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

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

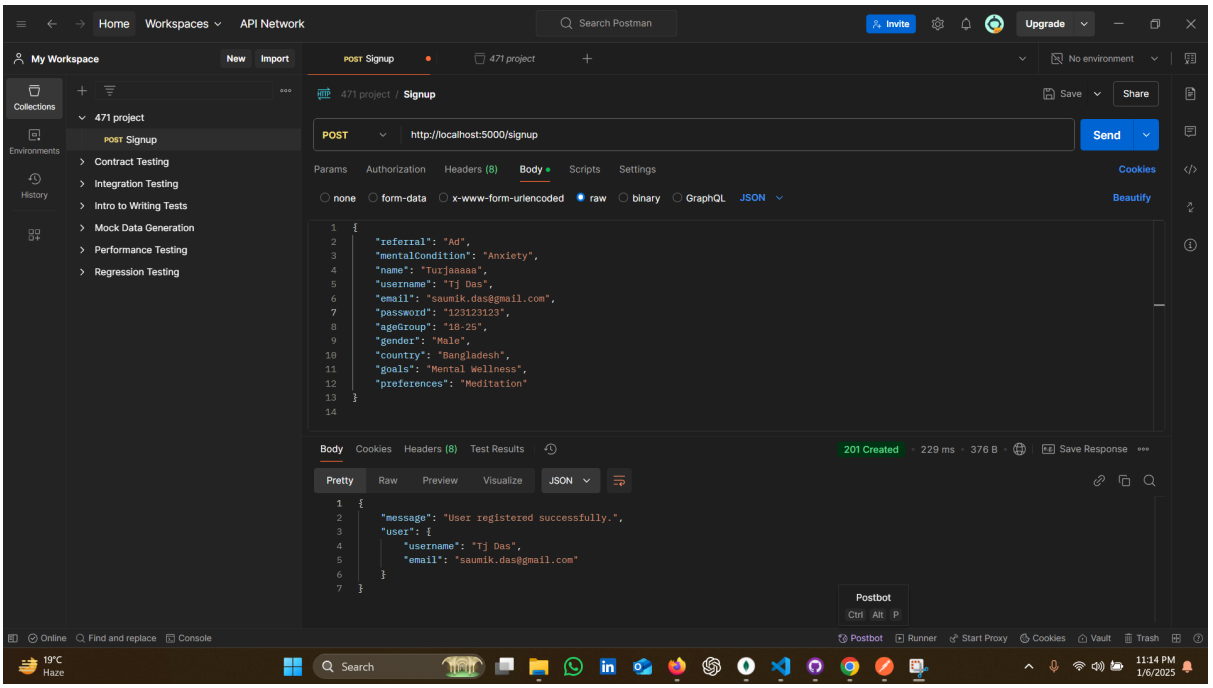
Backend: Node.js, Express.js

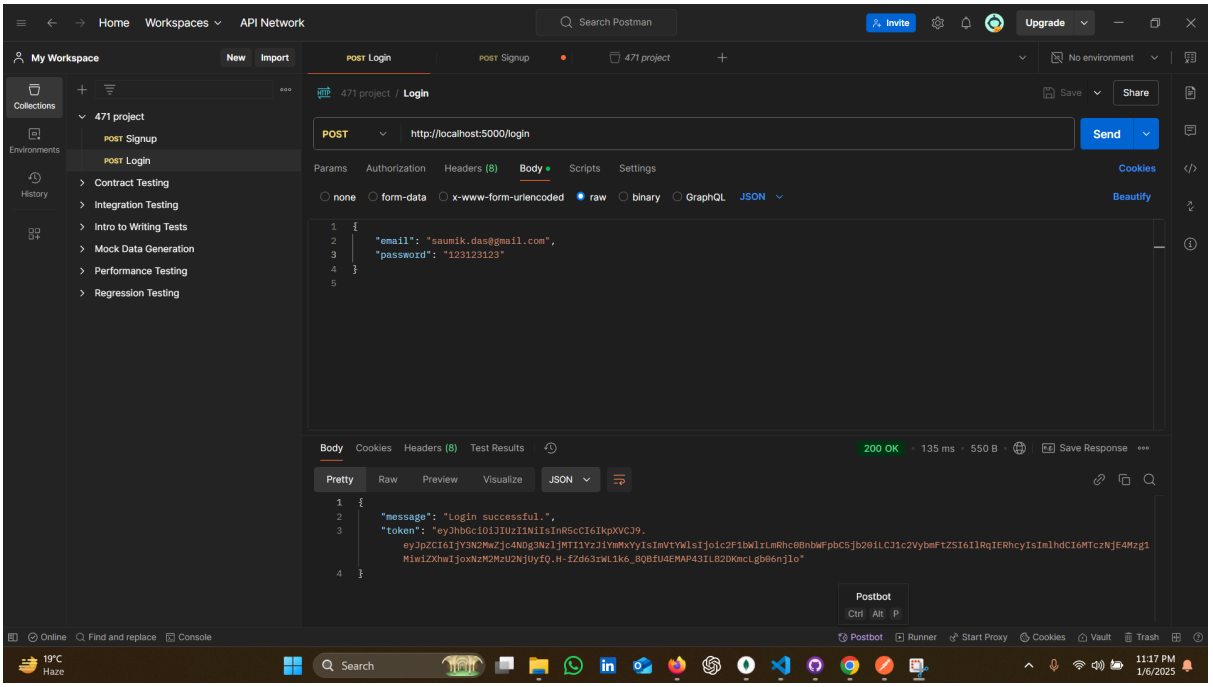
Database: MongoDB

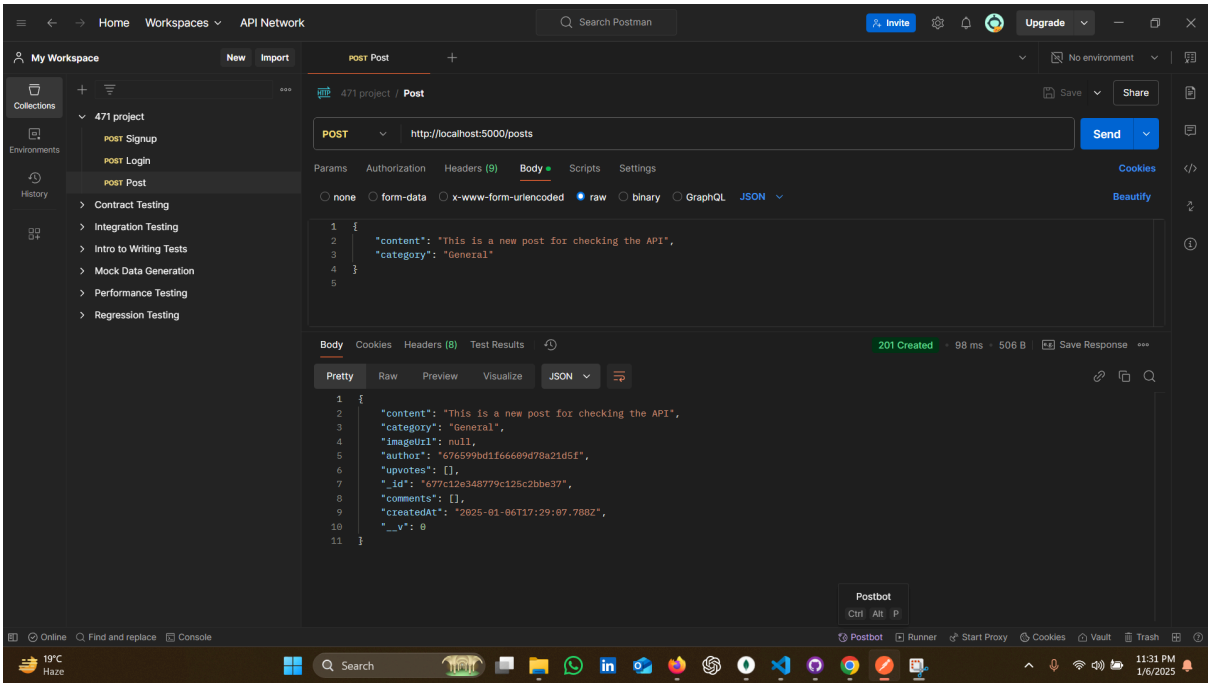
APIs: External Advice API for dynamic content

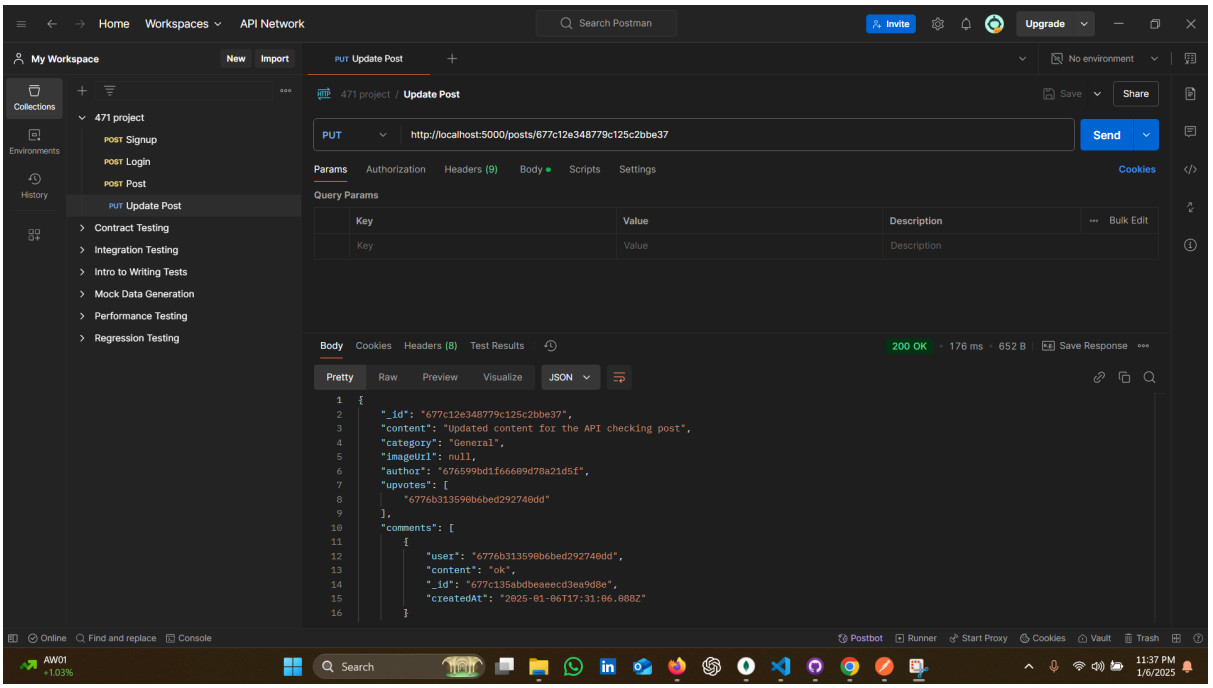
1. Backend Development

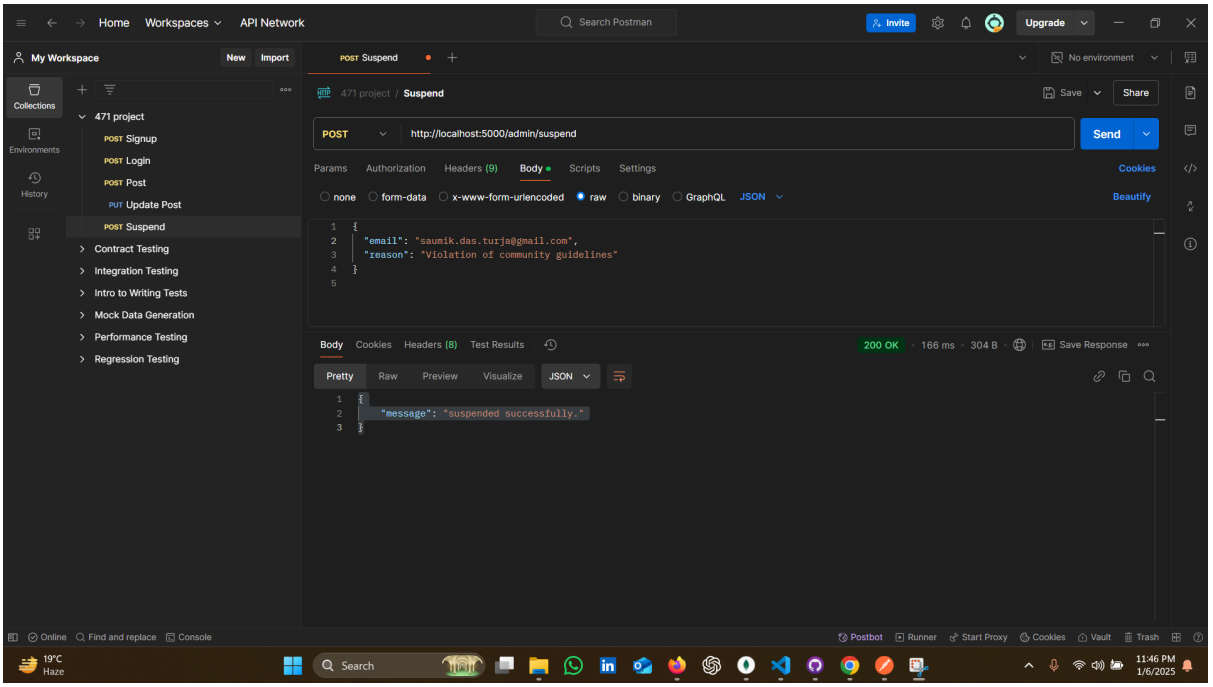
Attach at least 5 code snippets of Backend/API development with proper description. Also, attach corresponding screenshots of Postman API testing.



