutions often lack personalization, community engagement, and robust administrative controls to ensure safety and quality.

Business Requirements:

1. Develop an emotional support platform with secure user interactions and comprehensive administrative oversight.
2. Enable training and qualification for companions to provide peer-based support.
3. Foster a supportive community through forums and resource sharing.
4. Ensure privacy, security, and feedback integration to maintain trust and relevance.

Business value:

1. Helps users find a safe space for emotional expression and support.
2. Trains companions to ensure quality interactions and advice.
3. Empowers users with access to curated resources for self-help and learning.
4. Creates a moderated environment that reduces the risk of harmful interactions.

Special Issues or Constraints:

1. Maintaining user privacy and data security is critical.
2. Ensuring scalability of the platform as user engagement increases.
3. The platform must comply with data protection regulations (e.g., GDPR).
4. Seamless integration of APIs for dynamic content delivery (e.g., advice API).

2. Functional Requirements

1. **Member Profile Creation:** Users provide details such as gender preference, problem areas, language, and age range.
2. **Profile Editing:** Users can edit their profile information (e.g., name, email, password, preferences).
3. **Profile Picture Update:** Users can change their profile picture.
4. **Companion Qualification Test:** 20 scenario-based MCQ tests to qualify as a companion.
5. **Companion Training Program:** Interactive modules with articles, resources, and text-based modules for training.
6. **Retest Option:** After training completion, users can retake tests to qualify as companions.
7. **Email Verification:** Users can verify their email address after account creation.
8. **Email Change Verification:** Verification is required for updating the email address.
9. **Password Reset via Email:** Users can reset their password securely via email.
10. **Advice API Integration:** Users receive a unique piece of advice from an external API upon login.
11. **Chat Request Initiation:** Users can send chat requests to 3–4 suggested companions based on preferences and availability.
12. **Chat Interaction:** Users can chat with other companions.
13. **Chat Interaction Reporting:** Users can report inappropriate or concerning chats.
14. **Profile Reporting:** Users can report profiles with reasons for evaluation by admins.
15. **Report Feedback:** Users receive updates on actions taken regarding their reports.
16. **Post Creation:** Users can create posts in specific fields.
17. **Post Engagement:** Users can comment on and upvote posts to interact with others.
18. **Post Editing:** Users can edit their own posts.
19. **Post Deletion:** Users can delete their own posts.
20. **Anonymous Posting:** Users can post anonymously to protect their privacy.
21. **Themed Forums:** Forums organized into categories like "Mental Health," "Relationships," etc.
22. **Question Model:** Admin can access the question model and can update questions.
23. **Resource Sharing:** Users can share helpful articles, book recommendations, and mental health resources.
24. **Resource Recommendations Dashboard:** A curated dashboard of recommended resources for users.

25. **User Role Monitoring:** Admins can view and manage user roles and types.
26. **Report Management:** Admins can access and review reported issues.
27. **Feedback Review:** Admins can view user feedback to improve services.
28. **Feedback Analysis:** Admins use MongoDB Atlas to analyze and sort user feedback.
29. **Report Notifications:** Users receive email notifications about report issues.
30. **Report Download in PDF:** Admins can download reports as PDFs.
31. **Profile Suspension:** Admins can ban or suspend users for policy violations.

3. Framework:

MERN Stack (MongoDB, Express.js, React.js, Node.js)

Languages:

Frontend: React.js, JavaScript, HTML, tailwindCSS

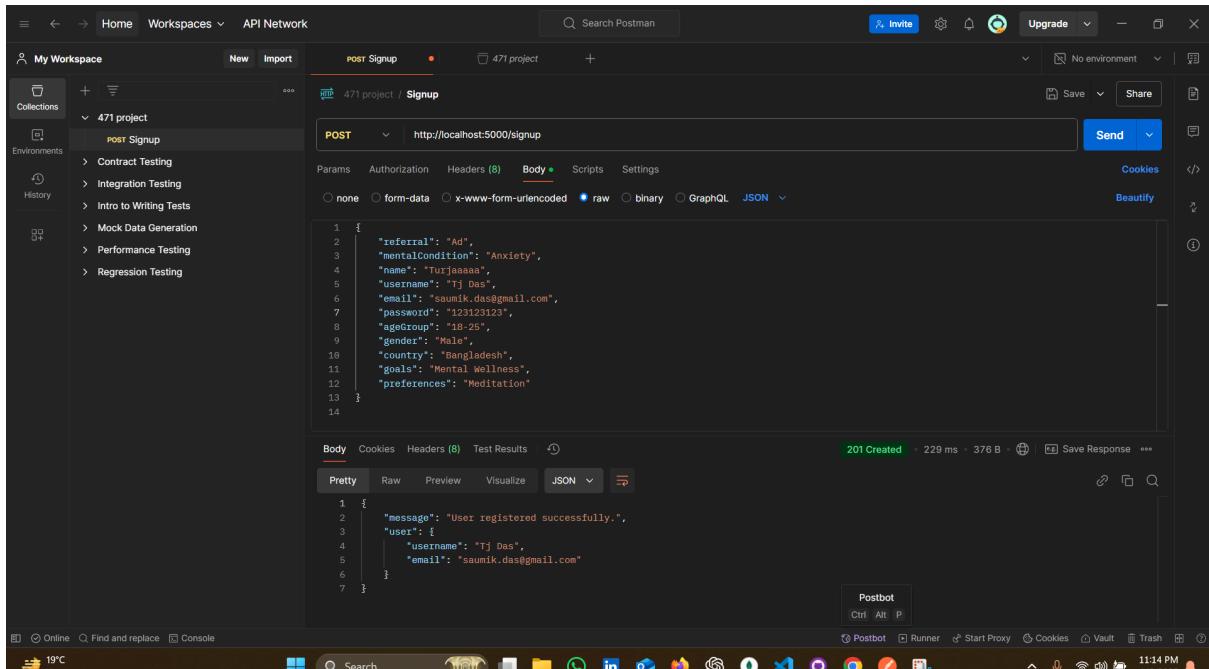
Backend: Node.js, Express.js

Database: MongoDB

APIs: External Advice API for dynamic content

1. Backend Development

Attach at least 5 code snippets of Backend/API development with proper description. Also, attach corresponding screenshots of Postman API testing.

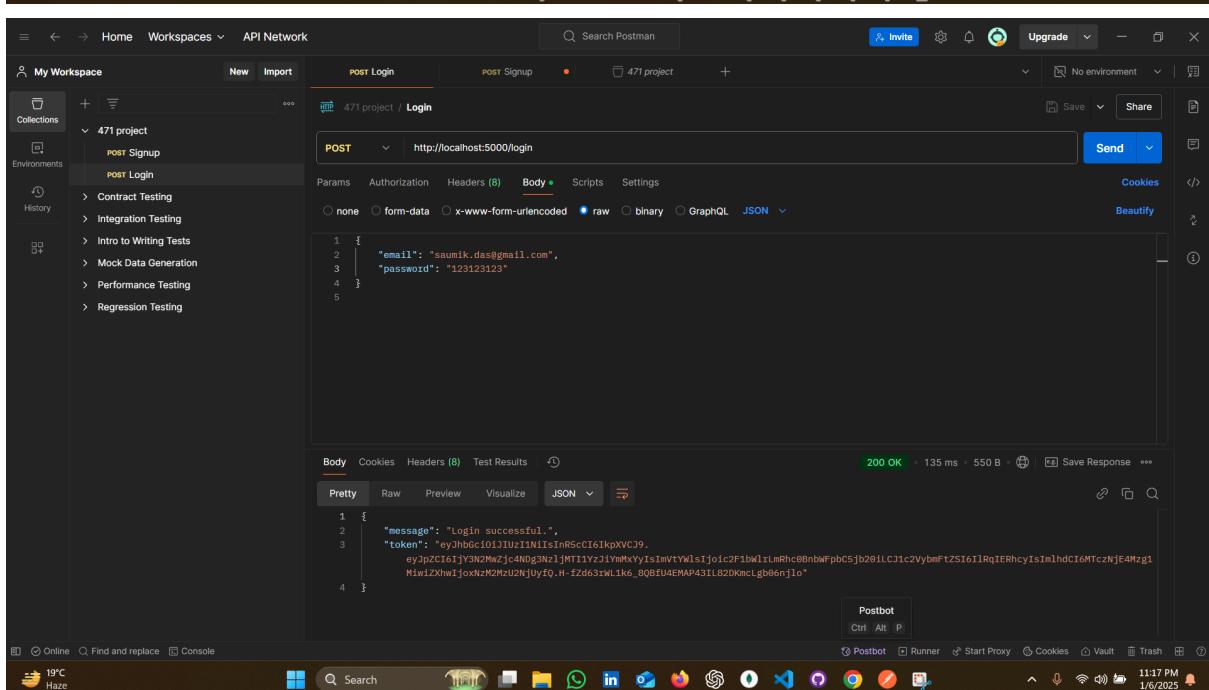


The screenshot shows the Postman interface for a '471 project'. A 'POST Signup' request is being tested against the URL `http://localhost:5000/signup`. The 'Body' tab is selected, showing a raw JSON payload:

```
1 {
2     "referral": "Ad",
3     "mentalcondition": "Anxiety",
4     "name": "Tanjaaaaa",
5     "username": "Tj Das",
6     "email": "saumik.das@gmail.com",
7     "password": "123123123",
8     "ageGroup": "18-25",
9     "gender": "Male",
10    "country": "Bangladesh",
11    "goals": "Mental Wellness",
12    "preferences": "Meditation"
13 }
```

The response status is '201 Created' with a duration of '229 ms' and a size of '376 B'. The response body is:

```
1 {
2     "message": "User registered successfully.",
3     "user": {
4         "username": "Tj Das",
5         "email": "saumik.das@gmail.com"
6     }
7 }
```

The screenshot shows the Postman interface for the same '471 project'. A 'POST Login' request is being tested against the URL `http://localhost:5000/login`. The 'Body' tab is selected, showing a raw JSON payload:

```
1 {
2     "email": "saumik.das@gmail.com",
3     "password": "123123123"
4 }
```

The response status is '200 OK' with a duration of '135 ms' and a size of '550 B'. The response body is:

```
1 {
2     "message": "Login successful.",
3     "token": "eyJhbGciOiJIUzI1NiJ9.RcC16IkpxVCJ9.
eyjpZC16IjY3N2MzJz4NDg3NzljMTI1YzI1YmMxYyIsImVtYmlsIjoiC2F1bwIrlmRhcB8nwFpbC5jb20iLC1c2VybmFtZSI6IlRqIERhcyIsImIhdCI6MTczNjE4Mzg1
MiwiZXhwIjoxNzM2MzU2NjuyfQ.H-fZde3lwLk6_8QBfU4EMAP43IlB2D0KmcLgb8njl0"
4 }
```

The screenshot shows the Postman interface with a project titled "471 project". A POST request is being made to `http://localhost:5000/posts`. The "Body" tab is selected, showing raw JSON data:

```
1 {
2   "content": "This is a new post for checking the API",
3   "category": "General"
4 }
```

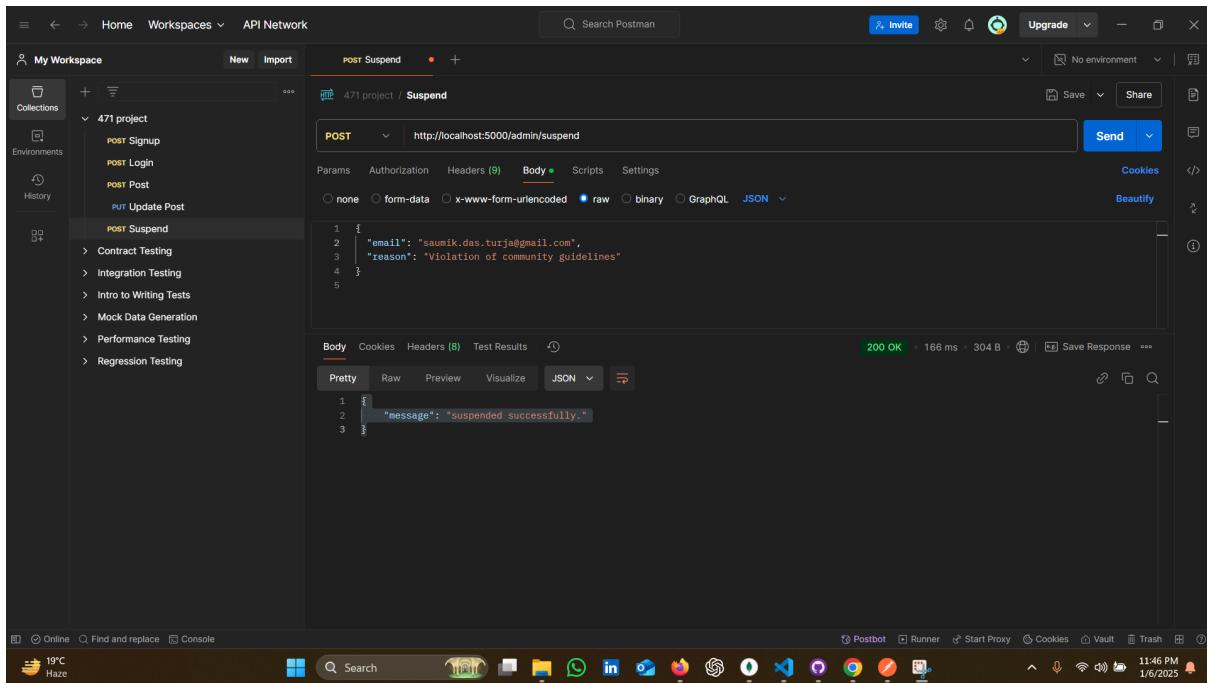
The response status is `201 Created` with a response time of 98 ms and a size of 506 B. The response body is:

```
1 {
2   "content": "This is a new post for checking the API",
3   "category": "General",
4   "imageurl": null,
5   "author": "676599bd1f6669d78a21d5f",
6   "upvotes": [],
7   "_id": "677c12e348779c125c2bbe37",
8   "comments": [],
9   "createdAt": "2025-01-06T17:29:07.700Z",
10  "__v": 0
11 }
```

The screenshot shows the Postman interface with the same project. A PUT request is being made to `http://localhost:5000/posts/677c12e348779c125c2bbe37`. The "Body" tab is selected, showing raw JSON data:

```
1 {
2   "content": "Updated content for the API checking post",
3   "category": "General",
4   "imageurl": null,
5   "author": "676599bd1f6669d78a21d5f",
6   "upvotes": [
7     "6776b31359b6bed292746dd"
8   ],
9   "comments": [
10    {
11      "user": "6776b31359b6bed292746dd",
12      "content": "ok",
13      "_id": "677c135abdbaeecd3ea9d8e",
14      "createdAt": "2025-01-06T17:31:06.000Z"
15    }
16 }
```

The response status is `200 OK` with a response time of 176 ms and a size of 652 B.



index.js

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows the project structure under the "SOUKSPAK" folder. The "index.js" file is open in the editor.
- Editor Area:** Displays the content of the "index.js" file, which is a Node.js application starting with Express and socket.io configurations.
- Top Bar:** Includes standard file operations like File, Edit, Selection, View, Go, Run, Terminal, Help, and a back/forward navigation bar.
- Search Bar:** Contains the text "SoulSpeak".
- Bottom Status Bar:** Shows the file path "index.js", line numbers (1-47), and other status indicators.

```
File Edit Selection View Go Run Terminal Help < > SoulSpeak
OPEN EDITORS 2 opened
SOUKSPAK
  > INDEX.js
    backend > index.js
      1 import express from 'express';
      2 import cors from 'cors';
      3 import dotenv from 'dotenv';
      4 import { connectDB } from './config/db.js';
      5 import adminRouter from './routes/adminRoute.js';
      6 import chatRouter from './routes/chatRoute.js';
      7 import http from 'http';
      8 import { Server } from 'socket.io';
      9 import chatSocket from './socket/chatSocket.js';
     10
     11 dotenv.config();
     12
     13 dotenv.config();
     14
     15 const app = express();
     16 const PORT = process.env.PORT || 5981;
     17
     18 const server = createServer(app);
     19 const io = new Server(server, {
     20   cors: {
     21     origin: "*",
     22     methods: ["GET", "POST"]
     23   }
     24 });
     25
     26 app.set("io", io);
     27
     28 // Middlewares
     29 app.use(cors());
     30 app.use(express.json());
     31
     32 app.use("", root)
     33 app.use("/admin", adminRouter)
     34 app.use("/chat", chatRouter);
     35
     36 app.use("/uploads", express.static('uploads'));
     37
     38
     39
     40   chatSocket(io);
     41
     42 connectDB();
     43
     44 // Start Server
     45 server.listen(PORT, () => console.log(`Server running on http://localhost:\${PORT}`));
     46
     47 |
```

db.js

A screenshot of a code editor interface, likely VS Code, showing a file named `db.js`. The code is a Node.js script for connecting to a MongoDB database using Mongoose. The code is as follows:

```
1 import mongoose from "mongoose"
2
3
4 export const connectDB = async () =>{
5   try {
6     const conn = await mongoose.connect(process.env.MONGO_URI)
7     console.log(`MongoDB connected: ${conn.connection.host}`)
8   } catch (error) {
9     console.log(`Error: ${error.message}`)
10    process.exit(1)
11  }
12}
13
14
```

middlewares.js



```
File Edit Selection View Go Run Terminal Help < > SouSpeak
middlewares.js U
backend > config > middlewares.js > ...
1 import jwt from "jsonwebtoken";
2
3 const JWT_SECRET = "soul";
4
5 export const authenticateToken = (req, res, next) => {
6   const token = req.headers.authorization && req.headers.authorization.split(" ")[1]; // Extract token from "Bearer <token>"
7
8   if (!token) {
9     return res.status(401).json({ message: "Access denied. No token provided." });
10  }
11  try {
12    const decoded = jwt.verify(token, JWT_SECRET);
13    req.user = decoded;
14    next();
15  } catch (error) {
16    return res.status(403).json({ message: "Invalid or expired token." });
17  }
18};
19
```

adminController.js

```

File Edit Selection View Go Run Terminal Help ← → ⌘ SoulSpeak
adminController.js 0 X
backed > controllers > adminController.js > updateReportStatus
1 import User from "../models/userModel.js";
2 import Question from "../models/questionModel.js";
3 import nodemailer from "nodemailer";
4 import nodemailer from "nodemailer";
5
6 const transporter = nodemailer.createTransport({
7   service: "gmail",
8   auth: {
9     user: process.env.EMAIL,
10    pass: process.env.EMAIL_PASS
11  }
12 });
13
14 export async function get_all_questions(request, response) {
15   try {
16     const questions = await Question.find();
17     return response.status(200).json(questions);
18   } catch (error) {
19     console.error("Error fetching questions:", error);
20     return response.status(500).json({ message: "Internal server error." });
21   }
22 }
23
24 export async function add_companion_test_question(request, response) {
25   console.log("Received request to add a question");
26   try {
27     const body = await request.body;
28     const { question, option1, option2, option3, option4, correct, resource } = body;
29     if (!question || !option1 || !option2 || !option3 || !option4 || !correct || !resource) {
30       return response.status(400).json({ message: "All fields are required." });
31     }
32     const user = await User.findById(request.user.id);
33     if (!user) {
34       return response.status(401).json({ message: "User not found." });
35     }
36     if (!user.email === "admin@gmail.com") {
37       return response.status(403).json({ message: "You are not authorized to add questions." });
38     }
39     const newQuestion = new Question({
40       question,
41       option1,
42       option2,
43       option3,
44       option4,
45       correct,
46       resource,
47     });
48     await newQuestion.save();
49     console.log("Question added successfully:", newQuestion);
50     return response.status(200).json({ message: "Question added successfully." });
51   } catch (error) {
52     console.error("Error during signout:", error);
53     return response.status(500).json({ message: "Internal server error." });
54   }
55 }

```

All.js

```

File Edit Selection View Go Run Terminal Help ← → ⌘ SoulSpeak
All.js 0 X
backed > controllers > All.js > sendPasswordResetEmail
1 import bcrypt from "bcrypt";
2 import User from "../models/userModel.js";
3 import Jwt from "jsonwebtoken";
4 import Post from "../models/postModel.js";
5 import transporter from "nodemailer/lib/transport";
6 import Question from "../models/questionModel.js";
7 import ProfilePicture from "../models/profilePicture.js";
8 import nodemailer from "nodemailer";
9 import multer from "multer";
10 import path from "path";
11
12
13
14
15 const storage = multer.diskStorage({
16   destination: (req, file, cb) => {
17     cb(null, "uploads/");
18   },
19   filename: (req, file, cb) => {
20     cb(null, Date.now() + path.basename(file.originalname));
21   }
22 });
23
24 const upload = multer({ storage,
25   fileFilter: (req, file, cb) => {
26     const allowedTypes = ["image/jpeg", "image/png", "image/gif"];
27     if (!allowedTypes.includes(file.mimetype)) {
28       return cb(new Error("Only images are allowed"), false);
29     }
30     cb(null, true);
31   })
32 });
33
34
35 export async function verifyEmail(req, res) {
36   try {
37     const token = jwt.sign({ id: req.user.id }, process.env.JWT_SECRET, { expiresIn: "1d" });
38     const url = `http://localhost:5000/confirm-email?token=${token}`;
39
40     const transporter = nodemailer.createTransport({
41       service: "gmail",
42       auth: {
43         user: process.env.EMAIL,
44         pass: process.env.EMAIL_PASS,
45       }
46     });
47     const user = await User.findById(req.user.id);
48     console.log("About to transport");
49     await transporter.sendMail({
50       to: user.email,
51       subject: "Verify Your Email",
52       html: `

Click here to verify your email.

`,
53     });
54     res.status(200).json({ message: "Verification email sent." });
55   } catch (error) {
56     res.status(500).json({ message: "Internal server error." });
57   }
58 }

```

chatController.js

```

chatController.js | X
File Edit Selection View Go Run Terminal Help ← → ⌂ SoulSpeak
1 backed > controllers > ChatController.js ...
2
3 import Chat from '../models/Chat.js';
4
5 export async function createChat(req, res) {
6   try {
7     const { participantId } = req.body;
8     const userId = req.user.id;
9
10    // Check if chat already exists
11    const existingChat = await Chat.findOne({
12      $or: [
13        { participantId: { $all: [userId, participantId] } },
14        { participantId: participantId, user: userId }
15      ],
16      'user.isOnline': true
17    });
18
19    if (existingChat) {
20      return res.status(200).json(existingChat);
21    }
22
23    // Create new chat
24    const newChat = await Chat.create({
25      participants: [userId, participantId],
26      messages: []
27    });
28
29    // Populate the chat with participant details
30    const populatedChat = await Chat.findById(newChat._id)
31      .populate('participants', 'username email isOnline isCompanion');
32
33    // Get Socket ID instance
34    const io = req.app.get('io');
35
36    // Emit to both participants' personal rooms
37    [userId, participantId].forEach(id => {
38      io.to(id.toString()).emit('newChat', populatedChat);
39    });
40
41    res.status(201).json(populatedChat);
42  } catch (error) {
43    console.error('Create chat error:', error);
44    res.status(500).json({ error: error.message });
45  }
46
47
48  export async function sendMessage(req, res) {
49    console.log('sendMessage function called');
50    try {
51      const { chatId, content } = req.body;
52      const userId = req.user.id;
53
54      const chat = await Chat.findById(chatId);
55
56      const message = {
57        sender: userId,
58        timestamp: Date.now(),
59        content,
60        ref: null
61      };
62
63      const messageSchema = new mongoose.Schema({
64        sender: {
65          type: mongoose.Schema.Types.ObjectId,
66          ref: 'User',
67          required: true
68        },
69        content: {
70          type: String,
71          required: true
72        },
73        timestamp: {
74          type: Date,
75          default: Date.now()
76        }
77      });
78
79      const messages = messageSchema;
80      const lastMessage = {
81        type: Date,
82        default: Date.now()
83      };
84
85      const chatSchema = new mongoose.Schema({
86        participants: {
87          type: mongoose.Schema.Types.ObjectId,
88          ref: 'User'
89        },
90        messages: [messages],
91        lastMessage: lastMessage
92      });
93
94      const Chat = mongoose.model('Chat', chatSchema);
95
96      const updatedChat = await Chat.findByIdAndUpdate(chatId, {
97        $push: { messages: message },
98        $set: { lastMessage }
99      });
100
101      res.json(updatedChat);
102    } catch (error) {
103      console.error('Send message error:', error);
104      res.status(500).json({ error: error.message });
105    }
106  }
107}

```

Chat.js

```

Chat.js | X
File Edit Selection View Go Run Terminal Help ← → ⌂ SoulSpeak
1 Chat.js | X
2
3 backed > models > Chat.js | W/default
4
5 import mongoose from 'mongoose';
6
7 const messageSchema = new mongoose.Schema({
8   sender: {
9     type: mongoose.Schema.Types.ObjectId,
10    ref: 'User',
11    required: true
12  },
13  content: {
14    type: String,
15    required: true
16  },
17  timestamp: {
18    type: Date,
19    default: Date.now()
20  }
21 });
22
23 const chatSchema = new mongoose.Schema({
24   participants: {
25     type: mongoose.Schema.Types.ObjectId,
26     ref: 'User'
27   },
28   messages: [messageSchema],
29   lastMessage: {
30     type: Date,
31     default: Date.now()
32   }
33 });
34
35 const Chat = mongoose.model('Chat', chatSchema);
36
37 export default Chat;