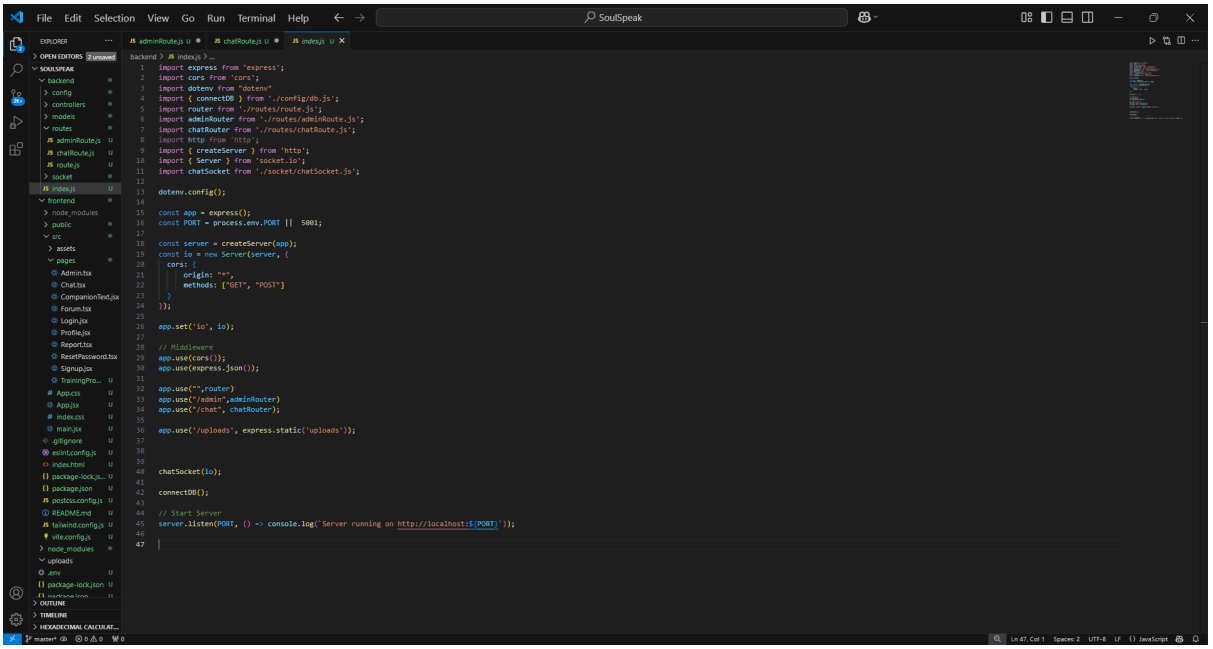



Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
POST Post
No environment
Collections
Environments
History
471 project / Post
Save
Share
POST
http://localhost:5000/posts
Send
Params
Authorization
Headers (9)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"content": "This is a new post for checking the API",
"category": "General"
}
Body
Cookies
Headers (8)
Test Results
201 Created
98 ms
506 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"content": "This is a new post for checking the API",
"category": "General",
"imageUrl": null,
"author": "676599bd1f66609d78a21d5f",
"upvotes": [],
"_id": "677c12e348779c125c2bbe37",
"comments": [],
"createdAt": "2025-01-06T17:29:07.788Z",
"__v": 0
}
471 project
POST Signup
POST Login
POST Post
Contract Testing
Integration Testing
Intro to Writing Tests
Mock Data Generation
Performance Testing
Regression Testing
Postbot
Ctrl Alt P
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash
19°C
Haze
Search
11:31 PM
1/6/2025

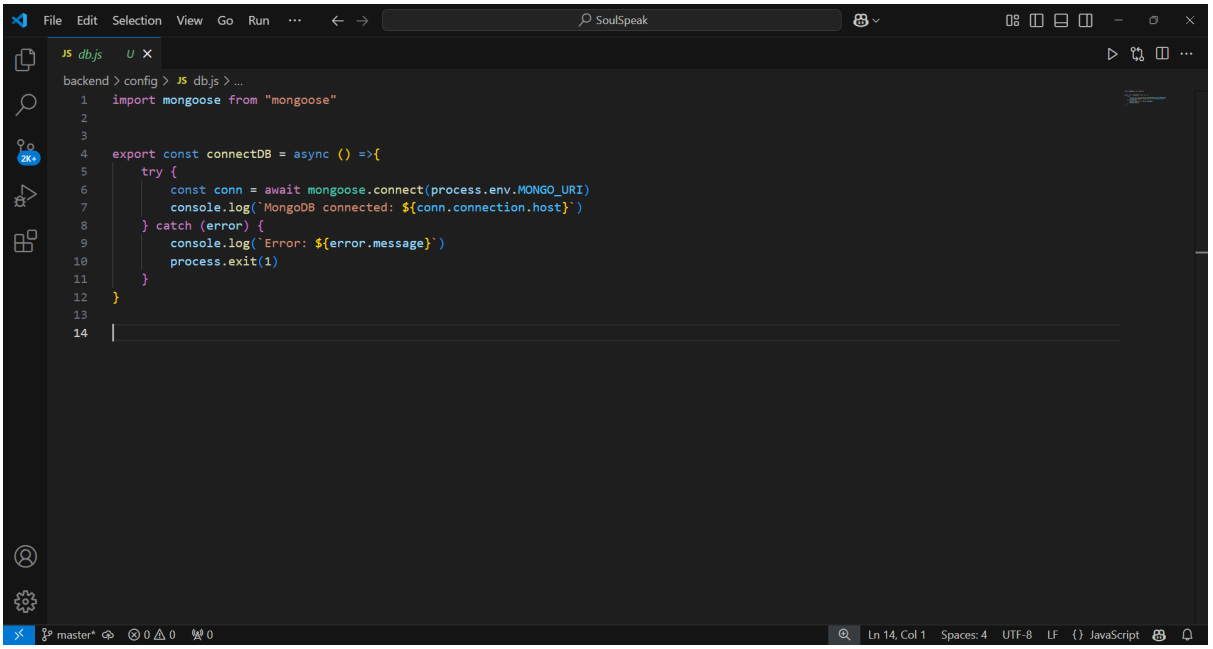

Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
PUT Update Post
No environment
Collections
Environments
History
471 project / Update Post
Save
Share
PUT
http://localhost:5000/posts/677c12e348779c125c2bbe37
Send
Params
Authorization
Headers (9)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (8)
Test Results
200 OK
176 ms
652 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"_id": "677c12e348779c125c2bbe37",
"content": "Updated content for the API checking post",
"category": "General",
"imageUrl": null,
"author": "676599bd1f66609d78a21d5f",
"upvotes": [
"6776b313590b6bed292740dd"
],
"comments": [
{
"user": "6776b313590b6bed292740dd",
"content": "ok",
"_id": "677c135abdbeaeecd3ea9d8e",
"createdAt": "2025-01-06T17:31:06.008Z"
}
471 project
POST Signup
POST Login
POST Post
PUT Update Post
Contract Testing
Integration Testing
Intro to Writing Tests
Mock Data Generation
Performance Testing
Regression Testing
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash
AW01
+1.03%
Search
11:37 PM
1/6/2025