```

Post.js



```
File Edit Selection View Go Run Terminal Help < > SoulSpeak
```

```
backend /models > # Post.js ->
  1 import mongoose from 'mongoose';
  2
  3 const postSchema = new mongoose.Schema({
  4   content: {
  5     type: String,
  6     required: true,
  7   },
  8   category: {
  9     type: String,
 10     required: true,
 11     enum: ['Welcome Center', 'Mindful Moments', 'Wellness Hub', 'Support Circle', 'General'],
 12     default: 'General'
 13   },
 14   imageUrl: {
 15     type: String,
 16   },
 17   author: {
 18     type: mongoose.Schema.Types.ObjectId,
 19     ref: 'User',
 20     required: true,
 21   },
 22   upvotes: [
 23     type: mongoose.Schema.Types.ObjectId,
 24     ref: 'User'
 25   ],
 26   comments: [
 27     type: mongoose.Schema.Types.ObjectId,
 28     ref: 'User'
 29   ],
 30   content: String,
 31   createdAt: {
 32     type: Date,
 33     default: Date.now
 34   },
 35 }, {
 36   toJSON: {
 37     virtuals: true,
 38     transform: {
 39       _id: null,
 40     }
 41   }
 42 };
 43
 44 const Post = mongoose.model('Post', postSchema);
 45 export default Post;
```

profilePicture.js

```
File Edit Selection View Go Run Terminal Help < > ⚡ SoulSpeak ⚡ Backend > models > profilePicture.js > ...  
js profilePicture.js U  
backend > models > profilePicture.js > ...  
1 import mongoose from 'mongoose';  
2  
3 const profile_pic_Schema = new mongoose.Schema({  
4   imageUrl: {  
5     type: String,  
6   },  
7   user: {  
8     type: mongoose.Schema.Types.ObjectId,  
9     ref: 'User',  
10    required: true,  
11  },  
12});  
13  
14 const ProfilePicture = mongoose.model('ProfilePicture', profile_pic_Schema);  
15  
16 export default ProfilePicture;
```

questionModel.js

```
questionModel.js
backend > models > questionModel.js > default
1 import mongoose from "mongoose";
2
3 const questionSchema = new mongoose.Schema({
4   question: {
5     type: String,
6     required: true,
7   },
8   option1: [
9     type: [String],
10    required: true,
11  ],
12   option2: [
13     type: [String],
14    required: true,
15  ],
16   option3: [
17     type: [String],
18    required: true,
19  ],
20   option4: [
21     type: [String],
22    required: true,
23  ],
24   resources: {
25     type: String,
26     required: true,
27   },
28   correct: {
29     type: Number,
30     required: true,
31   },
32 },
33 });
34
35 const Question = mongoose.model("Question", questionSchema);
36
37 export default Question;
```

Report.js

```
Report.js
backend > models > Report.js > default
1 import mongoose from "mongoose";
2
3 const reportSchema = new mongoose.Schema({
4   userEmail: {
5     type: String,
6     required: true
7   },
8   timestamp: {
9     type: Date,
10    default: Date.now
11  },
12   type: {
13     type: String,
14     enum: ['review', 'profile-report', 'chat-report'],
15     required: true
16   },
17   details: {
18     type: String,
19     required: true
20   },
21   status: {
22     type: String,
23     enum: ['pending', 'resolved', 'rejected'],
24     default: 'pending'
25   }
26 });
27
28 const Report = mongoose.model('Report', reportSchema);
29
30 export default Report;
```

userModel.js

```


1  import mongoose from "mongoose";
2
3  const userSchema = mongoose.Schema({
4      name: {
5          type: String,
6          required: [true, "Please provide your Name"]
7      },
8      username: {
9          type: String,
10         required: [true, "Please provide your Username"],
11         unique: true
12     },
13     email: {
14         type: String,
15         required: [true, "Please provide your Email Address"],
16         unique: true
17     },
18     password: {
19         type: String,
20         required: [true, "Please provide your Password"]
21     },
22     referral: {
23         type: String,
24         required: [true, "Please select how you found us"]
25     },
26     mentalCondition: {
27         type: String,
28         required: [true, "Please select your condition"]
29     },
30     ageGroup: {
31         type: String,
32         required: [true, "Please answer question 1"]
33     },
34     country: {
35         type: String,
36         required: [true, "Please answer question 2"]
37     },
38     goals: {
39         type: String,
40         required: [true, "Please answer question 3"]
41     },
42     preferences: {
43         type: String,
44         required: [true, "Please answer preferences"]
45     }
46 }
47
48 module.exports = mongoose.model("User", userSchema);


```

adminRoute.js

```


1  import express from "express";
2  import { activate, add_companion_test_question, delete_question, get_all_questions, update_question, get_all_users, suspend, isSuspended, updateReportStatus } from "../controllers/adminController.js";
3  import { authenticateToken } from "../config/middlewares.js";
4
5  const adminRouter = express.Router();
6
7  adminRouter.post("/questions", authenticateToken, add_companion_test_question);
8  adminRouter.get("/questions", get_all_questions);
9  adminRouter.delete("/questions/:id", authenticateToken, delete_question);
10 adminRouter.put("/questions/:id", authenticateToken, update_question);
11 adminRouter.get("/users", authenticateToken, get_all_users);
12
13 adminRouter.post("/suspend", authenticateToken, suspend);
14 adminRouter.post("/activate", authenticateToken, activate);
15 adminRouter.get("/suspension/:email", authenticateToken, isSuspended);
16
17 adminRouter.patch("/reports/:id/status", authenticateToken, updateReportStatus);
18
19 export default adminRouter;
20


```

chatRoute.js

SoulSpeak

```

File Edit Selection View Go Run ... ↶ ↷ ⌂ SoulSpeak
JS adminRoute.js U JS chatRoute.js U
backend > routes > JS chatRoute.js ...
1 import express from "express";
2 import { getChats, sendMessage, createChat } from "../controllers/chatController.js";
3 import { authenticateToken } from "../config/middlewares.js";
4
5 const chatRouter = express.Router();
6
7 chatRouter.get("/", authenticateToken, getChats);
8 chatRouter.post("/send", authenticateToken, sendMessage);
9 chatRouter.post("/create", authenticateToken, createChat);
10
11 export default chatRouter;
12

```

Ln 12, Col 1 Spaces:4 UTF-8 LF {} JavaScript ⌂ ⌂

route.js

SoulSpeak

```

File Edit Selection View Go Run Terminal Help ⌂ ⌂ SoulSpeak
JS adminRoute.js U JS chatRoute.js U JS route.js U
backend > routes > JS routes.js ...
1 import express from "express";
2 import { signup, login, profile, updateProfile, getPosts, createPost, updatePost, deletePost, upvotePost, addComment, getComments, deleteComment, getReports, createReport, companion, companions, verifyEmail, authenticateToken } from "../config/middlewares.js"
3
4 const router = express.Router();
5
6 router.post('/signup', signup)
7 router.post('/login', login)
8 router.get('/profile', authenticateToken, profile)
9 router.put('/profile', authenticateToken, updateProfile)
10
11 router.get('/posts', getPosts)
12 router.post('/posts', authenticateToken, createPost);
13 router.put('/posts/:id', authenticateToken, updatePost);
14 router.delete('/posts/:id', authenticateToken, deletePost);
15 router.post('/posts/:id/upvote', authenticateToken, upvotePost);
16 router.post('/posts/:id/comments', authenticateToken, addComment);
17 router.get('/posts/:id/comments', getComments);
18 router.delete('/posts/:postid/comments/:commentId', authenticateToken, deleteComment);
19
20
21 router.get('/reports', authenticateToken, getReports);
22 router.post('/reports', authenticateToken, createReport);
23
24 router.post('/companion', authenticateToken, companion)
25 router.get('/companions',companions)
26
27 router.post('/verify-email', authenticateToken, verifyEmail);
28 router.get('/confirm-email', confirmEmail )
29 router.post('/reset-password', sendPasswordResetEmail)
30 router.get('/confirm-reset-password', resetPassword)
31
32 router.get('/questions',authenticateToken, questions)
33
34
35 router.post('/upload-profile-picture', authenticateToken, uploadProfilePicture);
36 router.get('/profile-picture', authenticateToken, getProfilePicture);
37
38 export default router

```

Ln 14, Col 57 Spaces:4 UTF-8 LF {} JavaScript ⌂ ⌂

chatSocket.js

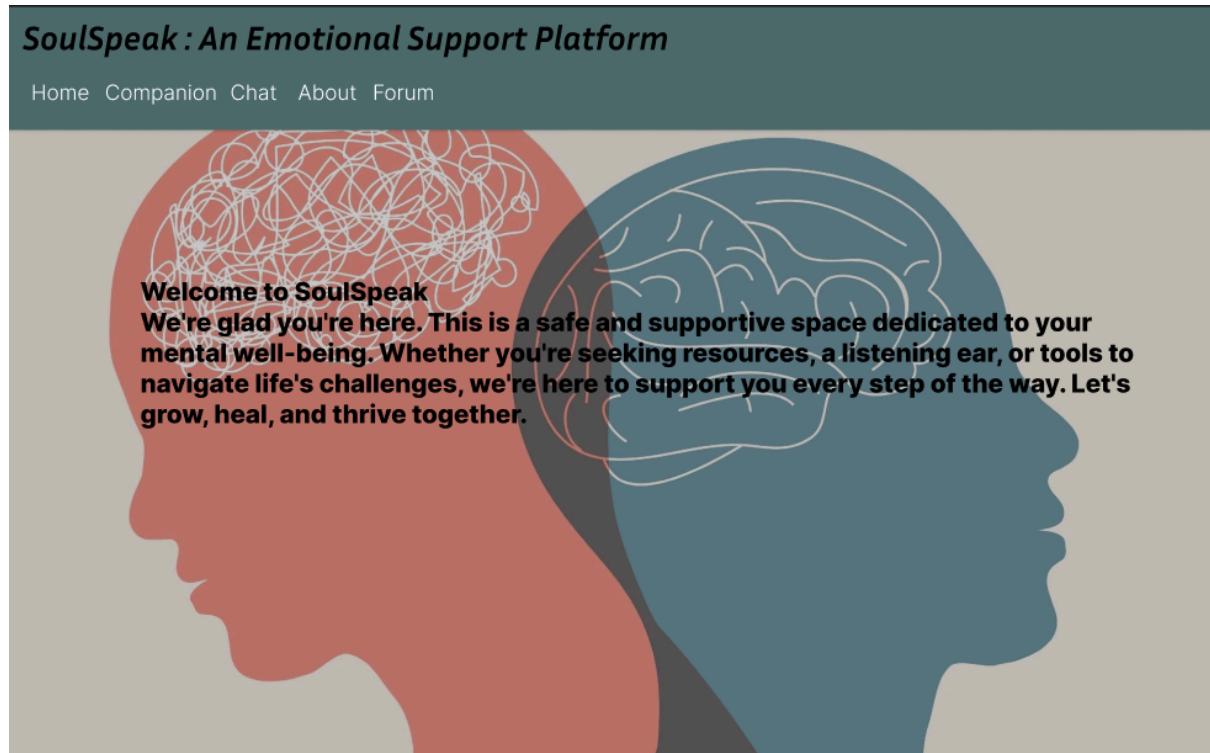
```
File Edit Selection View Go Run Terminal Help < > SoulSpeak
chatSocket.js
backend > socket > M chatSocket.js > ...
  1 import {Socket} from 'socket.io-client';
  2
  3 const chatSocket = (io) => {
  4   console.log('Chat socket is running');
  5
  6   io.on('connection', async (socket) => {
  7     console.log(`User connected: ${socket.id}`);
  8
  9     // Get user ID from auth token
 10    const token = socket.handshake.auth.token;
 11    if (token) {
 12      try {
 13        const decoded = jwt.verify(token, process.env.JWT_SECRET);
 14        const userId = decoded.id;
 15
 16        // Join a room specific to this user's ID
 17        socket.join(userId.toString());
 18        console.log(`User ${userId} joined their personal room`);
 19
 20        socket.on('join', (chatId) => {
 21          if (!chatId) {
 22            console.error('Invalid chatId');
 23            return;
 24          }
 25          socket.join(chatId);
 26          console.log(`User ${userId} joined chat: ${chatId}`);
 27        });
 28
 29        socket.on('leave chat', (chatId) => {
 30          socket.leave(chatId);
 31        });
 32
 33        socket.on('disconnect', () => {
 34          console.log(`User disconnected: ${socket.id}`);
 35        });
 36      } catch (error) {
 37        console.error('Invalid token:', error);
 38      }
 39    }
 40  });
 41
 42
 43 export default chatSocket;
44
```

2. User Interface Design

Attach at least 5 interface designs generated in Figma. Attach the figma project link.