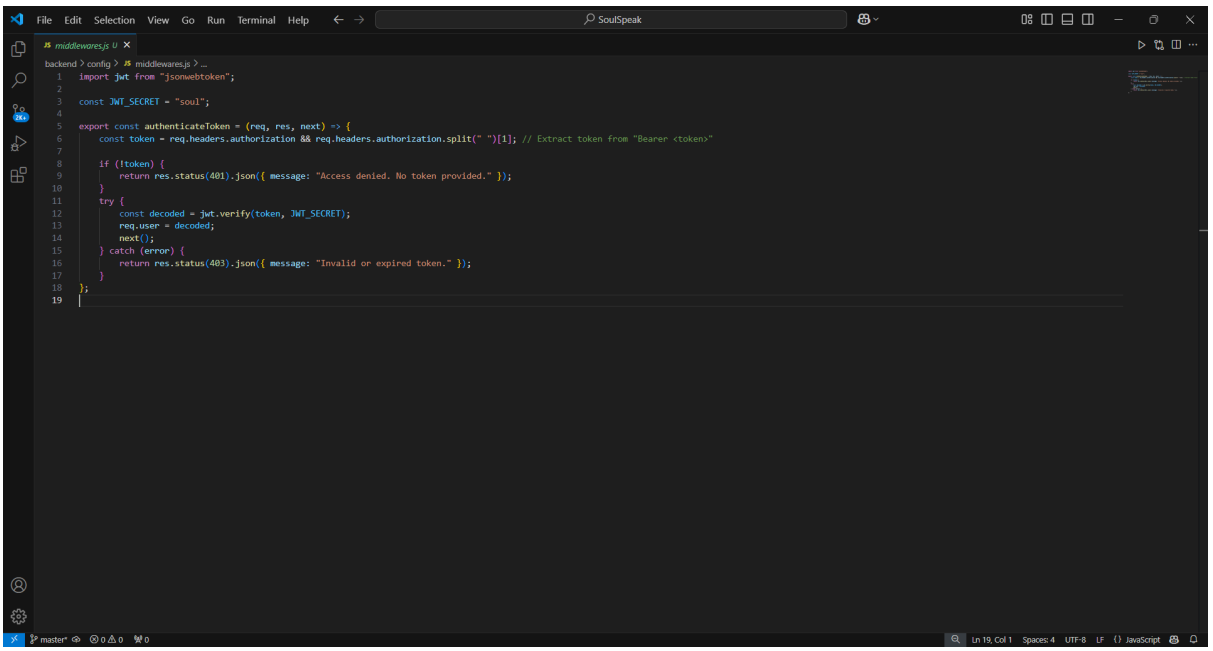
index.js



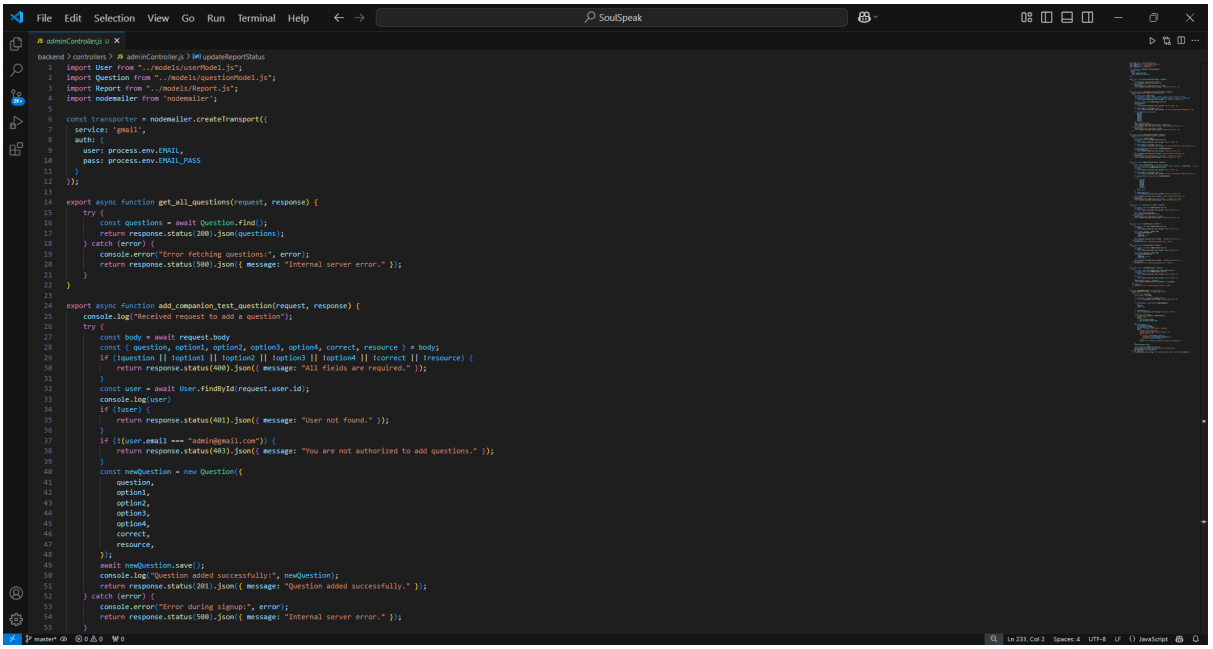
db.js



middlewares.js

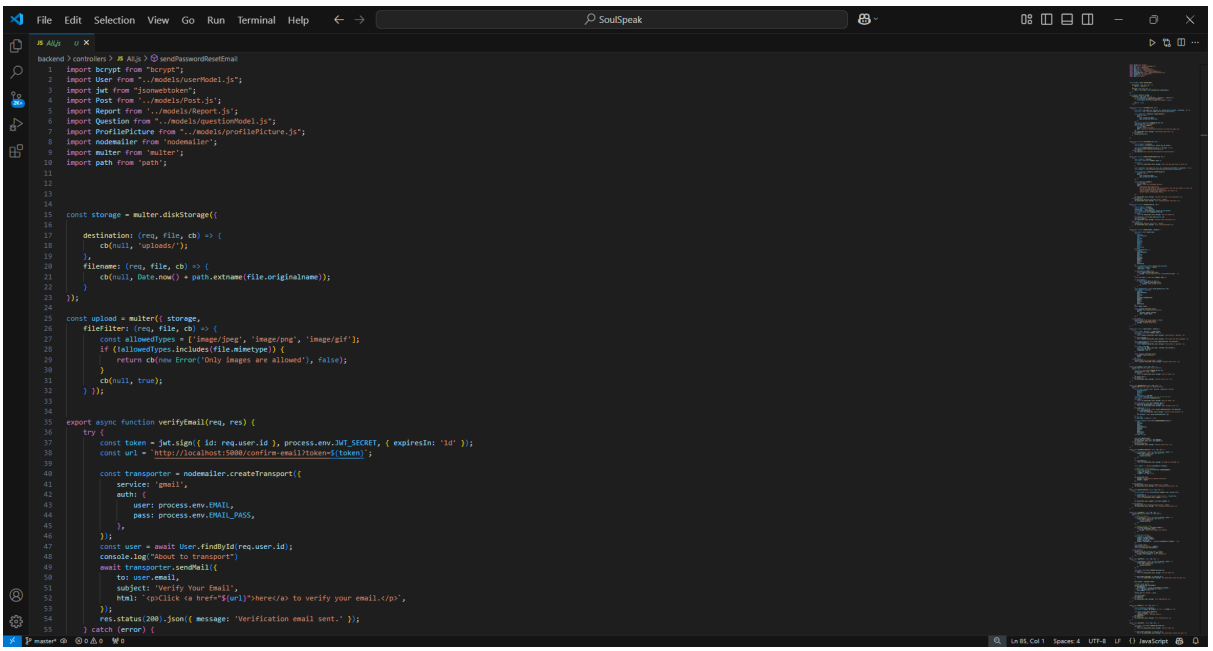


adminController.js



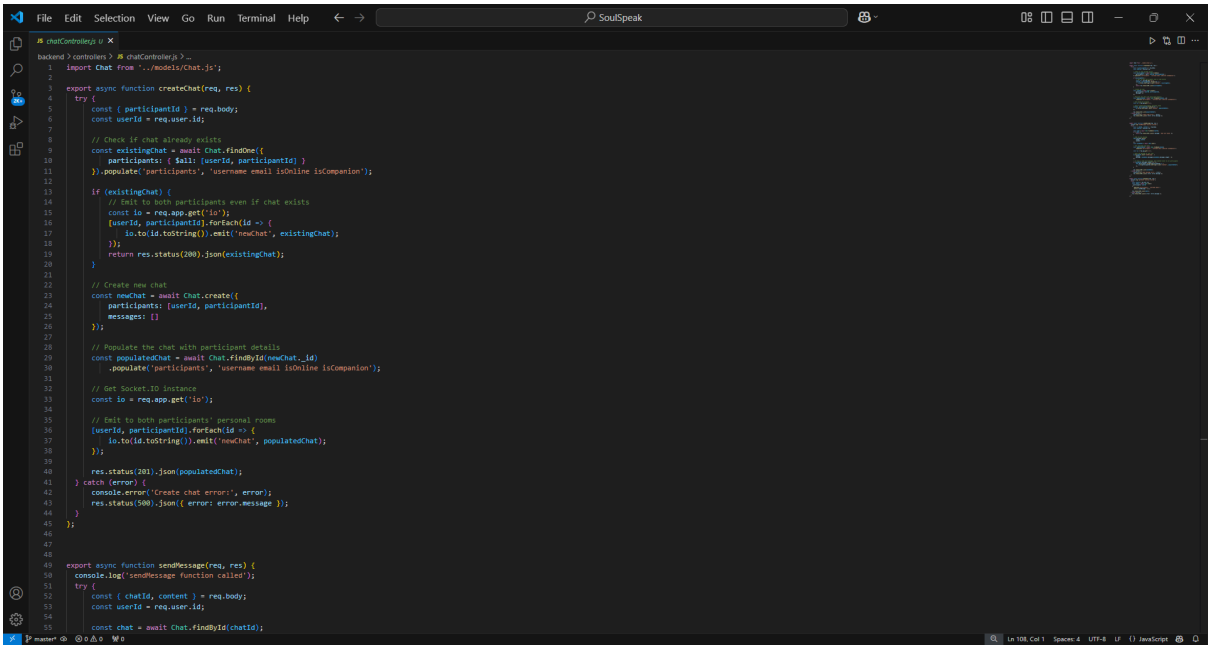
```
import User from "../models/userModel.js";
import Question from "../models/questionModel.js";
import Report from "../models/Report.js";
import nodemailer from 'nodemailer';

const transporter = nodemailer.createTransport({
  service: 'gmail',
  auth: {
    user: process.env.EMAIL,
    pass: process.env.EMAIL_PASS
  }
});

export async function get_all_questions(request, response) {
    try {
        const questions = await Question.find();
        return response.status(200).json(questions);
    } catch (error) {
        console.error("Error fetching questions:", error);
        return response.status(500).json({ message: "Internal server error." });
    }
}

export async function add_companion_test_question(request, response) {
    console.log("Received request to add a question");
    try {
        const body = await request.body
        const { question, option1, option2, option3, option4, correct, resource } = body;
        if (!question || !option1 || !option2 || !option3 || !option4 || !correct || !resource) {
            return response.status(400).json({ message: "All fields are required." });
        }
        const user = await User.findById(request.user.id);
        console.log(user)
        if (!user) {
            return response.status(401).json({ message: "User not found." });
        }
        if (!(user.email === "admin@gmail.com")) {
            return response.status(403).json({ message: "You are not authorized to add questions." });
        }
        const newQuestion = new Question({
            question,
            option1,
            option2,
            option3,
            option4,
            correct,
            resource,
        });
        await newQuestion.save();
        console.log("Question added successfully:", newQuestion);
        return response.status(201).json({ message: "Question added successfully." });
    } catch (error) {
        console.error("Error during signup:", error);
        return response.status(500).json({ message: "Internal server error." });
    }
```

All.js



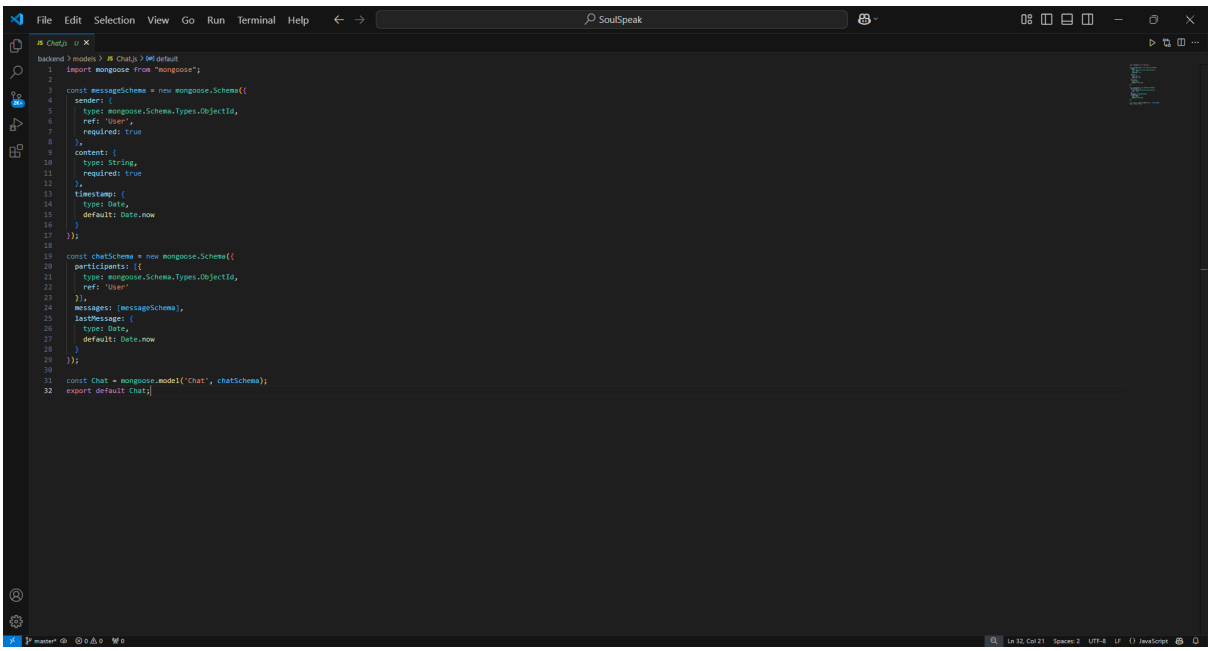
```
import bcrypt from "bcrypt";
import User from "../models/userModel.js";
import jwt from "jsonwebtoken";
import Post from '../models/Post.js';
import Report from '../models/Report.js';
import Question from "../models/questionModel.js";
import ProfilePicture from "../models/profilePicture.js";
import nodemailer from 'nodemailer';
import multer from 'multer';
import path from 'path';



const storage = multer.diskStorage({

    destination: (req, file, cb) => {
        cb(null, 'uploads/');
    },
    filename: (req, file, cb) => {
        cb(null, Date.now() + path.extname(file.originalname));
    }
});

const upload = multer({ storage,
    fileFilter: (req, file, cb) => {
        const allowedTypes = ['image/jpeg', 'image/png', 'image/gif'];
        if (!allowedTypes.includes(file.mimetype)) {
            return cb(new Error('Only images are allowed'), false);
        }
        cb(null, true);
    } });


export async function verifyEmail(req, res) {
    try {
        const token = jwt.sign({ id: req.user.id }, process.env.JWT_SECRET, { expiresIn: '1d' });
        const url = `http://localhost:5000/confirm-email?token=${token}`;

        const transporter = nodemailer.createTransport({
            service: 'gmail',
            auth: {
                user: process.env.EMAIL,
                pass: process.env.EMAIL_PASS,
            },
        });
        const user = await User.findById(req.user.id);
        console.log("About to transport")
        await transporter.sendMail({
            to: user.email,
            subject: 'Verify Your Email',
            html: `<p>Click <a href="${url}">here</a> to verify your email.</p>`,
        });
        res.status(200).json({ message: 'Verification email sent.' });
    } catch (error) {
```

chatController.js



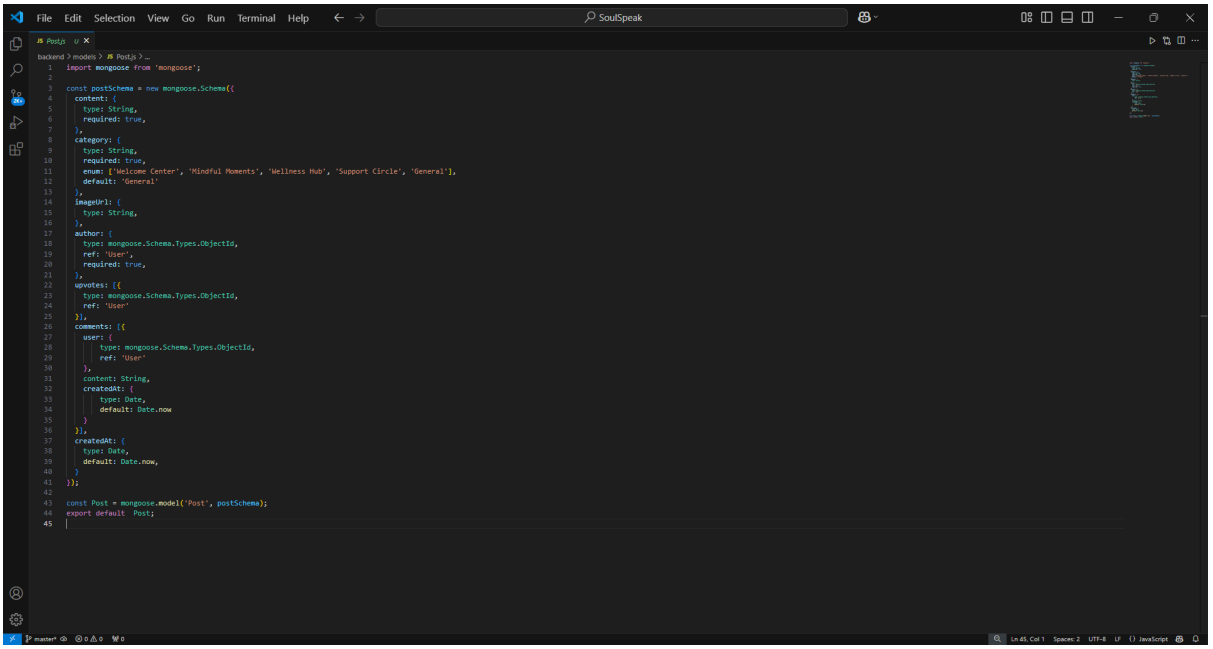
```
import Chat from '../models/Chat.js';

export async function createChat(req, res) {
  try {
    const { participantId } = req.body;
    const userId = req.user.id;

    // Check if chat already exists
    const existingChat = await Chat.findOne({
      participants: { $all: [userId, participantId] }
    }).populate('participants', 'username email isOnline isCompanion');

    if (existingChat) {
      // Emit to both participants even if chat exists
      const io = req.app.get('io');
      [userId, participantId].forEach(id => {
        io.to(id.toString()).emit('newChat', existingChat);
      });
      return res.status(200).json(existingChat);
    }

    // Create new chat
    const newChat = await Chat.create({
      participants: [userId, participantId],
      messages: []
    });

    // Populate the chat with participant details
    const populatedChat = await Chat.findById(newChat._id)
      .populate('participants', 'username email isOnline isCompanion');

    // Get Socket.IO instance
    const io = req.app.get('io');

    // Emit to both participants' personal rooms
    [userId, participantId].forEach(id => {
      io.to(id.toString()).emit('newChat', populatedChat);
    });

    res.status(201).json(populatedChat);
  } catch (error) {
    console.error('Create chat error:', error);
    res.status(500).json({ error: error.message });
  }
};



export async function sendMessage(req, res) {
  console.log('sendMessage function called');
  try {
    const { chatId, content } = req.body;
    const userId = req.user.id;

    const chat = await Chat.findById(chatId);
```