Figma File Link:

https://www.figma.com/design/MZ5a4JW633fYjsr1RVA8AP/CSE471_Project_Figma?node-id=0-1&t=pCKcy4jucReTWE3V-1



SoulSpeak : An Emotional Support Platform

Home Companion Chat About Forum

Explore With us

Most Searched Blogs

Search forums

Explore Communities

New here!

Have some fun!

health and well being

Mental Health

About Us!

How Does Social Anxiety Affect the Brain?



See what's people posting



SoulSpeak : An Emotional Support Platform

Home Companion Chat About Forum

Popular paths ▾

- 10... Your journey to healing starts here.
- Mindfulness exercises →
- Self-help Guides →
- Health Assessment →
- Your Progress →

Your Timeline

Share your thoughts with us.....

Photo Link

My Communities ▾

Discover More →

Quick Links ▾

We are always here to help you



SoulSpeak : An Emotional Support Platform

Home Companion Chat About

Chat Request Initiation

Available Companions:

naruto341 sasuke432 kakashi57 sakura341

Notification:

You have one new chat request

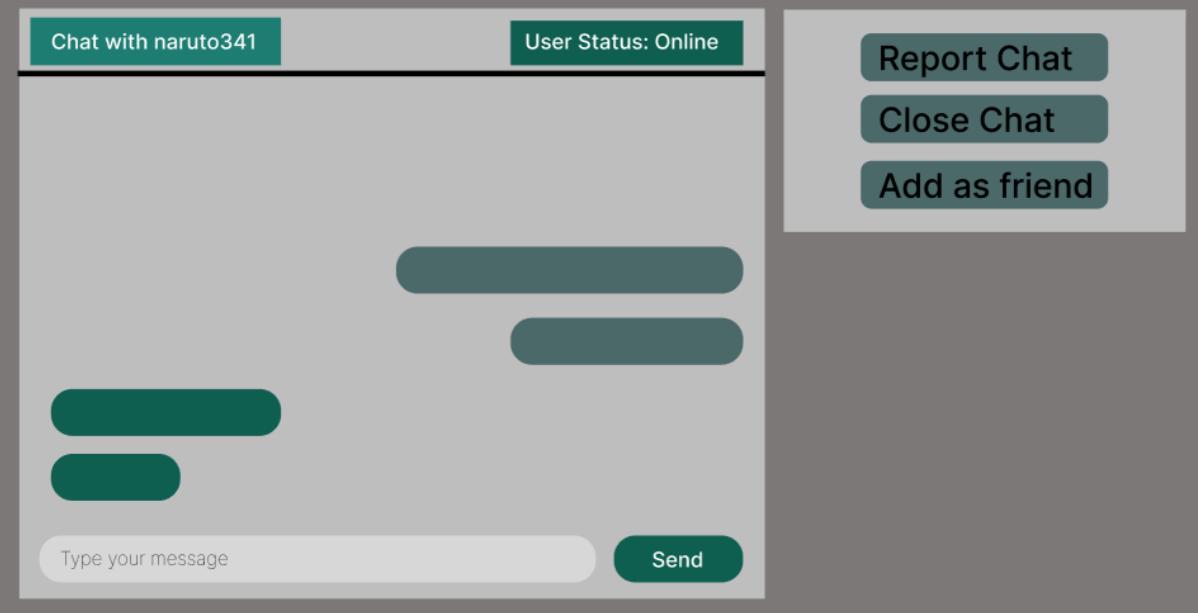
Accept

Decline

User Status: Active

SoulSpeak : An Emotional Support Platform

Home Companion Chat About



3. Frontend Development

Attach at least 5 frontend code snippets with proper description.

Admin.tsx (Handles front end interface of admin panel)