Chat.js



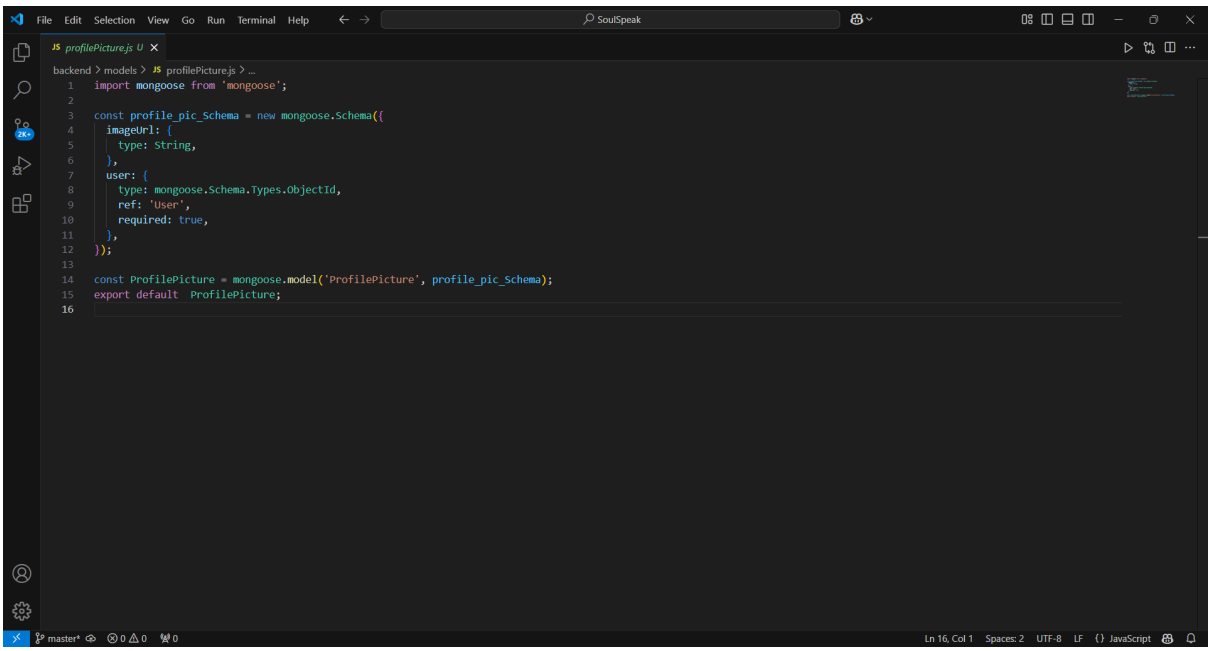
```
import mongoose from "mongoose";

const messageSchema = new mongoose.Schema({
  sender: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
    required: true
  },
  content: {
    type: String,
    required: true
  },
  timestamp: {
    type: Date,
    default: Date.now
  }
});

const chatSchema = new mongoose.Schema({
  participants: [{
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User'
  }],
  messages: [messageSchema],
  lastMessage: {
    type: Date,
    default: Date.now
  }
});

const Chat = mongoose.model('Chat', chatSchema);
export default Chat;
```

Post.js



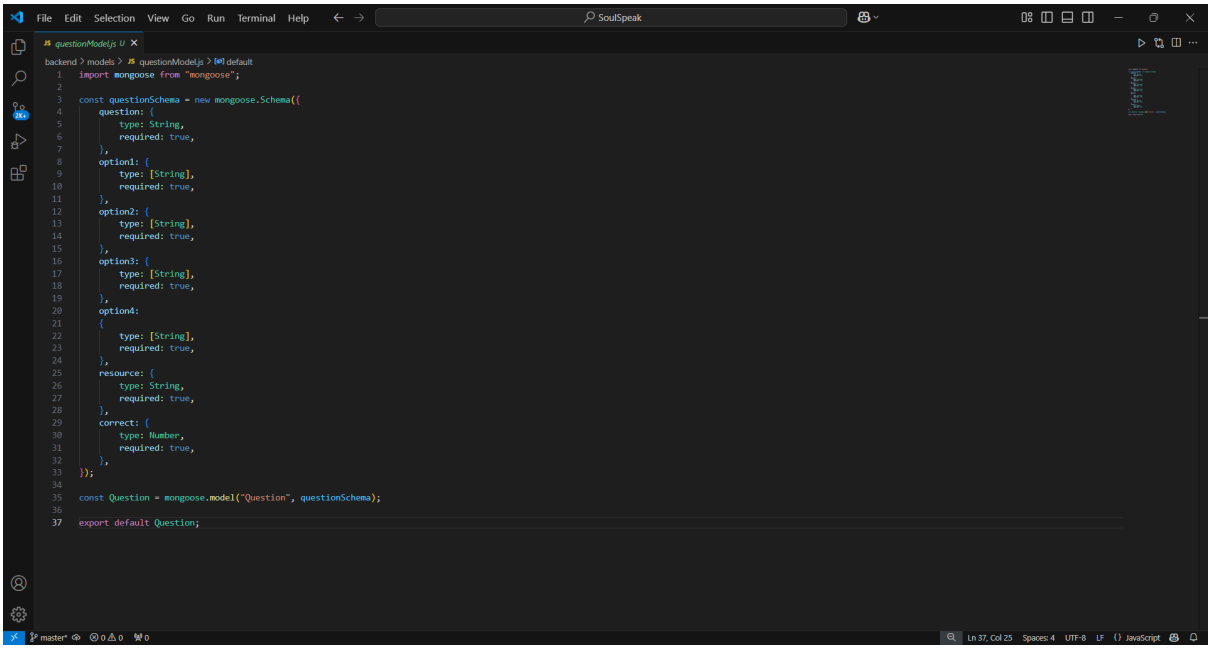
```
import mongoose from 'mongoose';

const postSchema = new mongoose.Schema({
  content: {
    type: String,
    required: true,
  },
  category: {
    type: String,
    required: true,
    enum: ['Welcome Center', 'Mindful Moments', 'Wellness Hub', 'Support Circle', 'General'],
    default: 'General'
  },
  imageUrl: {
    type: String,
  },
  author: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
    required: true,
  },
  upvotes: [{
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User'
  }],
  comments: [{
    user: {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'User'
    },
    content: String,
    createdAt: {
        type: Date,
        default: Date.now
    }
  }],
  createdAt: {
    type: Date,
    default: Date.now,
  }
});

const Post = mongoose.model('Post', postSchema);
export default  Post;
```

profilePicture.js

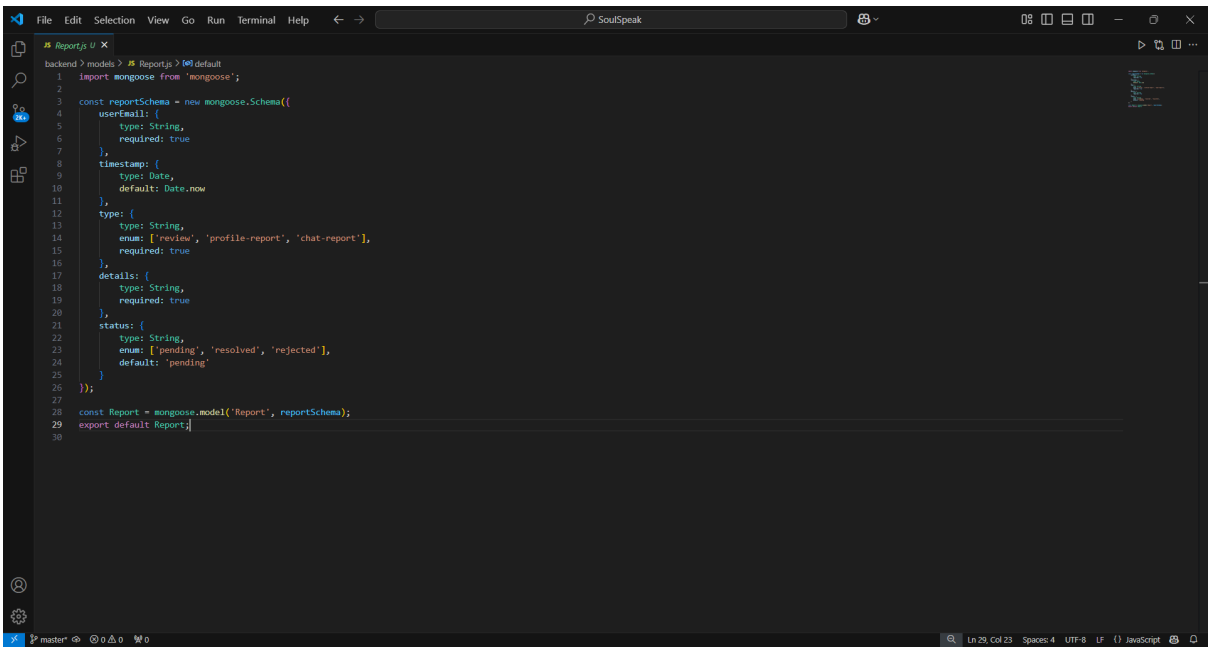


```
import mongoose from 'mongoose';

const profile_pic_Schema = new mongoose.Schema({
  imageUrl: {
    type: String,
  },
  user: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
    required: true,
  },
});

const ProfilePicture = mongoose.model('ProfilePicture', profile_pic_Schema);
export default  ProfilePicture;
```

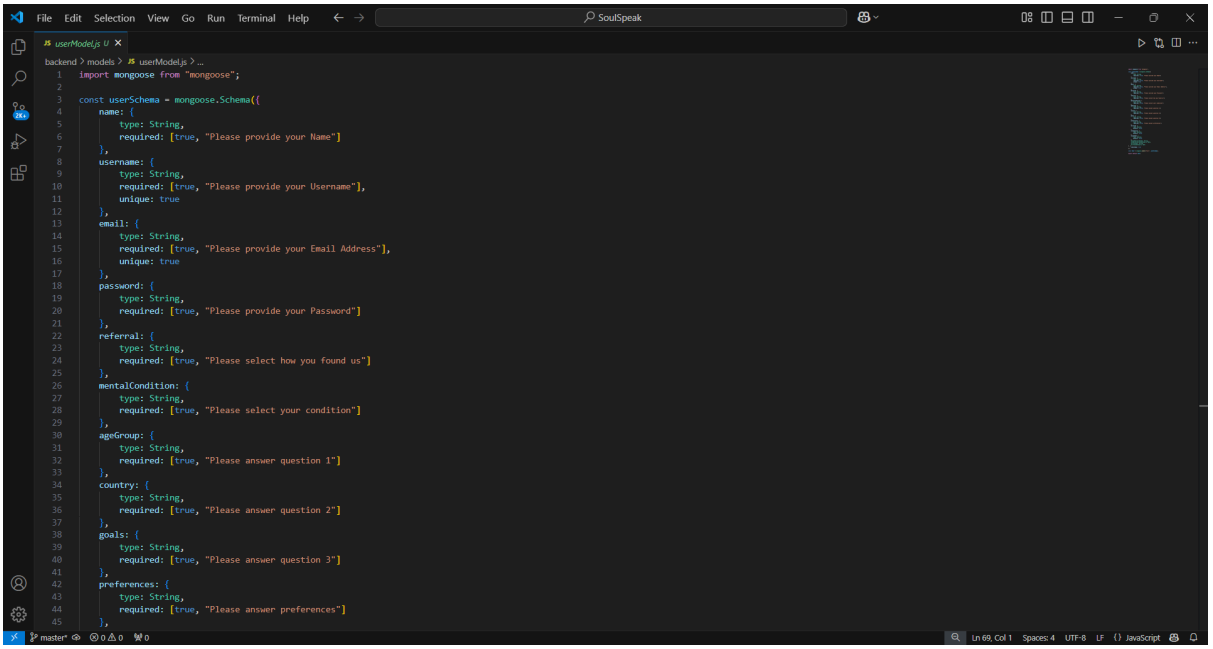
questionModel.js



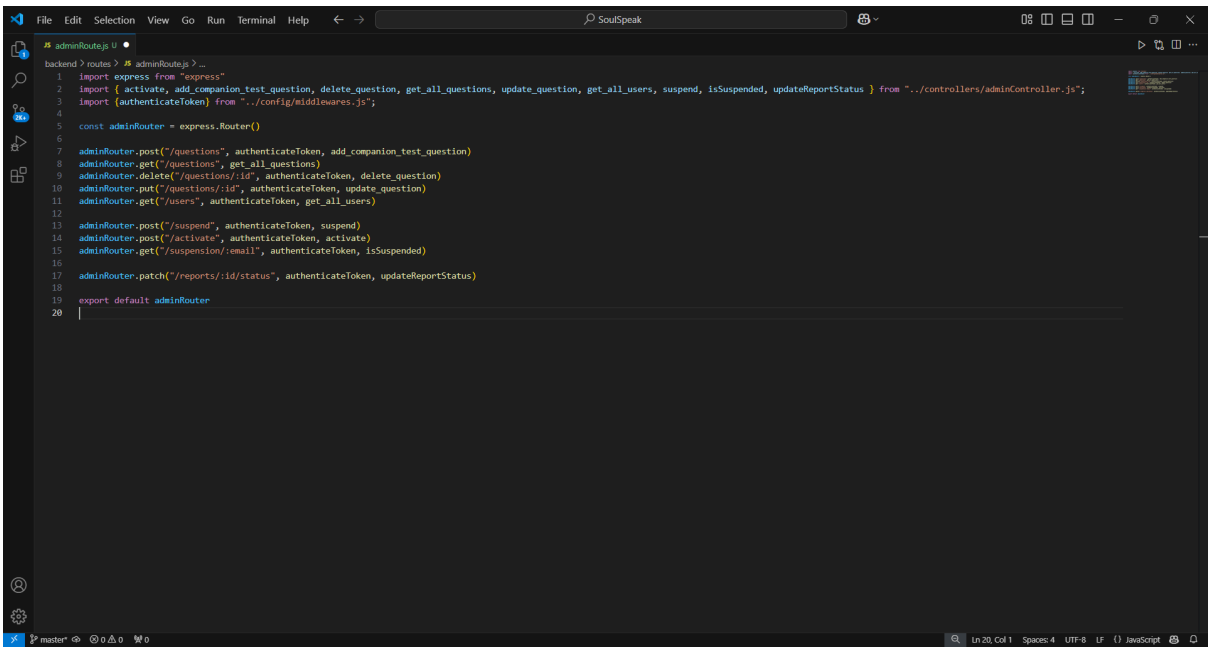
Report.js



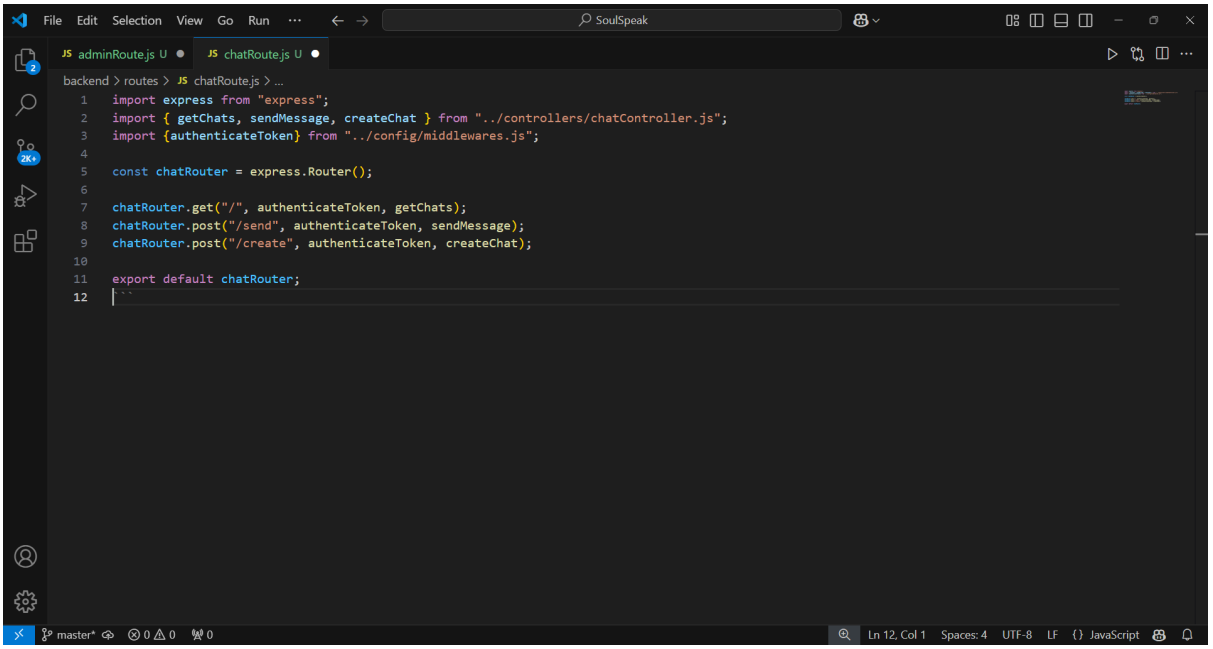
userModel.js



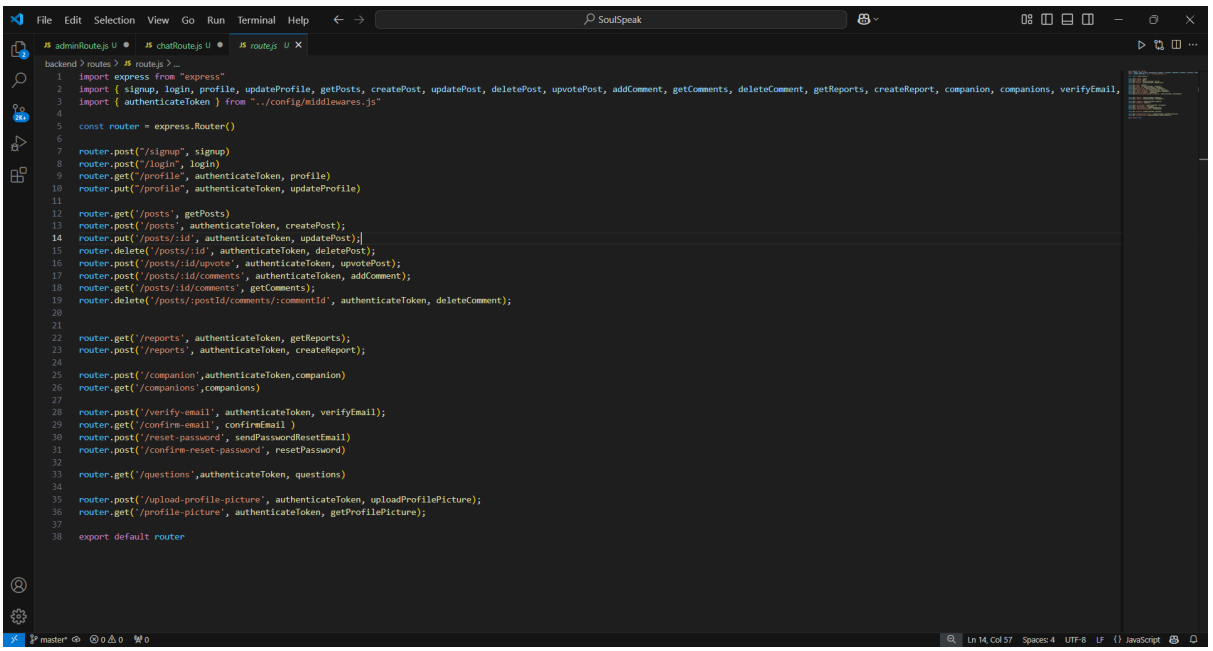
adminRoute.js



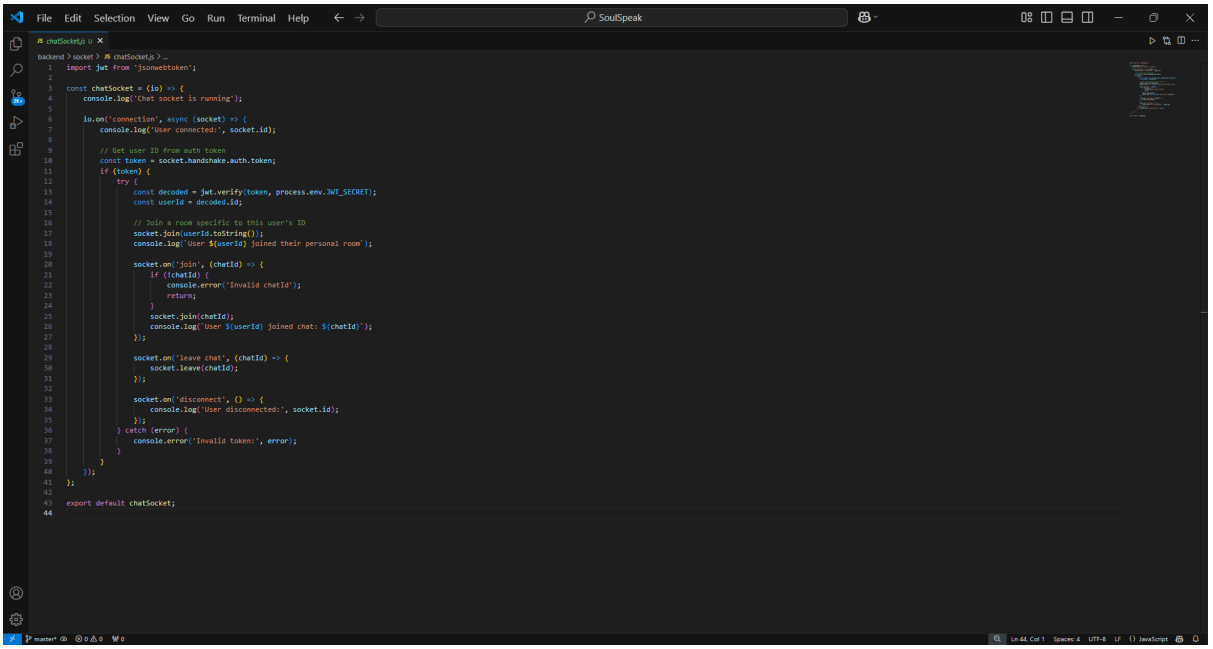
chatRoute.js



route.js

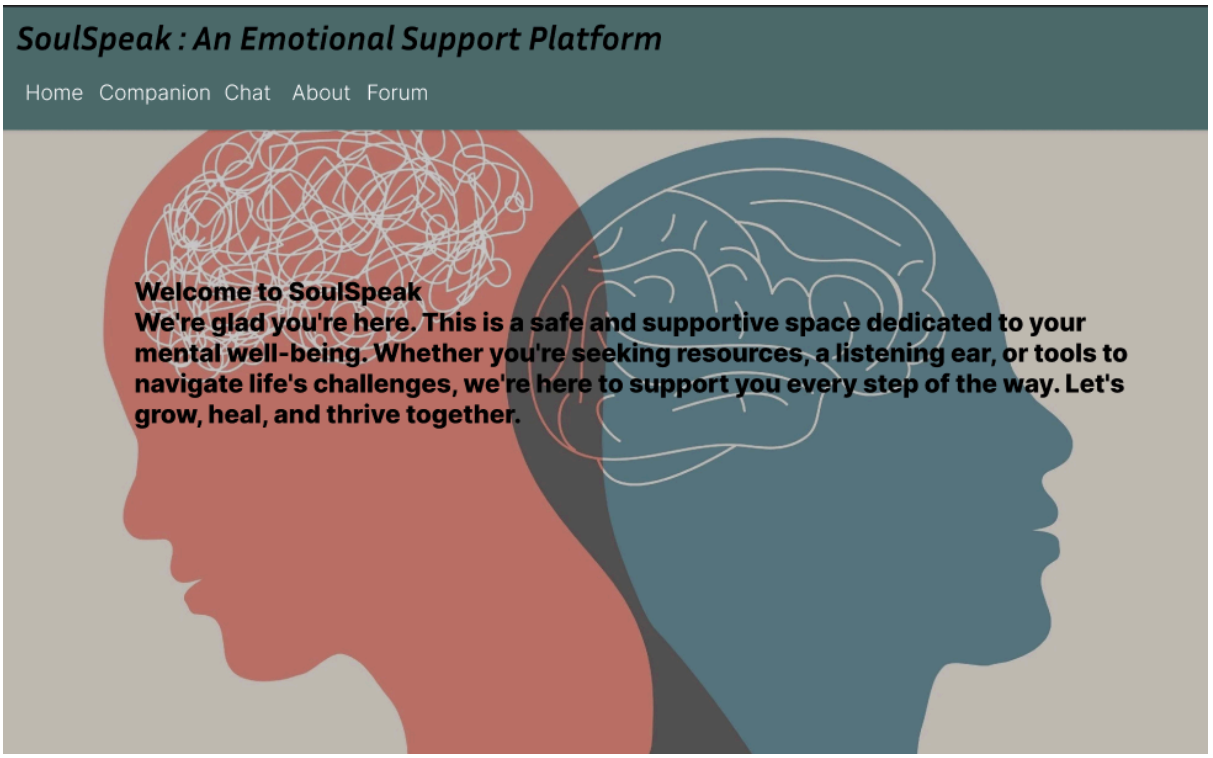


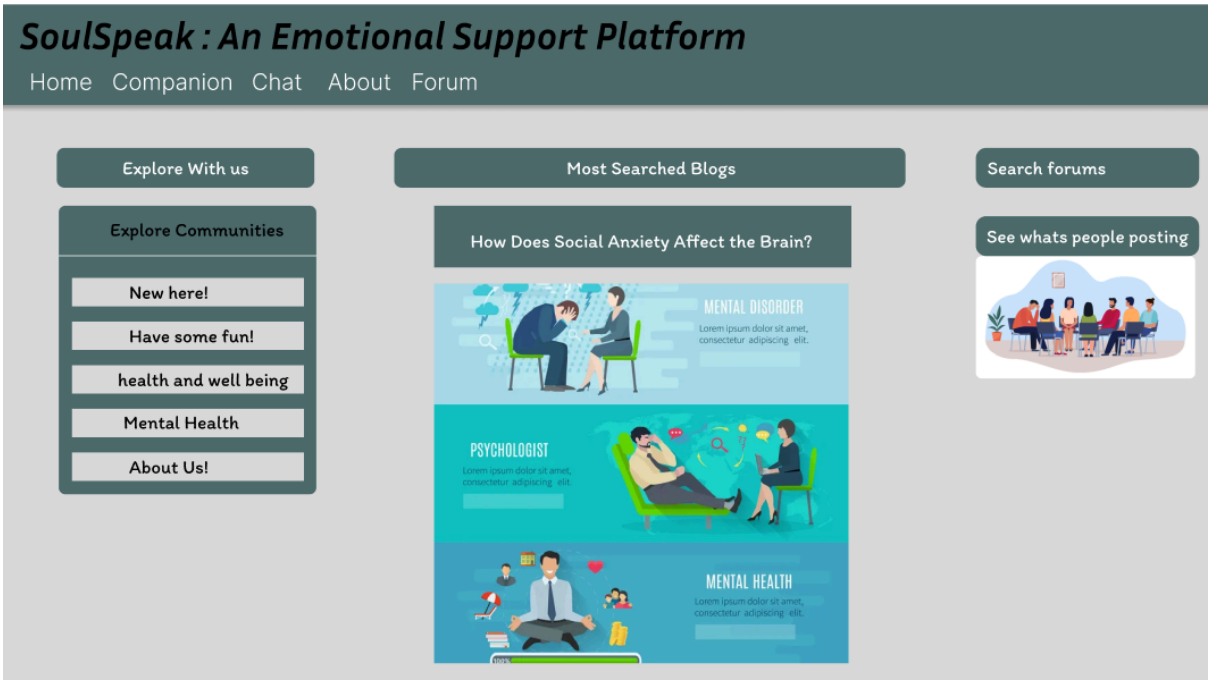
chatSocket.js

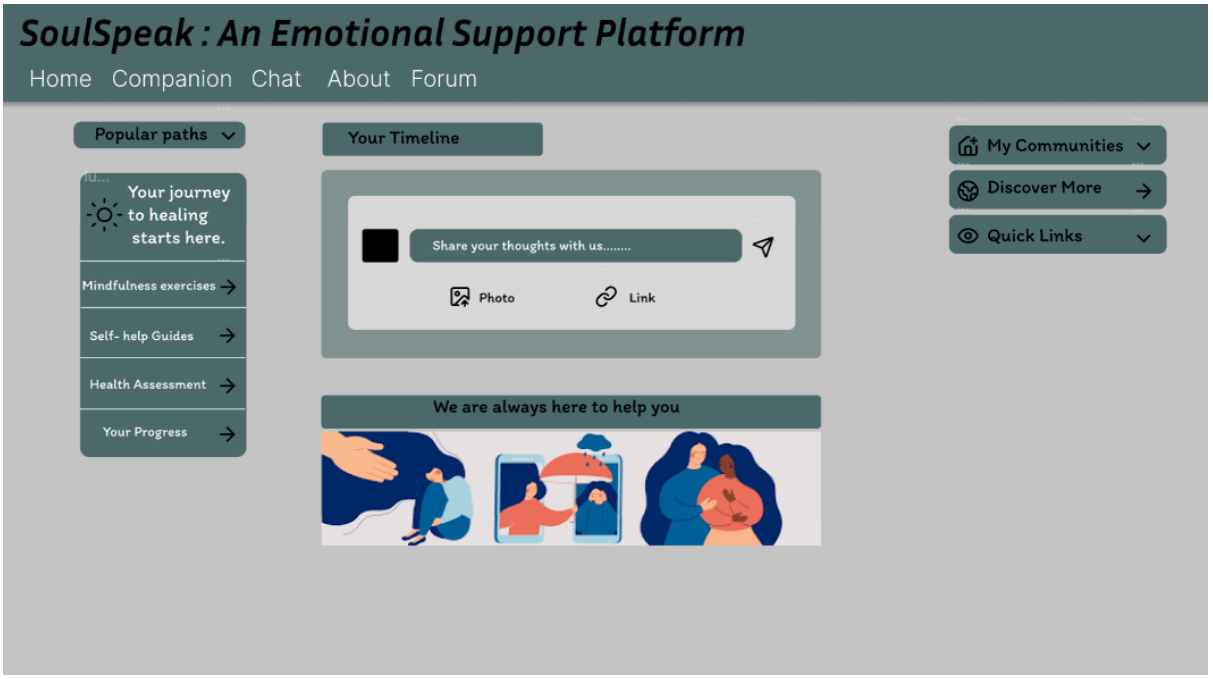

File Edit Selection View Go Run Terminal Help
SoulSpeak
chatSocket.js
backend > socket > chatSocket.js > ...
import jwt from 'jsonwebtoken';
const chatSocket = (io) => {
console.log('Chat socket is running');
io.on('connection', async (socket) => {
console.log('User connected:', socket.id);
// Get user ID from auth token
const token = socket.handshake.auth.token;
if (token) {
try {
const decoded = jwt.verify(token, process.env.JWT_SECRET);
const userId = decoded.id;
// Join a room specific to this user's ID
socket.join(userId.toString());
console.log(`User ${userId} joined their personal room`);
socket.on('join', (chatId) => {