Chat.tsx (Handles the front end interface of Chat Window)

```
File Edit Selection View Go Run Terminal Help < > ⚡ SoulSpeak

Frontend > src > pages > @ Chatsx > @ Chat > @ initializeSocket > auth

21 interface Chat {
22   _id: string;
23   participants: User[];
24   messages: Message[];
25   lastMessage?: Message;
26 }
27
28 interface TypingStatus {
29   chatId: string;
30   userId: string;
31   username: string;
32 }
33
34 const Chat: React.FC<{}> = () => {
35   const [chat, setChat] = useState<Chat>({ _id: null });
36   const [currentChat, setCurrentChat] = useState<Chat | null>(null);
37   const [message, setMessage] = useState<string>("");
38   const [currentUser, setCurrentUser] = useState<User | null>(null);
39   const [error, setError] = useState<string>("");
40   const [typingStatus, setTypingStatus] = useState<TypingStatus | null>(null);
41   const [typingStatuses, setTypingStatuses] = useState<TypingStatus[] | null>([]);
42   const [showCompanions, setShowCompanions] = useState<boolean>(false);
43   const [user, setUser] = useState<User>({ _id: null });
44   const socket = useWebsocket("ws://localhost:5000");
45   const messageEndRef = useRef<HTMLDivElement>(null);
46   const typingEndRef = useRef<NodeJS.Timeout>();
47   const currentCharter = userRef.chat | null>(null);
48
49   const fetchCompanions = async () => {
50     try {
51       const response = await axios.get("http://localhost:5000/companions");
52       setCompanions(response.data);
53     } catch (error) {
54       setError("Failed to fetch companions");
55     }
56   };
57   const startCompanionChat = async (companionId: string) => {
58     try {
59       const response = await axios.post(
60         "http://localhost:5000/chat/create",
61         { participantId: companionId },
62         { headers: { Authorization: `Bearer ${localStorage.getItem('token')}` } }
63       );
64
65       // Set current chat immediately
66       setCurrentChat(response.data);
67
68       // Add to chats list if not already present
69       setPrevChats((prevChats: Chat[]) => {
70         if (!prevChats.find(chat => chat._id === response.data._id)) {
71           if (chatExists) {
72             return prevChats;
73           }
74         }
75         return [...prevChats, response.data];
76       });
77     } catch (error) {
78       setError("Failed to start companion chat");
79     }
80   };
81
82   useEffect(() => {
83     if (socket) {
84       socket.on("chat", (chat: Chat) => {
85         if (chat._id === currentCharter._id) {
86           setCurrentChat(chat);
87         } else {
88           setPrevChats((prevChats: Chat[]) => [
89             ...prevChats,
90             chat
91           ]);
92         }
93       });
94     }
95   }, [socket, currentCharter]);
96
97   const handleTextChange = (e: React.ChangeEvent<TextEventTarget>) => {
98     setMessage(e.target.value);
99   };
100
101   const handleSend = (e: React.MouseEvent<TextEventTarget>) => {
102     e.preventDefault();
103     if (socket) {
104       socket.emit("chat", { message, chatId: currentCharter._id });
105       setMessage("");
106     }
107   };
108
109   const handleTyping = (status: TypingStatus) => {
110     if (socket) {
111       socket.emit("typing", status);
112     }
113   };
114
115   const handleTypingEnd = () => {
116     if (socket) {
117       socket.emit("stopTyping", { chatId: currentCharter._id });
118     }
119   };
120
121   const handleCompanions = (companion: User) => {
122     if (socket) {
123       socket.emit("companion", { companionId: companion._id });
124     }
125   };
126
127   const handleUser = (user: User) => {
128     if (socket) {
129       socket.emit("user", { userId: user._id });
130     }
131   };
132
133   const handleSocketError = (error: Error) => {
134     if (socket) {
135       socket.close();
136     }
137   };
138
139   const handleSocketClose = () => {
140     if (socket) {
141       socket.close();
142     }
143   };
144
145   const handleSocketOpen = () => {
146     if (socket) {
147       socket.on("connect_error", handleSocketError);
148       socket.on("close", handleSocketClose);
149     }
150   };
151
152   const handleSocketMessage = (data: any) => {
153     if (socket) {
154       if (data.type === "chat") {
155         const chat = data.chat;
156         if (chat._id === currentCharter._id) {
157           setCurrentChat(chat);
158         } else {
159           setPrevChats((prevChats: Chat[]) => [
160             ...prevChats,
161             chat
162           ]);
163         }
164       } else if (data.type === "typing") {
165         const status = data.status;
166         if (status.chatId === currentCharter._id) {
167           setTypingStatus(status);
168         }
169       } else if (data.type === "companion") {
170         const companion = data.companion;
171         if (companion._id === currentCharter._id) {
172           startCompanionChat(companion);
173         }
174       } else if (data.type === "user") {
175         const user = data.user;
176         if (user._id === currentCharter._id) {
177           setUser(user);
178         }
179       }
180     }
181   };
182
183   const handleSocketConnect = () => {
184     if (socket) {
185       socket.on("connect", handleSocketOpen);
186       socket.on("message", handleSocketMessage);
187     }
188   };
189
190   const handleSocketReconnect = () => {
191     if (socket) {
192       socket.on("reconnect", handleSocketOpen);
193       socket.on("message", handleSocketMessage);
194     }
195   };
196
197   const handleSocketReconnecting = () => {
198     if (socket) {
199       socket.on("reconnecting", handleSocketOpen);
200       socket.on("message", handleSocketMessage);
201     }
202   };
203
204   const handleSocketDisconnect = () => {
205     if (socket) {
206       socket.off("connect_error", handleSocketError);
207       socket.off("close", handleSocketClose);
208     }
209   };
210
211   const handleSocketError = (error: Error) => {
212     if (socket) {
213       socket.close();
214     }
215   };
216
217   const handleSocketClose = () => {
218     if (socket) {
219       socket.close();
220     }
221   };
222
223   const handleSocketOpen = () => {
224     if (socket) {
225       socket.on("connect_error", handleSocketError);
226       socket.on("close", handleSocketClose);
227     }
228   };
229
230   const handleSocketMessage = (data: any) => {
231     if (socket) {
232       if (data.type === "chat") {
233         const chat = data.chat;
234         if (chat._id === currentCharter._id) {
235           setCurrentChat(chat);
236         } else {
237           setPrevChats((prevChats: Chat[]) => [
238             ...prevChats,
239             chat
240           ]);
241         }
242       } else if (data.type === "typing") {
243         const status = data.status;
244         if (status.chatId === currentCharter._id) {
245           setTypingStatus(status);
246         }
247       } else if (data.type === "companion") {
248         const companion = data.companion;
249         if (companion._id === currentCharter._id) {
250           startCompanionChat(companion);
251         }
252       } else if (data.type === "user") {
253         const user = data.user;
254         if (user._id === currentCharter._id) {
255           setUser(user);
256         }
257       }
258     }
259   };
260
261   const handleSocketConnect = () => {
262     if (socket) {
263       socket.on("connect", handleSocketOpen);
264       socket.on("message", handleSocketMessage);
265     }
266   };
267
268   const handleSocketReconnect = () => {
269     if (socket) {
270       socket.on("reconnect", handleSocketOpen);
271       socket.on("message", handleSocketMessage);
272     }
273   };
274
275   const handleSocketReconnecting = () => {
276     if (socket) {
277       socket.on("reconnecting", handleSocketOpen);
278       socket.on("message", handleSocketMessage);
279     }
280   };
281
282   const handleSocketDisconnect = () => {
283     if (socket) {
284       socket.off("connect_error", handleSocketError);
285       socket.off("close", handleSocketClose);
286     }
287   };
288
289   const handleSocketError = (error: Error) => {
290     if (socket) {
291       socket.close();
292     }
293   };
294
295   const handleSocketClose = () => {
296     if (socket) {
297       socket.close();
298     }
299   };
300
301   const handleSocketOpen = () => {
302     if (socket) {
303       socket.on("connect_error", handleSocketError);
304       socket.on("close", handleSocketClose);
305     }
306   };
307
308   const handleSocketMessage = (data: any) => {
309     if (socket) {
310       if (data.type === "chat") {
311         const chat = data.chat;
312         if (chat._id === currentCharter._id) {
313           setCurrentChat(chat);
314         } else {
315           setPrevChats((prevChats: Chat[]) => [
316             ...prevChats,
317             chat
318           ]);
319         }
320       } else if (data.type === "typing") {
321         const status = data.status;
322         if (status.chatId === currentCharter._id) {
323           setTypingStatus(status);
324         }
325       } else if (data.type === "companion") {
326         const companion = data.companion;
327         if (companion._id === currentCharter._id) {
328           startCompanionChat(companion);
329         }
330       } else if (data.type === "user") {
331         const user = data.user;
332         if (user._id === currentCharter._id) {
333           setUser(user);
334         }
335       }
336     }
337   };
338
339   const handleSocketConnect = () => {
340     if (socket) {
341       socket.on("connect", handleSocketOpen);
342       socket.on("message", handleSocketMessage);
343     }
344   };
345
346   const handleSocketReconnect = () => {
347     if (socket) {
348       socket.on("reconnect", handleSocketOpen);
349       socket.on("message", handleSocketMessage);
350     }
351   };
352
353   const handleSocketReconnecting = () => {
354     if (socket) {
355       socket.on("reconnecting", handleSocketOpen);
356       socket.on("message", handleSocketMessage);
357     }
358   };
359
360   const handleSocketDisconnect = () => {
361     if (socket) {
362       socket.off("connect_error", handleSocketError);
363       socket.off("close", handleSocketClose);
364     }
365   };
366
367   const handleSocketError = (error: Error) => {
368     if (socket) {
369       socket.close();
370     }
371   };
372
373   const handleSocketClose = () => {
374     if (socket) {
375       socket.close();
376     }
377   };
378
379   const handleSocketOpen = () => {
380     if (socket) {
381       socket.on("connect_error", handleSocketError);
382       socket.on("close", handleSocketClose);
383     }
384   };
385
386   const handleSocketMessage = (data: any) => {
387     if (socket) {
388       if (data.type === "chat") {
389         const chat = data.chat;
390         if (chat._id === currentCharter._id) {
391           setCurrentChat(chat);
392         } else {
393           setPrevChats((prevChats: Chat[]) => [
394             ...prevChats,
395             chat
396           ]);
397         }
398       } else if (data.type === "typing") {
399         const status = data.status;
400         if (status.chatId === currentCharter._id) {
401           setTypingStatus(status);
402         }
403       } else if (data.type === "companion") {
404         const companion = data.companion;
405         if (companion._id === currentCharter._id) {
406           startCompanionChat(companion);
407         }
408       } else if (data.type === "user") {
409         const user = data.user;
410         if (user._id === currentCharter._id) {
411           setUser(user);
412         }
413       }
414     }
415   };
416
417   const handleSocketConnect = () => {
418     if (socket) {
419       socket.on("connect", handleSocketOpen);
420       socket.on("message", handleSocketMessage);
421     }
422   };
423
424   const handleSocketReconnect = () => {
425     if (socket) {
426       socket.on("reconnect", handleSocketOpen);
427       socket.on("message", handleSocketMessage);
428     }
429   };
430
431   const handleSocketReconnecting = () => {
432     if (socket) {
433       socket.on("reconnecting", handleSocketOpen);
434       socket.on("message", handleSocketMessage);
435     }
436   };
437
438   const handleSocketDisconnect = () => {
439     if (socket) {
440       socket.off("connect_error", handleSocketError);
441       socket.off("close", handleSocketClose);
442     }
443   };
444
445   const handleSocketError = (error: Error) => {
446     if (socket) {
447       socket.close();
448     }
449   };
450
451   const handleSocketClose = () => {
452     if (socket) {
453       socket.close();
454     }
455   };
456
457   const handleSocketOpen = () => {
458     if (socket) {
459       socket.on("connect_error", handleSocketError);
460       socket.on("close", handleSocketClose);
461     }
462   };
463
464   const handleSocketMessage = (data: any) => {
465     if (socket) {
466       if (data.type === "chat") {
467         const chat = data.chat;
468         if (chat._id === currentCharter._id) {
469           setCurrentChat(chat);
470         } else {
471           setPrevChats((prevChats: Chat[]) => [
472             ...prevChats,
473             chat
474           ]);
475         }
476       } else if (data.type === "typing") {
477         const status = data.status;
478         if (status.chatId === currentCharter._id) {
479           setTypingStatus(status);
480         }
481       } else if (data.type === "companion") {
482         const companion = data.companion;
483         if (companion._id === currentCharter._id) {
484           startCompanionChat(companion);
485         }
486       } else if (data.type === "user") {
487         const user = data.user;
488         if (user._id === currentCharter._id) {
489           setUser(user);
490         }
491       }
492     }
493   };
494
495   const handleSocketConnect = () => {
496     if (socket) {
497       socket.on("connect", handleSocketOpen);
498       socket.on("message", handleSocketMessage);
499     }
500   };
501
502   const handleSocketReconnect = () => {
503     if (socket) {
504       socket.on("reconnect", handleSocketOpen);
505       socket.on("message", handleSocketMessage);
506     }
507   };
508
509   const handleSocketReconnecting = () => {
510     if (socket) {
511       socket.on("reconnecting", handleSocketOpen);
512       socket.on("message", handleSocketMessage);
513     }
514   };
515
516   const handleSocketDisconnect = () => {
517     if (socket) {
518       socket.off("connect_error", handleSocketError);
519       socket.off("close", handleSocketClose);
520     }
521   };
522
523   const handleSocketError = (error: Error) => {
524     if (socket) {
525       socket.close();
526     }
527   };
528
529   const handleSocketClose = () => {
530     if (socket) {
531       socket.close();
532     }
533   };
534
535   const handleSocketOpen = () => {
536     if (socket) {
537       socket.on("connect_error", handleSocketError);
538       socket.on("close", handleSocketClose);
539     }
540   };
541
542   const handleSocketMessage = (data: any) => {
543     if (socket) {
544       if (data.type === "chat") {
545         const chat = data.chat;
546         if (chat._id === currentCharter._id) {
547           setCurrentChat(chat);
548         } else {
549           setPrevChats((prevChats: Chat[]) => [
550             ...prevChats,
551             chat
552           ]);
553         }
554       } else if (data.type === "typing") {
555         const status = data.status;
556         if (status.chatId === currentCharter._id) {
557           setTypingStatus(status);
558         }
559       } else if (data.type === "companion") {
560         const companion = data.companion;
561         if (companion._id === currentCharter._id) {
562           startCompanionChat(companion);
563         }
564       } else if (data.type === "user") {
565         const user = data.user;
566         if (user._id === currentCharter._id) {
567           setUser(user);
568         }
569       }
570     }
571   };
572
573   const handleSocketConnect = () => {
574     if (socket) {
575       socket.on("connect", handleSocketOpen);
576       socket.on("message", handleSocketMessage);
577     }
578   };
579
580   const handleSocketReconnect = () => {
581     if (socket) {
582       socket.on("reconnect", handleSocketOpen);
583       socket.on("message", handleSocketMessage);
584     }
585   };
586
587   const handleSocketReconnecting = () => {
588     if (socket) {
589       socket.on("reconnecting", handleSocketOpen);
590       socket.on("message", handleSocketMessage);
591     }
592   };
593
594   const handleSocketDisconnect = () => {
595     if (socket) {
596       socket.off("connect_error", handleSocketError);
597       socket.off("close", handleSocketClose);
598     }
599   };
599
600   const handleSocketError = (error: Error) => {
601     if (socket) {
602       socket.close();
603     }
604   };
604
605   const handleSocketClose = () => {
606     if (socket) {
607       socket.close();
608     }
609   };
609
610   const handleSocketOpen = () => {
611     if (socket) {
612       socket.on("connect_error", handleSocketError);
613       socket.on("close", handleSocketClose);
614     }
615   };
616
617   const handleSocketMessage = (data: any) => {
618     if (socket) {
619       if (data.type === "chat") {
620         const chat = data.chat;
621         if (chat._id === currentCharter._id) {
622           setCurrentChat(chat);
623         } else {
624           setPrevChats((prevChats: Chat[]) => [
625             ...prevChats,
626             chat
627           ]);
628         }
629       } else if (data.type === "typing") {
630         const status = data.status;
631         if (status.chatId === currentCharter._id) {
632           setTypingStatus(status);
633         }
634       } else if (data.type === "companion") {
635         const companion = data.companion;
636         if (companion._id === currentCharter._id) {
637           startCompanionChat(companion);
638         }
639       } else if (data.type === "user") {
640         const user = data.user;
641         if (user._id === currentCharter._id) {
642           setUser(user);
643         }
644       }
645     }
646   };
647
648   const handleSocketConnect = () => {
649     if (socket) {
650       socket.on("connect", handleSocketOpen);
651       socket.on("message", handleSocketMessage);
652     }
653   };
654
655   const handleSocketReconnect = () => {
656     if (socket) {
657       socket.on("reconnect", handleSocketOpen);
658       socket.on("message", handleSocketMessage);
659     }
660   };
661
662   const handleSocketReconnecting = () => {
663     if (socket) {
664       socket.on("reconnecting", handleSocketOpen);
665       socket.on("message", handleSocketMessage);
666     }
667   };
668
669   const handleSocketDisconnect = () => {
670     if (socket) {
671       socket.off("connect_error", handleSocketError);
672       socket.off("close", handleSocketClose);
673     }
674   };
674
675   const handleSocketError = (error: Error) => {
676     if (socket) {
677       socket.close();
678     }
679   };
679
680   const handleSocketClose = () => {
681     if (socket) {
682       socket.close();
683     }
684   };
684
685   const handleSocketOpen = () => {
686     if (socket) {
687       socket.on("connect_error", handleSocketError);
688       socket.on("close", handleSocketClose);
689     }
690   };
691
692   const handleSocketMessage = (data: any) => {
693     if (socket) {
694       if (data.type === "chat") {
695         const chat = data.chat;
696         if (chat._id === currentCharter._id) {
697           setCurrentChat(chat);
698         } else {
699           setPrevChats((prevChats: Chat[]) => [
700             ...prevChats,
701             chat
702           ]);
703         }
704       } else if (data.type === "typing") {
705         const status = data.status;
706         if (status.chatId === currentCharter._id) {
707           setTypingStatus(status);
708         }
709       } else if (data.type === "companion") {
710         const companion = data.companion;
711         if (companion._id === currentCharter._id) {
712           startCompanionChat(companion);
713         }
714       } else if (data.type === "user") {
715         const user = data.user;
716         if (user._id === currentCharter._id) {
717           setUser(user);
718         }
719       }
720     }
721   };
722
723   const handleSocketConnect = () => {
724     if (socket) {
725       socket.on("connect", handleSocketOpen);
726       socket.on("message", handleSocketMessage);
727     }
728   };
729
730   const handleSocketReconnect = () => {
731     if (socket) {
732       socket.on("reconnect", handleSocketOpen);
733       socket.on("message", handleSocketMessage);
734     }
735   };
736
737   const handleSocketReconnecting = () => {
738     if (socket) {
739       socket.on("reconnecting", handleSocketOpen);
740       socket.on("message", handleSocketMessage);
741     }
742   };
743
744   const handleSocketDisconnect = () => {
745     if (socket) {
746       socket.off("connect_error", handleSocketError);
747       socket.off("close", handleSocketClose);
748     }
749   };
749
750   const handleSocketError = (error: Error) => {
751     if (socket) {
752       socket.close();
753     }
754   };
754
755   const handleSocketClose = () => {
756     if (socket) {
757       socket.close();
758     }
759   };
759
760   const handleSocketOpen = () => {
761     if (socket) {
762       socket.on("connect_error", handleSocketError);
763       socket.on("close", handleSocketClose);
764     }
765   };
766
767   const handleSocketMessage = (data: any) => {
768     if (socket) {
769       if (data.type === "chat") {
770         const chat = data.chat;
771         if (chat._id === currentCharter._id) {
772           setCurrentChat(chat);
773         } else {
774           setPrevChats((prevChats: Chat[]) => [
775             ...prevChats,
776             chat
777           ]);
778         }
779       } else if (data.type === "typing") {
780         const status = data.status;
781         if (status.chatId === currentCharter._id) {
782           setTypingStatus(status);
783         }
784       } else if (data.type === "companion") {
785         const companion = data.companion;
786         if (companion._id === currentCharter._id) {
787           startCompanionChat(companion);
788         }
789       } else if (data.type === "user") {
790         const user = data.user;
791         if (user._id === currentCharter._id) {
792           setUser(user);
793         }
794       }
795     }
796   };
797
798   const handleSocketConnect = () => {
799     if (socket) {
800       socket.on("connect", handleSocketOpen);
801       socket.on("message", handleSocketMessage);
802     }
803   };
804
805   const handleSocketReconnect = () => {
806     if (socket) {
807       socket.on("reconnect", handleSocketOpen);
808       socket.on("message", handleSocketMessage);
809     }
810   };
811
812   const handleSocketReconnecting = () => {
813     if (socket) {
814       socket.on("reconnecting", handleSocketOpen);
815       socket.on("message", handleSocketMessage);
816     }
817   };
818
819   const handleSocketDisconnect = () => {
820     if (socket) {
821       socket.off("connect_error", handleSocketError);
822       socket.off("close", handleSocketClose);
823     }
824   };
824
825   const handleSocketError = (error: Error) => {
826     if (socket) {
827       socket.close();
828     }
829   };
829
830   const handleSocketClose = () => {
831     if (socket) {
832       socket.close();
833     }
834   };
834
835   const handleSocketOpen = () => {
836     if (socket) {
837       socket.on("connect_error", handleSocketError);
838       socket.on("close", handleSocketClose);
839     }
840   };
841
842   const handleSocketMessage = (data: any) => {
843     if (socket) {
844       if (data.type === "chat") {
845         const chat = data.chat;
846         if (chat._id === currentCharter._id) {
847           setCurrentChat(chat);
848         } else {
849           setPrevChats((prevChats: Chat[]) => [
850             ...prevChats,
851             chat
852           ]);
853         }
854       } else if (data.type === "typing") {
855         const status = data.status;
856         if (status.chatId === currentCharter._id) {
857           setTypingStatus(status);
858         }
859       } else if (data.type === "companion") {
860         const companion = data.companion;
861         if (companion._id === currentCharter._id) {
862           startCompanionChat(companion);
863         }
864       } else if (data.type === "user") {
865         const user = data.user;
866         if (user._id === currentCharter._id) {
867           setUser(user);
868         }
869       }
870     }
871   };
872
873   const handleSocketConnect = () => {
874     if (socket) {
875       socket.on("connect", handleSocketOpen);
876       socket.on("message", handleSocketMessage);
877     }
878   };
879
880   const handleSocketReconnect = () => {
881     if (socket) {
882       socket.on("reconnect", handleSocketOpen);
883       socket.on("message", handleSocketMessage);
884     }
885   };
886
887   const handleSocketReconnecting = () => {
888     if (socket) {
889       socket.on("reconnecting", handleSocketOpen);
890       socket.on("message", handleSocketMessage);
891     }
892   };
893
894   const handleSocketDisconnect = () => {
895     if (socket) {
896       socket.off("connect_error", handleSocketError);
897       socket.off("close", handleSocketClose);
898     }
899   };
899
900   const handleSocketError = (error: Error) => {
901     if (socket) {
902       socket.close();
903     }
904   };
904
905   const handleSocketClose = () => {
906     if (socket) {
907       socket.close();
908     }
909   };
909
910   const handleSocketOpen = () => {
911     if (socket) {
912       socket.on("connect_error", handleSocketError);
913       socket.on("close", handleSocketClose);
914     }
915   };
916
917   const handleSocketMessage = (data: any) => {
918     if (socket) {
919       if (data.type === "chat") {
920         const chat = data.chat;
921         if (chat._id === currentCharter._id) {
922           setCurrentChat(chat);
923         } else {
924           setPrevChats((prevChats: Chat[]) => [
925             ...prevChats,
926             chat
927           ]);
928         }
929       } else if (data.type === "typing") {
930         const status = data.status;
931         if (status.chatId === currentCharter._id) {
932           setTypingStatus(status);
933         }
934       } else if (data.type === "companion") {
935         const companion = data.companion;
936         if (companion._id === currentCharter._id) {
937           startCompanionChat(companion);
938         }
939       } else if (data.type === "user") {
940         const user = data.user;
941         if (user._id === currentCharter._id) {
942           setUser(user);
943         }
944       }
945     }
946   };
947
948   const handleSocketConnect = () => {
949     if (socket) {
950       socket.on("connect", handleSocketOpen);
951       socket.on("message", handleSocketMessage);
952     }
953   };
954
955   const handleSocketReconnect = () => {
956     if (socket) {
957       socket.on("reconnect", handleSocketOpen);
958       socket.on("message", handleSocketMessage);
959     }
960   };
961
962   const handleSocketReconnecting = () => {
963     if (socket) {
964       socket.on("reconnecting", handleSocketOpen);
965       socket.on("message", handleSocketMessage);
966     }
967   };
968
969   const handleSocketDisconnect = () => {
970     if (socket) {
971       socket.off("connect_error", handleSocketError);
972       socket.off("close", handleSocketClose);
973     }
974   };
974
975   const handleSocketError = (error: Error) => {
976     if (socket) {
977       socket.close();
978     }
979   };
979
980   const handleSocketClose = () => {
981     if (socket) {
982       socket.close();
983     }
984   };
984
985   const handleSocketOpen = () => {
986     if (socket) {
987       socket.on("connect_error", handleSocketError);
988       socket.on("close", handleSocketClose);
989     }
990   };
991
992   const handleSocketMessage = (data: any) => {
993     if (socket) {
994       if (data.type === "chat") {
995         const chat = data.chat;
996         if (chat._id === currentCharter._id) {
997           setCurrentChat(chat);
998         } else {
999           setPrevChats((prevChats: Chat[]) => [
1000             ...prevChats,
1001             chat
1002           ]);
1003         }
1004       } else if (data.type === "typing") {
1005         const status = data.status;
1006         if (status.chatId === currentCharter._id) {
1007           setTypingStatus(status);
1008         }
1009       } else if (data.type === "companion") {
1010         const companion = data.companion;
1011         if (companion._id === currentCharter._id) {
1012           startCompanionChat(companion);
1013         }
1014       } else if (data.type === "user") {
1015         const user = data.user;
1016         if (user._id === currentCharter._id) {
1017           setUser(user);
1018         }
1019       }
1020     }
1021   };
1022
1023   const handleSocketConnect = () => {
1024     if (socket) {
1025       socket.on("connect", handleSocketOpen);
1026       socket.on("message", handleSocketMessage);
1027     }
1028   };
1029
1030   const handleSocketReconnect = () => {
1031     if (socket) {
1032       socket.on("reconnect", handleSocketOpen);
1033       socket.on("message", handleSocketMessage);
1034     }
1035   };
1036
1037   const handleSocketReconnecting = () => {
1038     if (socket) {
1039       socket.on("reconnecting", handleSocketOpen);
1040       socket.on("message", handleSocketMessage);
1041     }
1042   };
1043
1044   const handleSocketDisconnect = () => {
1045     if (socket) {
1046       socket.off("connect_error", handleSocketError);
1047       socket.off("close", handleSocketClose);
1048     }
1049   };
1049
1050   const handleSocketError = (error: Error) => {
1051     if (socket) {
1052       socket.close();
1053     }
1054   };
1054
1055   const handleSocketClose = () => {
1056     if (socket) {
1057       socket.close();
1058     }
1059   };
1059
1060   const handleSocketOpen = () => {
1061     if (socket) {
1062       socket.on("connect_error", handleSocketError);
1063       socket.on("close", handleSocketClose);
1064     }
1065   };
1066
1067   const handleSocketMessage = (data: any) => {
1068     if (socket) {
1069       if (data.type === "chat") {
1070         const chat = data.chat;
1071         if (chat._id === currentCharter._id) {
1072           setCurrentChat(chat);
1073         } else {
1074           setPrevChats((prevChats: Chat[]) => [
1075             ...prevChats,
1076             chat
1077           ]);
1078         }
1079       } else if (data.type === "typing") {
1080         const status = data.status;
1081         if (status.chatId === currentCharter._id) {
1082           setTypingStatus(status);
1083         }
1084       } else if (data.type === "companion") {
1085         const companion = data.companion;
1086         if (companion._id === currentCharter._id) {
1087           startCompanionChat(companion);
1088         }
1089       } else if (data.type === "user") {
1090         const user = data.user;
1091         if (user._id === currentCharter._id) {
1092           setUser(user);
1093         }
1094       }
1095     }
1096   };
1097
1098   const handleSocketConnect = () => {
1099     if (socket) {
1100       socket.on("connect", handleSocketOpen);
1101       socket.on("message", handleSocketMessage);
1102     }
1103   };
1104
1105   const handleSocketReconnect = () => {
1106     if (socket) {
1107       socket.on("reconnect", handleSocketOpen);
1108       socket.on("message", handleSocketMessage);
1109     }
1110   };
1111
1112   const handleSocketReconnecting = () => {
1113     if (socket) {
1114       socket.on("reconnecting", handleSocketOpen);
1115       socket.on("message", handleSocketMessage);
1116     }
1117   };
1118
1119   const handleSocketDisconnect = () => {
1120     if (socket) {
1121       socket.off("connect_error", handleSocketError);
1122       socket.off("close", handleSocketClose);
1123     }
1124   };
1124
1125   const handleSocketError = (error: Error) => {
1126     if (socket) {
1127       socket.close();
1128     }
1129   };
1129
1130   const handleSocketClose = () => {
1131     if (socket) {
1132       socket.close();
1133     }
1134   };
1134
1135   const handleSocketOpen = () => {
1136     if (socket) {
1137       socket.on("connect_error", handleSocketError);
1138       socket.on("close", handleSocketClose);
1139     }
1140   };
1141
1142   const handleSocketMessage = (data: any) => {
1143     if (socket) {
1144       if (data.type === "chat") {
1145         const chat = data.chat;
1146         if (chat._id === currentCharter._id) {
1147           setCurrentChat(chat);
1148         } else {
1149           setPrevChats((prevChats: Chat[]) => [
1150             ...prevChats,
1151             chat
1152           ]);
1153         }
1154       } else if (data.type === "typing") {
1155         const status = data.status;
1156         if (status.chatId === currentCharter._id) {
1157           setTypingStatus(status);
1158         }
1159       } else if (data.type === "companion") {
1160         const companion = data.companion;
1161         if (companion._id === currentCharter._id) {
1162           startCompanionChat(companion);
1163         }
1164       } else if (data.type === "user") {
1165         const user = data.user;
1166         if (user._id === currentCharter._id) {
1167           setUser(user);
1168         }
1169       }
1170     }
1171   };
1172
1173   const handleSocketConnect = () => {
1174     if (socket) {
1175       socket.on("connect", handleSocketOpen);
1176       socket.on("message", handleSocketMessage);
1177     }
1178   };
1179
1180   const handleSocketReconnect = () => {
1181     if (socket) {
1182       socket.on("reconnect", handleSocketOpen);
1183       socket.on("message", handleSocketMessage);
1184     }
1185   };
1186
1187   const handleSocketReconnecting = () => {
1188     if (socket) {
1189       socket.on("reconnecting", handleSocketOpen);
1190       socket.on("message", handleSocketMessage);
1191     }
1192   };
1193
1194   const handleSocketDisconnect = () => {
1195     if (socket) {
1196       socket.off("connect_error", handleSocketError);
1197       socket.off("close", handleSocketClose);
1198     }
1199   };
1199
1200   const handleSocketError = (error: Error) => {
1201     if (socket) {
1202       socket.close();
1203     }
1204   };
1204
1205   const handleSocketClose = () => {
1206     if (socket) {
1207       socket.close();
1208     }
1209   };
1209
1210   const handleSocketOpen = () => {
1211     if (socket) {
1212       socket.on("connect_error", handleSocketError);
1213       socket.on("close", handleSocketClose);
1214     }
1215   };
1216
1217   const handleSocketMessage = (data: any) => {
1218     if (socket) {
1219       if (data.type === "chat") {
1220         const chat = data.chat;
1221         if (chat._id === currentCharter._id) {
1222           setCurrentChat(chat);
1223         } else {
1224           setPrevChats((prevChats: Chat[]) => [
1225             ...prevChats,
1226             chat
1227           ]);
1228         }
1229       } else if (data.type === "typing") {
1230         const status = data.status;
1231         if (status.chatId === currentCharter._id) {
1232           setTypingStatus(status);
1233         }
1234       } else if (data.type === "companion") {
1235         const companion = data.companion;
1236         if (companion._id === currentCharter._id) {
1237           startCompanionChat(companion);
1238         }
1239       } else if (data.type === "user") {
1240         const user = data.user;
1241         if (user._id === currentCharter._id) {
1242           setUser(user);
1243         }
1244       }
1245     }
1246   };
1247
1248   const handleSocketConnect = () => {
1249     if (socket) {
1250       socket.on("connect", handleSocketOpen);
1251       socket.on("message", handleSocketMessage);
1252     }
1253   };
1254
1255   const handleSocketReconnect = () => {
1256     if (socket) {
1257       socket.on("reconnect", handleSocketOpen);
1258       socket.on("message", handleSocketMessage);
1259     }
1260
```