if (!chatId) {
console.error('Invalid chatId');
return;
}
socket.join(chatId);
console.log(`User ${userId} joined chat: ${chatId}`);
});
socket.on('leave chat', (chatId) => {
socket.leave(chatId);
});
socket.on('disconnect', () => {
console.log('User disconnected:', socket.id);
});
} catch (error) {
console.error('Invalid token:', error);
}
}
});
};
export default chatSocket;
master*
Ln 44, Col 1
Spaces: 4
UTF-8
LF
JavaScript

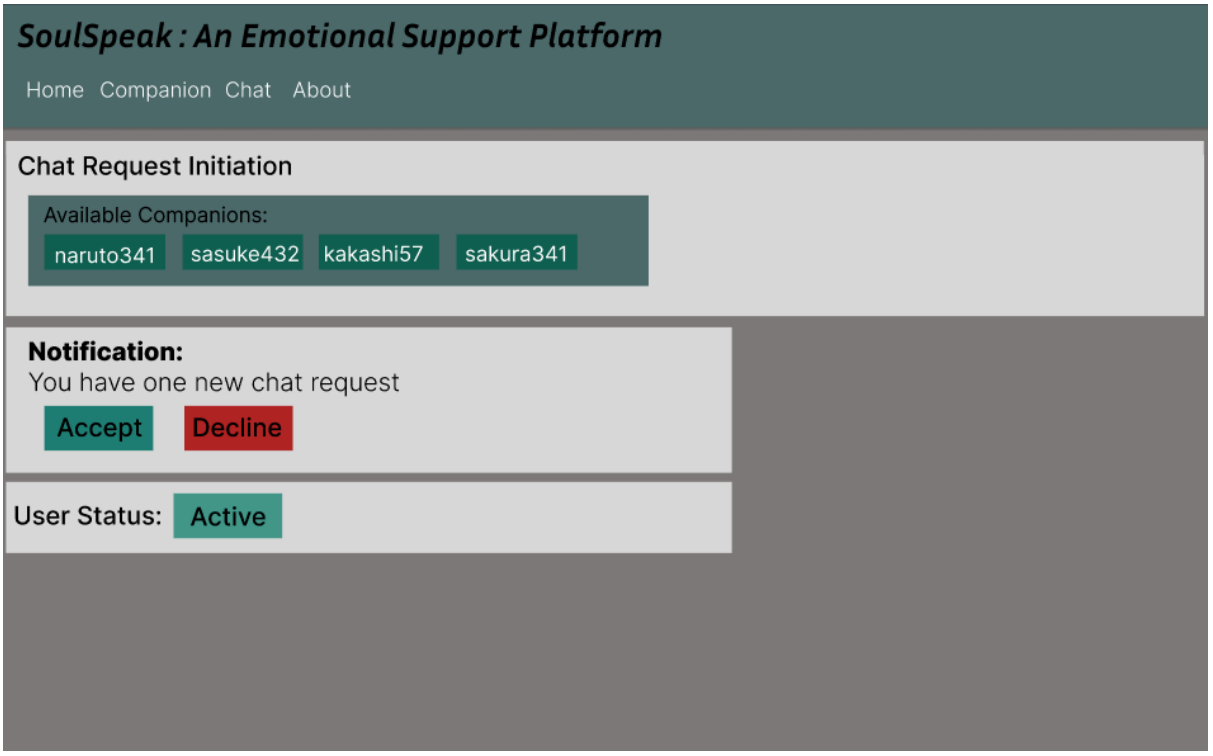
2. User Interface Design

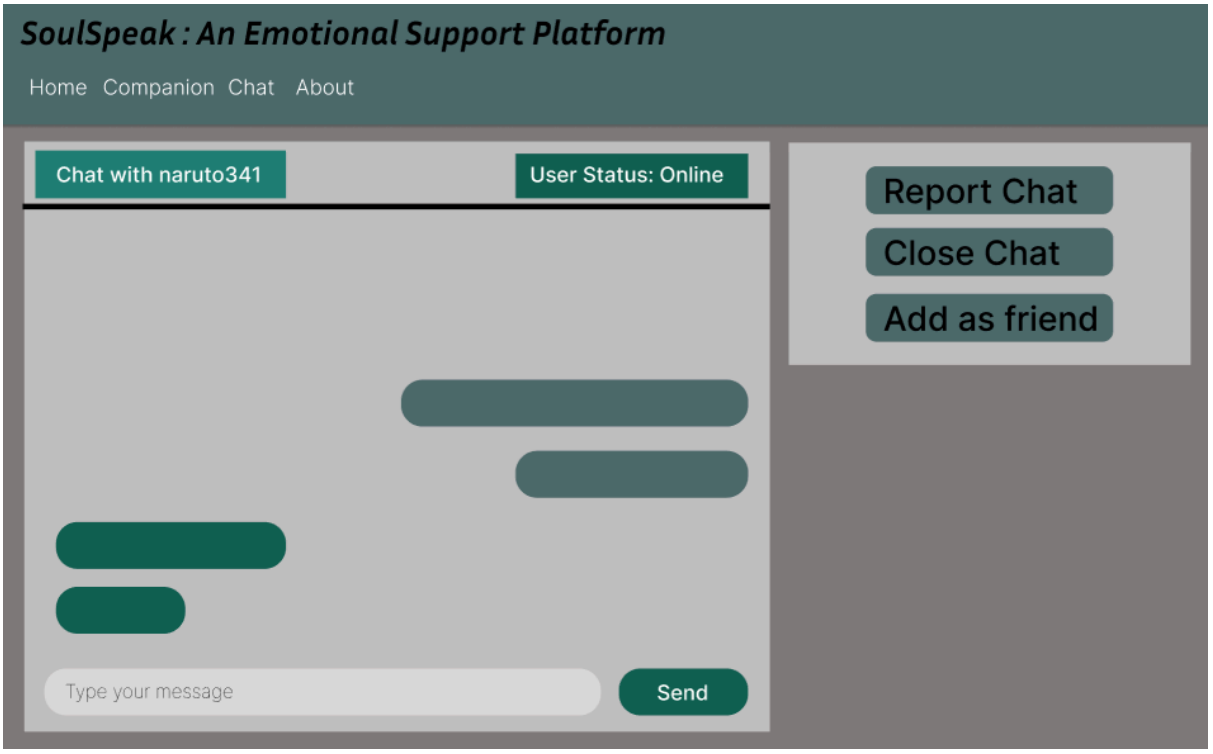
Attach at least 5 interface designs generated in Figma. Attach the figma project link.
Figma File Link:
https://www.figma.com/design/MZ5a4JW633fYjsr1RVA8AP/CSE471_Project_Figma?node-id=0-1&t=pCKcy4jucReTWE3V-1






SoulSpeak : An Emotional Support Platform
Home Companion Chat About Forum
Popular paths
Your journey to healing starts here.
Mindfulness exercises
Self- help Guides
Health Assessment
Your Progress
Your Timeline
Share your thoughts with us.....
Photo
Link
We are always here to help you
My Communities
Discover More
Quick Links

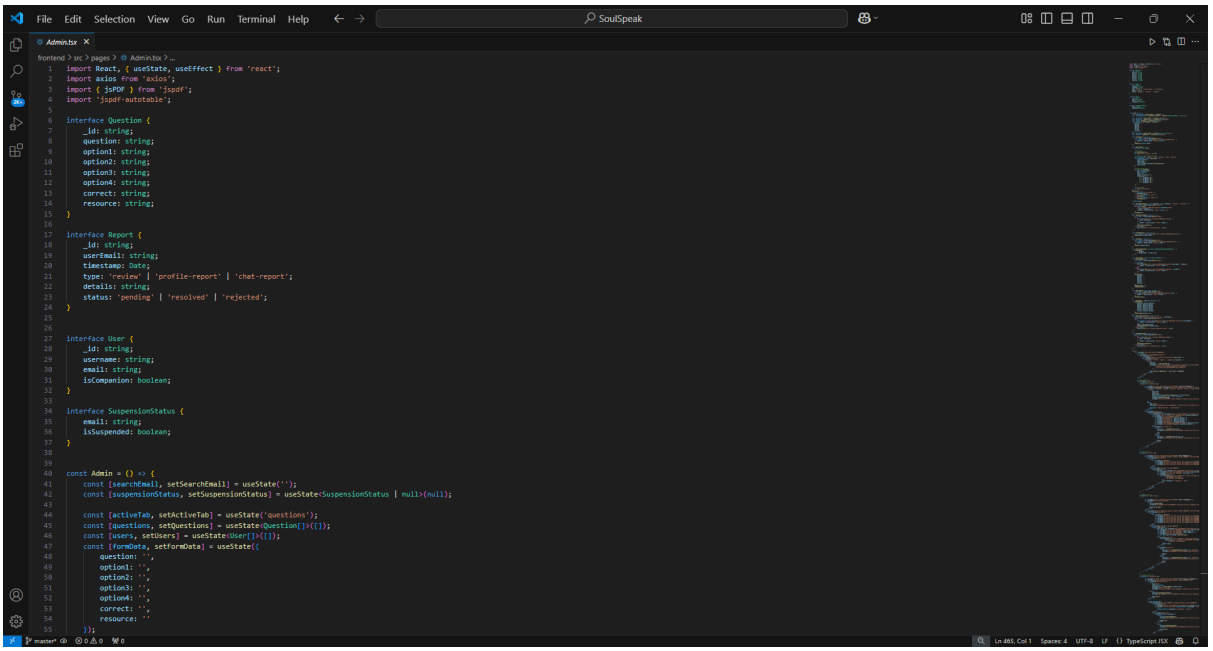

SoulSpeak : An Emotional Support Platform
Home Companion Chat About
Chat Request Initiation
Available Companions:
naruto341
sasuke432
kakashi57
sakura341
Notification:
You have one new chat request
Accept
Decline
User Status:
Active


SoulSpeak : An Emotional Support Platform
Home Companion Chat About
Chat with naruto341
User Status: Online
Report Chat
Close Chat
Add as friend
Type your message
Send

3. Frontend Development

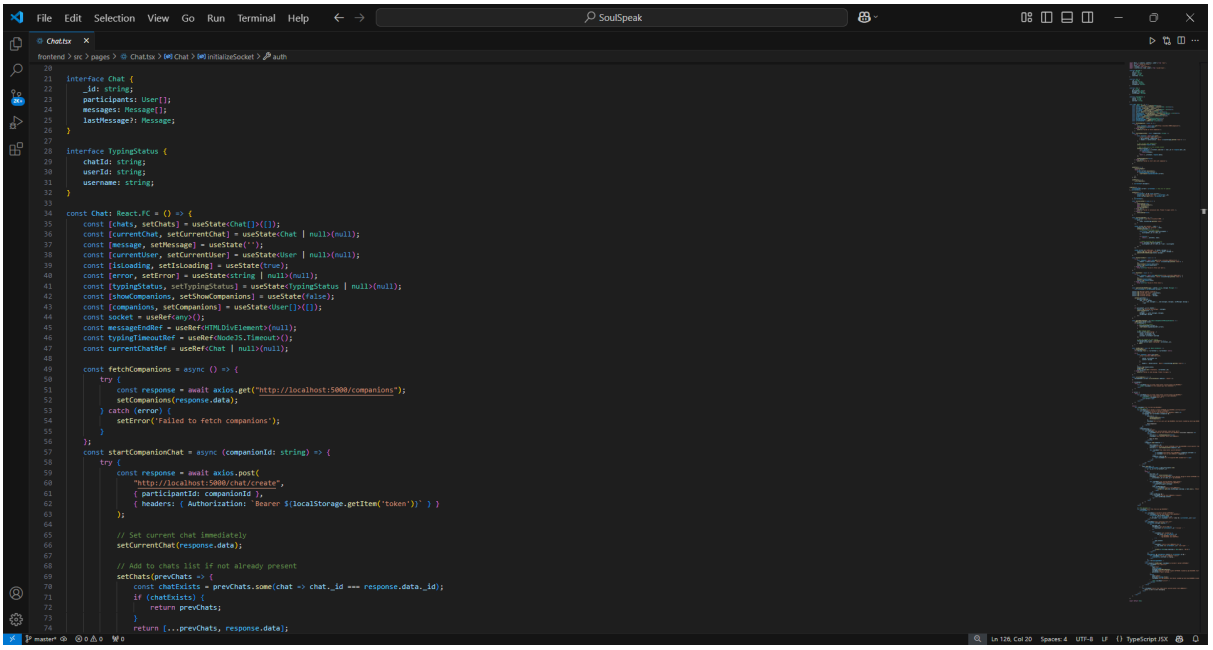
Attach at least 5 frontend code snippets with proper description.

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



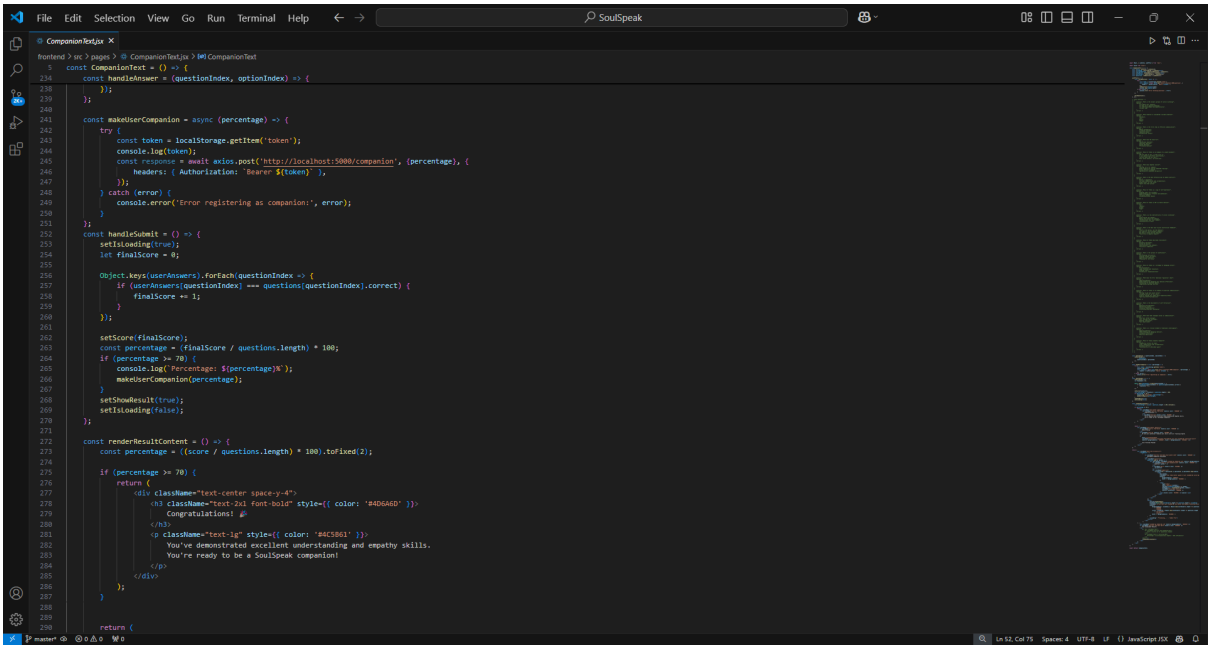
```tsx
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { jsPDF } from 'jspdf';
import 'jspdf-autotable';

interface Question {
    _id: string;
    question: string;
    option1: string;
    option2: string;
    option3: string;
    option4: string;
    correct: string;
    resource: string;
}

interface Report {
    _id: string;
    userEmail: string;
    timestamp: Date;
    type: 'review' | 'profile-report' | 'chat-report';
    details: string;
    status: 'pending' | 'resolved' | 'rejected';
}


interface User {
    _id: string;
    username: string;
    email: string;
    isCompanion: boolean;
}

interface SuspensionStatus {
    email: string;
    isSuspended: boolean;
}


const Admin = () => {
    const [searchEmail, setSearchEmail] = useState('');
    const [suspensionStatus, setSuspensionStatus] = useState<SuspensionStatus | null>(null);

    const [activeTab, setActiveTab] = useState('questions');
    const [questions, setQuestions] = useState<Question[]>([]);
    const [users, setUsers] = useState<User[]>([]);
    const [formData, setFormData] = useState({
        question: '',
        option1: '',
        option2: '',
        option3: '',
        option4: '',
        correct: '',
        resource: ''
    });
```

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

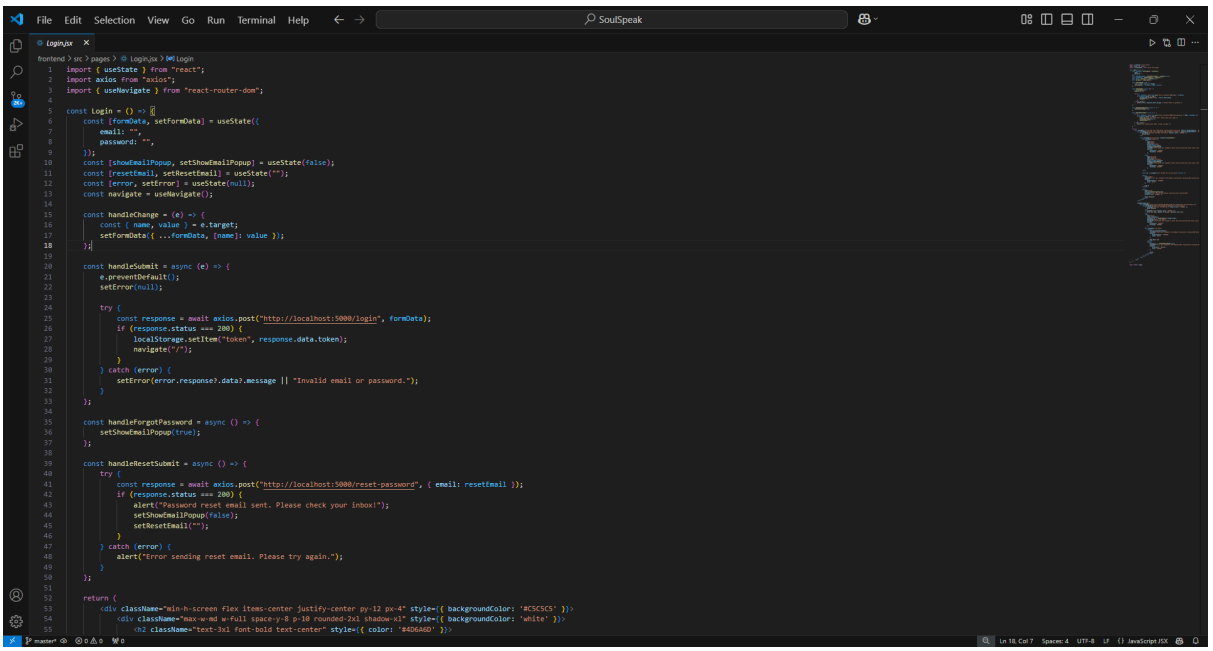


```tsx
interface Chat {
    _id: string;
    participants: User[];
    messages: Message[];
    lastMessage?: Message;
}

interface TypingStatus {
    chatId: string;
    userId: string;
    username: string;
}

const Chat: React.FC = () => {
    const [chats, setChats] = useState<Chat[]>([]);
    const [currentChat, setCurrentChat] = useState<Chat | null>(null);
    const [message, setMessage] = useState('');
    const [currentUser, setCurrentUser] = useState<User | null>(null);
    const [isLoading, setIsLoading] = useState(true);
    const [error, setError] = useState<string | null>(null);
    const [typingStatus, setTypingStatus] = useState<TypingStatus | null>(null);
    const [showCompanions, setShowCompanions] = useState(false);
    const [companions, setCompanions] = useState<User[]>([]);
    const socket = useRef<any>();
    const messageEndRef = useRef<HTMLDivElement>(null);
    const typingTimeoutRef = useRef<NodeJS.Timeout>();
    const currentChatRef = useRef<Chat | null>(null);

    const fetchCompanions = async () => {
        try {
            const response = await axios.get("http://localhost:5000/companions");
            setCompanions(response.data);
        } catch (error) {
            setError('Failed to fetch companions');
        }
    };
    const startCompanionChat = async (companionId: string) => {
        try {
            const response = await axios.post(
                "http://localhost:5000/chat/create",
                { participantId: companionId },
                { headers: { Authorization: `Bearer ${localStorage.getItem('token')}` } }
            );

            // Set current chat immediately
            setCurrentChat(response.data);

            // Add to chats list if not already present
            setChats(prevChats => {
                const chatExists = prevChats.some(chat => chat._id === response.data._id);
                if (chatExists) {
                    return prevChats;
                }
                return [...prevChats, response.data];
```