CompanionTest.jsx(Handles the front end interface of Questioning the User for companion compatibility)

```

    import React from 'react';
    import { useState } from 'react';
    import axios from 'axios';
    import { useNavigate } from 'react-router-dom';

    const CompanionText = () => {
        const [questionIndex, setQuestionIndex] = useState(0);
        const [userAnswers, setUserAnswers] = useState({});

        const handleAnswer = (questionIndex, optionIndex) => {
            setUserAnswers((prev) => ({ ...prev, [questionIndex]: optionIndex }));
        };

        const makeUserCompanion = async (percentage) => {
            try {
                const token = localStorage.getItem('token');
                console.log(token);
                const response = await axios.post('http://localhost:5000/companion', { percentage }, {
                    headers: { Authorization: `Bearer ${token}` },
                });
            } catch (error) {
                console.error(`Error registering as companion: ${error}`);
            }
        };

        const handleSubmit = () => {
            setQuestionIndex(0);
            setUserAnswers({});
            let finalScore = 0;

            Object.keys(userAnswers).forEach(questionIndex => {
                if (userAnswers[questionIndex] === questions[questionIndex].correct) {
                    finalScore += 1;
                }
            });

            setScore(finalScore);
            const percentage = (finalScore / questions.length) * 100;
            if (percentage >= 70) {
                makeUserCompanion(percentage);
                setShouldDefault(true);
                setShowing(false);
            }
        };

        const renderResultContent = () => {
            const percentage = (score / questions.length) * 100;
            if (percentage >= 70) {
                return (
                    <div className="text-center space-y-4">
                        <h3 className="text-2xl font-bold" style={{ color: '#4D9A6D' }}>
                            Congratulations!
                        </h3>
                        <p>
                            You've demonstrated excellent understanding and empathy skills.
                            You're ready to be a SoulSpeak companion!
                        </p>
                    </div>
                );
            }
        };
    };

    return (
        <div>
            <h1>SoulSpeak</h1>
            <h2>Companion</h2>
            <h3>Answer the questions below to earn points!</h3>
            <div>
                <h4>Question <span>{questionIndex + 1}</span></h4>
                <div>
                    <h5>What is the capital of France?</h5>
                    <ul style={{ listStyleType: 'none' }}>
                        <li>A. Paris</li>
                        <li>B. London</li>
                        <li>C. Rome</li>
                        <li>D. Berlin</li>
                    </ul>
                </div>
                <div>
                    <button onClick={() => handleAnswer(questionIndex, 0)}>A</button>
                    <button onClick={() => handleAnswer(questionIndex, 1)}>B</button>
                    <button onClick={() => handleAnswer(questionIndex, 2)}>C</button>
                    <button onClick={() => handleAnswer(questionIndex, 3)}>D</button>
                </div>
            </div>
            <div>
                <button onClick={handleSubmit}>Submit</button>
            </div>
            <div>
                {renderResultContent()}
            </div>
        </div>
    );
};

export default CompanionText;

```

Forum.tsx (Handles the front end interface of forum page)

```

    import React from 'react';
    import { useState } from 'react';
    import axios from 'axios';
    import { useNavigate } from 'react-router-dom';

    const Login = () => {
        const [formData, setFormData] = useState({
            email: '',
            password: ''
        });
        const [showEmailPopUp, setShowEmailPopUp] = useState(false);
        const [resetEmail, setResetEmail] = useState('');
        const [error, setError] = useState(null);
        const [navigate, setNavigate] = useState('');

        const handleChange = (e) => {
            const { name, value } = e.target;
            setFormData({ ...formData, [name]: value });
        };

        const handleSubmit = async (e) => {
            e.preventDefault();
            setError(null);
            try {
                const response = await axios.post('http://localhost:5000/login', formData);
                if (response.status === 200) {
                    localStorage.setItem('token', response.data.token);
                    navigate('/');
                }
            } catch (error) {
                setError(error.response.data.message || 'Invalid email or password.');
            }
        };

        const handleForgotPassword = async () => {
            setShowEmailPopUp(true);
        };

        const handleResetSubmit = async () => {
            try {
                const response = await axios.post('http://localhost:5000/reset-password', { email: resetEmail });
                if (response.status === 200) {
                    alert('Reset password email sent. Please check your inbox!');
                    setShowEmailPopUp(false);
                    setResetEmail('');
                }
            } catch (error) {
                alert('Error sending reset email. Please try again.');
            }
        };
    };

    return (
        <div>
            <h1>SoulSpeak</h1>
            <h2>Login</h2>
            <div>
                <h3>Enter your credentials</h3>
                <div>
                    <input type="text" name="email" placeholder="Email" value={formData.email} onChange={handleChange} />
                    <input type="password" name="password" placeholder="Password" value={formData.password} onChange={handleChange} />
                </div>
                <div>
                    <button onClick={handleSubmit}>Login</button>
                </div>
                <div>
                    <button onClick={handleForgotPassword}>Forgot Password?</button>
                </div>
            </div>
            <div>
                {showEmailPopUp ? <div>Email sent!</div> : null}
            </div>
        </div>
    );
};

export default Login;

```

Login.jsx (Handles the front end interface of Login page)

```
File Edit Selection View Go Run Terminal Help < > SoulSpeak
```

```
Forum.tsx
```

```
const Forum = () => {
  const [selectedCategory, setSelectedCategory] = useState<string>('all');
  const [postCategory, setPostCategory] = useState<string>('General');

  const categories = ['Welcome Center', 'Mindful Moments', 'Wellness Hub', 'Support Circle', 'General'];
  const [editingPost, setEditingPost] = useState<string>(null);

  return (
    <div>
      <h1>My Forum</h1>
      <div>
        <ul>
          {categories.map((category) => (
            <li key={category}>
              <button onClick={() => setSelectedCategory(category)}>{category}</button>
            <ul>
              {category === 'all' ? (
                <li><LinkIcon /> Home</li>
                <li><HeartIcon /> Heart</li>
                <li><BrainIcon /> Brain</li>
                <li><CloudIcon /> Cloud</li>
                <li><SunIcon /> Sun</li>
                <li><ChevronRightIcon /> Chevron Right</li>
                <li><SendIcon /> Send</li>
                <li><LinkIcon /> Link</li>
                <li><ImageIcon /> Image</li>
                <li><ChevronDownIcon /> Chevron Down</li>
                <li><UserIcon /> Users</li>
                <li><SearchIcon /> Search</li>
                <li><PencilIcon /> Pencil</li>
                <li><TrashIcon /> Trash</li>
                <li><ThumbsUpIcon /> Thumbs Up</li>
                <li><ThumbsDownIcon /> Thumbs Down</li>
                <li><HeartOutlineIcon /> Heart Outline</li>
              ) : null}
            </ul>
          )}
        </ul>
      </div>
      <div>
        <h2>Recent Posts</h2>
        <ul>
          {posts.map((post) => (
            <li key={post._id}>
              <div>
                <img alt={post.avatar} />
                <div>
                  <strong>{post.username}</strong>
                  <p>{post.content}</p>
                  <small>Created at: {post.createdAt}</small>
                </div>
              </div>
            </li>
          ))}
        </ul>
      </div>
    </div>
  );
}

export default Forum;
```

Profile.jsx ((Handles the front end interface of profile updating))

Report.tsx ((Handles the front end interface of user side reporting page))

```

import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { Plus, X } from 'lucide-react';

interface Report {
  _id: string;
  title: string;
  type: string;
  details: string;
  status: string;
}

const Report = () => {
  const [profiled, setProfiled] = useState("");
  const [reports, setReports] = useState([]);
  const [showModal, setShowModal] = useState(false);
  const [formData, setFormData] = useState({
    review: '',
    details: '',
    profiled: '',
    chatId: ''
  });

  const FetchUserEmail = async () => {
    const token = localStorage.getItem("token");
    try {
      const response = await axios.get(`http://localhost:5000/profile`, {
        headers: { Authorization: `Bearer ${token}` }
      });
      setProfiled(response.data.user.email);
    } catch (error) {
      console.error("Failed to fetch user email");
    }
  };

  const FetchReports = async () => {
    const token = localStorage.getItem("token");
    try {
      const response = await axios.get(`http://localhost:5000/reports`, {
        headers: { Authorization: `Bearer ${token}` }
      });
      setReports(response.data);
    } catch (error) {
      console.error("Failed to fetch reports");
    }
  };

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    const token = localStorage.getItem("token");
    let enhancedDetails = '';
    if (formData.type === 'profile-report' && formData.profileId) {
      enhancedDetails = `Profile: ${formData.profileId}\n${formData.details}`;
    } else if (formData.type === 'chat-report' && formData.chatId) {
      enhancedDetails = `Chat ID: ${formData.chatId}\n${formData.details}`;
    }
  };
}

```

ResetPassword.tsx

```

const ResetPassword = () => {
  const ResetPassword = () => {
    const handleSubmit = async (e: React.FormEvent) => {
      if (password !== confirmPassword) {
        setError('Passwords do not match');
        return;
      }

      try {
        const response = await axios.post(`http://localhost:5000/confirm-reset-password`, {
          password,
          token
        });

        setMessage('Password reset successful!');
        setTimeout(() => {
          navigate('/login');
        }, 2000);
      } catch (err) {
        setError('Failed to reset password. Please try again.');
      }
    };

    return (
      <div className="min-h-screen flex items-center justify-center py-12 px-4" style={{ background: '#ECECEC' }}>
        <div className="max-w-md w-full space-y-8 p-10 rounded-2xl shadow-xl" style={{ background: 'white' }}>
          <div>
            <h2>Reset Password</h2>
            <div>
              <div>
                <p>{message}</p>
              </div>
            </div>
            <div>
              <form onSubmit={handleSubmit} className="space-y-6">
                <div>
                  <label>New Password</label>
                  <input type="password" value={password} onChange={(e) => setPassword(e.target.value)} className="w-full px-3 py-3 rounded-2xl transition-all duration-300 focus:outline-none focus:ring-2 border-l-2 border-solid #EAEAEA" style={{ color: '#EAEAEA' }} />
                </div>
                <div>
                  <button type="submit" style={{ background: '#FFB703', color: '#EAEAEA' }}>Reset</button>
                </div>
              </form>
            </div>
          </div>
        </div>
      </div>
    );
  };
}

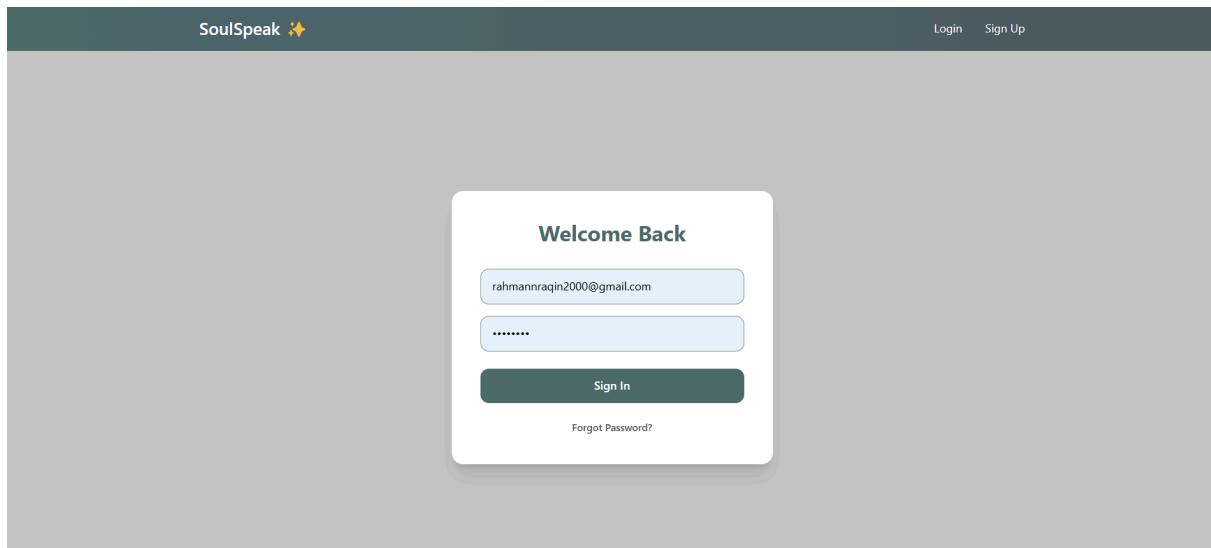
```

Signup.jsx (Handles the front end interface of signup page)

TrainingProgram.jsx (Handles the front end interface of Resource Training page)

4. User Manual

How a new user will navigate or use the system with screenshots.



SoulSpeak Companion Assessment

Question 1

You are assisting an elderly client who prefers to take their medication with breakfast. One morning, you notice the medication is missing from its usual spot. What should you do first?

- Search the entire house for the medication.
- Ask the client if they remember where they last placed the medication.
- Inform the client that they cannot have breakfast until the medication is found.
- Immediately contact the client's physician for a replacement prescription.

Question 2

You notice that a friend has been withdrawing from social activities, appearing unusually tired, and frequently expressing feelings of worthlessness. What would be the best course of action?

- Ignore the behavior; they'll recover on their own.

Chats

[Find Companions](#)

Available Companions

[Back to Chats](#)**edward12**

Companion

Admin

Companion

ZCutie

Companion

raqp

Companion

Tas

Companion

Select a chat to start messaging

Chats

[Find Companions](#)

Tas (Chat ID: 677d27b07d79980a9ccfe56)

Admin

19:10

Hello How are you doing?

19:10

edward12

19:10

Tas

19:10

Type a message...



SoulSpeak ✨

Become A Companion Chat Forum Report Profile Logout



ABC Khan

khan31

abc@gmail.com

How did you find us?

Anxiety

Under 18

Ranulaksh

SoulSpeak ✨

Become A Companion Chat Forum Report Profile Logout



ABC Khan

Email: abc@gmail.com

Edit Profile

SoulSpeak ✨

Become A Companion Chat Forum Report Profile Logout

Reports

+ New Report

Time	Type	Details	Status



This screenshot shows the "SoulSpeak Training Program" screen. At the top, the "SoulSpeak" logo is visible. The main content area features two topics in separate boxes:

- Topic 1:** You are assisting an elderly client who prefers to take their medication with breakfast. One morning, you notice the medication is missing from its usual spot. What should you do first?
Related Resources:
<https://www.youtube.com/watch?v=0af7GOTMh7M>
- Topic 2:** You notice that a friend has been withdrawing from social activities, appearing unusually tired, and frequently expressing feelings of worthlessness. What would be the best course of action?
Related Resources:
https://www.thesun.ie/health/10748691/depression-symptoms-surprising/?utm_source=chatgpt.com

This screenshot shows the "Questions Management" screen within the "Admin Panel". The left sidebar lists management options: "Questions Management" (which is selected and highlighted in white), "Users Management", "Reports Management", and "Suspension Management". The main content area is titled "Questions Management" and contains a form with the following fields:

- Question
- Option1
- Option2
- Option3
- Option4
- Correct
- Resource

A dark blue "Add Question" button is located at the bottom of the form.

You are assisting an elderly client who prefers to take their medication with breakfast. One morning, you notice the medication is missing from its usual spot. What should you do first?

- Search the entire house for the medication.
- Ask the client if they remember where they last placed the medication.
- Inform the client that they cannot have breakfast until the medication is found.
- Immediately contact the client's physician for a replacement prescription.

Answer: 1

Resource: <https://www.youtube.com/watch?v=0af7G0TMh7M>

[Edit](#) [Delete](#)

You notice that a friend has been withdrawing from social activities, appearing unusually tired, and frequently expressing feelings of worthlessness. What would be the best course of action?

- Ignore the behavior; they'll recover on their own.
- Encourage them to open up about their feelings and suggest seeking professional help.
- Tell them they need to be more positive and focus on the bright side of life.
- Avoid them until they feel better to give them space.

Answer: 1

Resource: https://www.thesun.ie/health/10748691/depression-symptoms-surprising/?utm_source=chatgpt.com

[Edit](#) [Delete](#)

SoulSpeak

Become A Companion Chat Forum Report Profile Logout

Admin Panel

- Questions Management
- [Users Management](#)
- Reports Management
- Suspension Management

User Management

USERNAME	EMAIL	ROLE
edward12	saumik.das.turja@g.bracu.ac.bd	Companion
sdfgsdf	abdullah.al.zubayer@g.bracu.ac.bd	User
qwe	qwe@gmail.com	User
TJ	saumik.das.turja@gmail.com	User
Admin	admin@gmail.com	Companion
Gwen	sultanaishaiera09@gmail.com	User
ZCutie	Sadia@gmai.com	Companion
raqp	rahmannraqin2000@gmail.com	Companion
gnadhi11	gandhi@gmail.com	User
-	-	-

SoulSpeak ✨

Become A Companion Chat Forum Report Profile Logout

Admin Panel

- Questions Management
- Users Management
- Reports Management**
- Suspension Management

Reports Management

Generate PDF Report

Email	Type	Details	Time	Status	ACTIONS

SoulSpeak ✨

Become A Companion Chat Forum Report Profile Logout

Admin Panel

- Questions Management
- Users Management
- Reports Management**
- Suspension Management

User Suspension Management

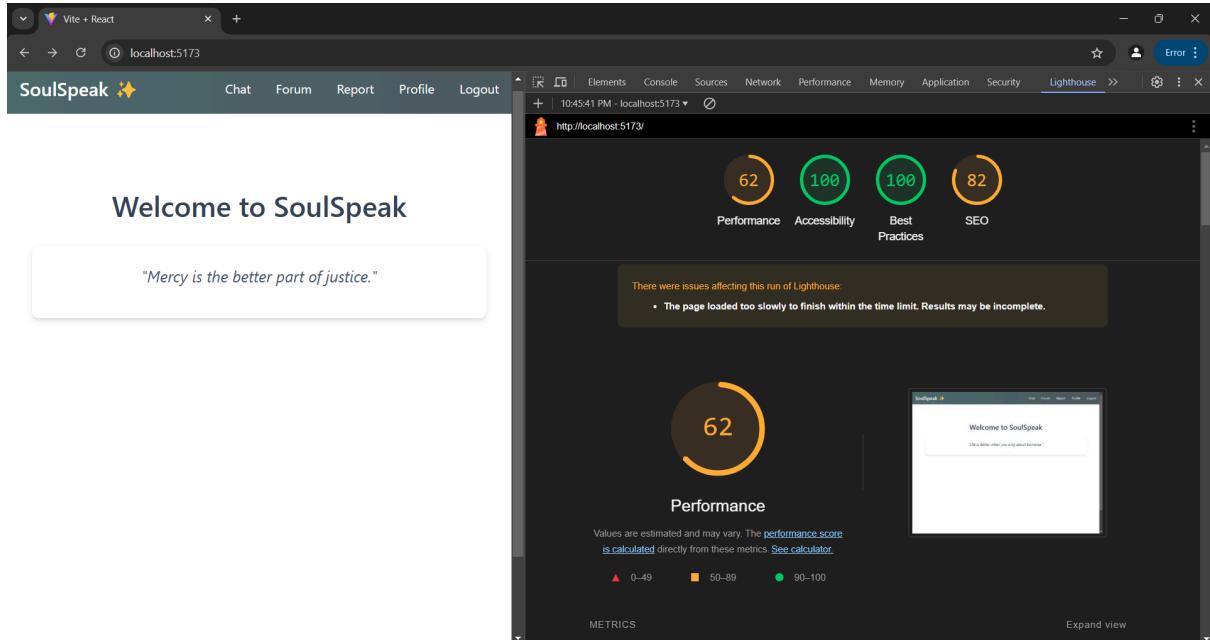
Check Status

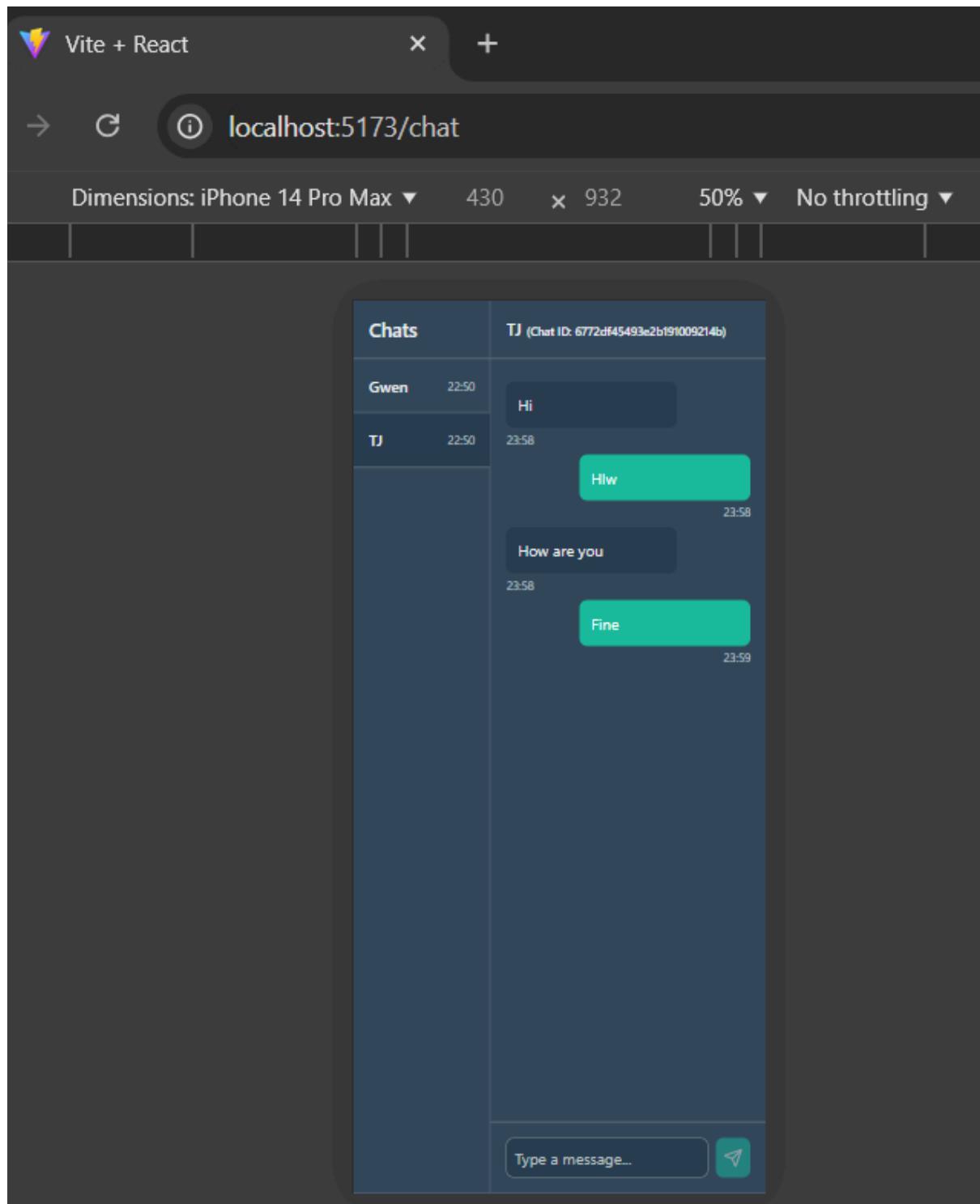
Status for :
Active

Suspend User

5. Performance and Network Analysis

Generate Performance and Network analysis report using lighthouse DevTool. Add screenshots of the report. Also, attach a screenshot of your system in any mobile viewport. **The system UI must be responsive.**





6. Github Repo [Public] Link

<https://github.com/Turja1111/CSE471---Group-2.git>

7. Link of Deployed Project

8. Individual Contribution

Group member - 01	
Name: Saumik Das Turja	Student ID: 22101090
Functional Requirements which are developed by this member:	
1. Member Profile Creation: Users provide details such as gender preference, problem areas, language, and age range.	
2. Email Verification: Users can verify their email address after account creation.	
3. Email Change Verification: Verification is required for updating the email address.	
4. Password Reset via Email: Users can reset their password securely via email.	
5. Advice API Integration: Users receive a unique piece of advice from an external API upon login.	
6. Chat Interaction Reporting: Users can report inappropriate or concerning chats via chatID.	
7. Post Editing: Users can edit their own posts.	
8. Post Deletion: Users can delete their own posts.	
9. Anonymous Posting: Users can post anonymously to protect their privacy.	
10. Profile Suspension: Admins can ban or suspend users for policy violations.	

Group member - 02	
Name: Shaiera Sultana Oishe	Student ID: 21201339
Functional Requirements which are developed by this member:	
1. Profile Editing: Users can edit their profile information (e.g., name, email, password, preferences).	
2. Profile Picture Update: Users can change their profile picture.	
3. Profile Reporting: Users can report profiles with reasons for evaluation by admins.	
4. Post Creation: Users can create posts in specific fields.	
5. Post Engagement: Users can comment on and upvote posts to interact with others.	
6. Report Management: Admins can access and review reported issues.	

Group member - 03

Name: Mohammed Raqin Rahman	Student ID: 21301275
Functional Requirements which are developed by this member:	
<ol style="list-style-type: none"> 1. Companion Training Program: Interactive modules with articles, resources, and text-based modules for training. 2. Question Model: Admin can access the question model and can update questions. 3. Chat Request Initiation: Users can send chat requests to 3–4 suggested companions based on preferences and availability. 4. Report Feedback: Users receive updates on actions taken regarding their reports. 5. Feedback Review: Admins can view user feedback to improve services. 6. Feedback Analysis: Admins use MongoDB Atlas to analyze and sort user feedback. 7. Report Notifications: Users receive email notifications about report issues. 8. Themed Forums: Forums organized into categories like "Mental Health," "Relationships," etc. 	

Group member - 04	
Name: Sibgatullah Tasnim	Student ID: 22101428
Functional Requirements which are developed by this member:	
<ol style="list-style-type: none"> 1. Companion Qualification Test: 20 scenario-based MCQ tests to qualify as a companion. 2. Retest Option: Users can retake tests to qualify as companions. 3. Chat interaction: Users can chat with other companions. 4. Resource Sharing: Users can share helpful articles, book recommendations, and mental health resources. 5. Resource Recommendations Dashboard: A curated dashboard of recommended resources for users. 6. User Role Monitoring: Admins can view and manage user roles and types. 7. Download Report PDF: Admins can download reports as PDFs. 	