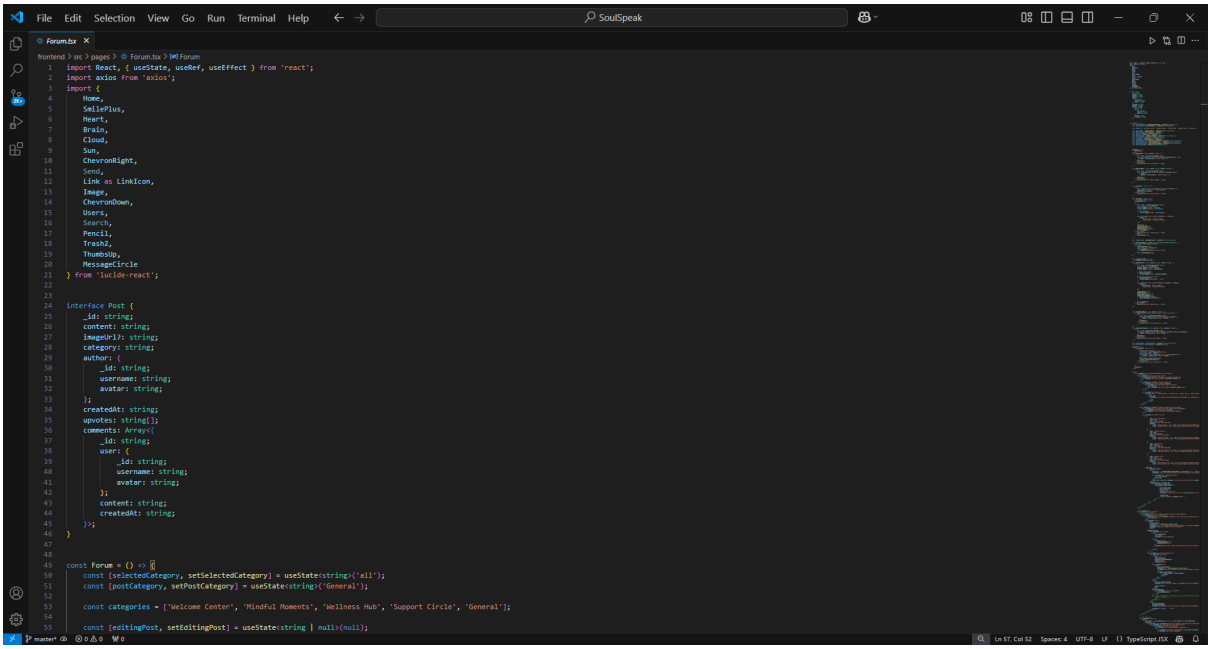
CompanionTest.jsx(Handles the front end interface of Questioning the User for companion compatibility)



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

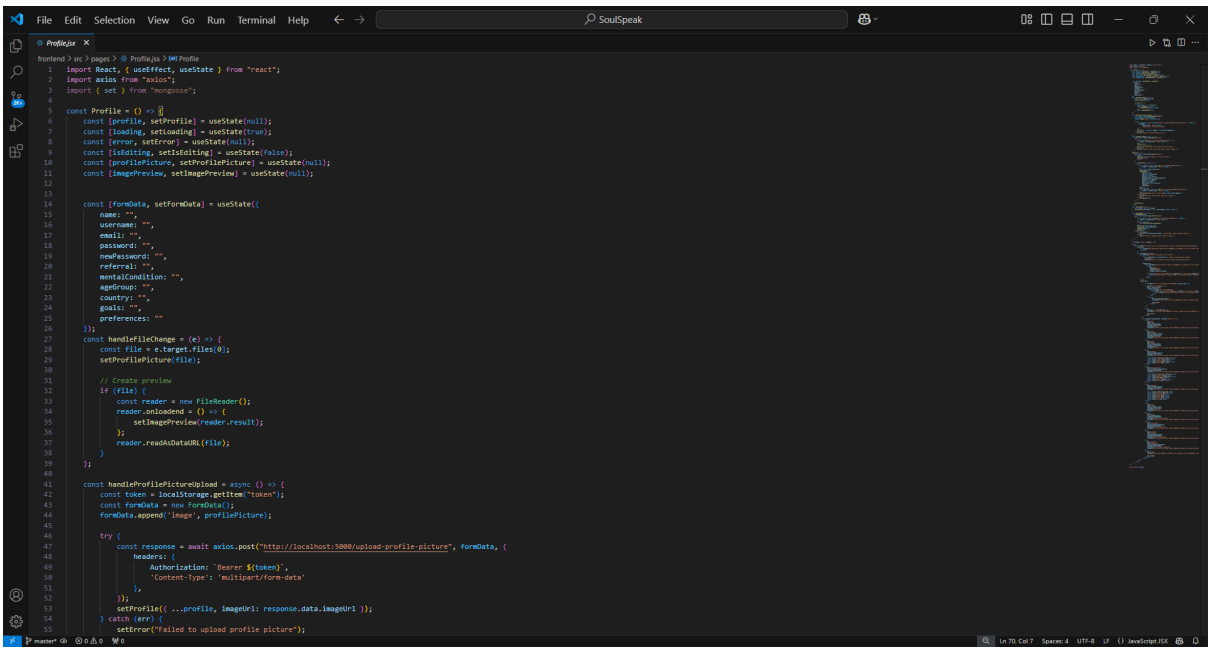


Login.jsx (Handles the front end interface of Login page)



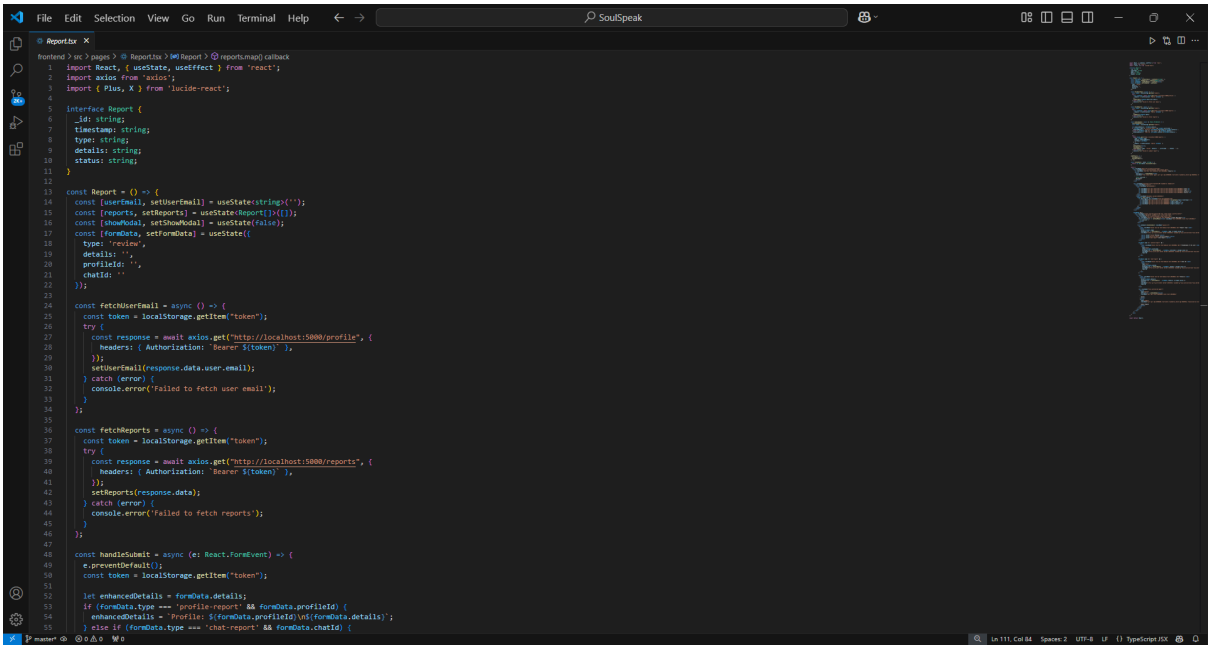
```tsx
import React, { useState, useRef, useEffect } from 'react';
import axios from 'axios';
import {
    Home,
    SmilePlus,
    Heart,
    Brain,
    Cloud,
    Sun,
    ChevronRight,
    Send,
    Link as LinkIcon,
    Image,
    ChevronDown,
    User,
    Search,
    Pencil,
    Trash2,
    ThumbsUp,
    MessageCircle
} from 'lucide-react';


interface Post {
    _id: string;
    content: string;
    imageUrl?: string;
    category: string;
    author: {
        _id: string;
        username: string;
        avatar: string;
    };
    createdAt: string;
    upvotes: string[];
    comments: Array<{
        _id: string;
        user: {
            _id: string;
            username: string;
            avatar: string;
        };
        content: string;
        createdAt: string;
    }>;
}


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>(null);
```

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



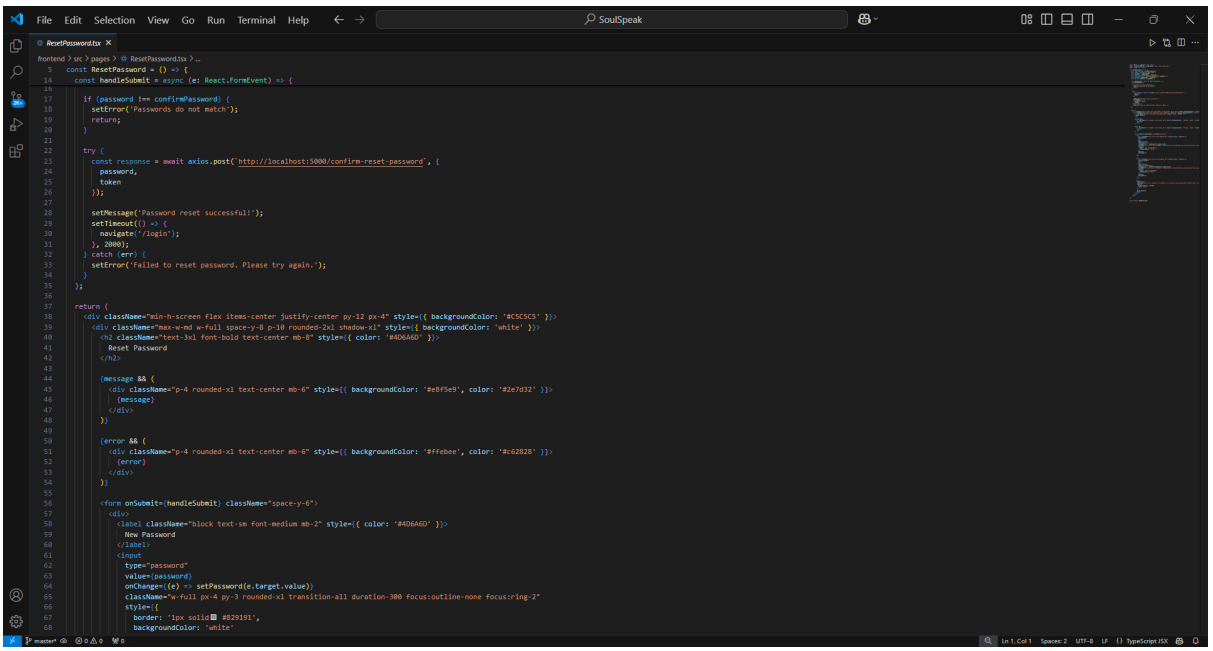
```tsx
import React, { useEffect, useState } from "react";
import axios from "axios";
import { set } from "mongoose";

const Profile = () => {
    const [profile, setProfile] = useState(null);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);
    const [isEditing, setIsEditing] = useState(false);
    const [profilePicture, setProfilePicture] = useState(null);
    const [imagePreview, setImagePreview] = useState(null);


    const [formData, setFormData] = useState({
        name: "",
        username: "",
        email: "",
        password: "",
        newPassword: "",
        referral: "",
        mentalCondition: "",
        ageGroup: "",
        country: "",
        goals: "",
        preferences: ""
    });
    const handleFileChange = (e) => {
        const file = e.target.files[0];
        setProfilePicture(file);

        // Create preview
        if (file) {
            const reader = new FileReader();
            reader.onloadend = () => {
                setImagePreview(reader.result);
            };
            reader.readAsDataURL(file);
        }
    };

    const handleProfilePictureUpload = async () => {
        const token = localStorage.getItem("token");
        const formData = new FormData();
        formData.append('image', profilePicture);

        try {
            const response = await axios.post("http://localhost:5000/upload-profile-picture", formData, {
                headers: {
                    Authorization: `Bearer ${token}`,
                    'Content-Type': 'multipart/form-data'
                },
            });
            setProfile({ ...profile, imageUrl: response.data.imageUrl });
        } catch (err) {
            setError("Failed to upload profile picture");
```

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



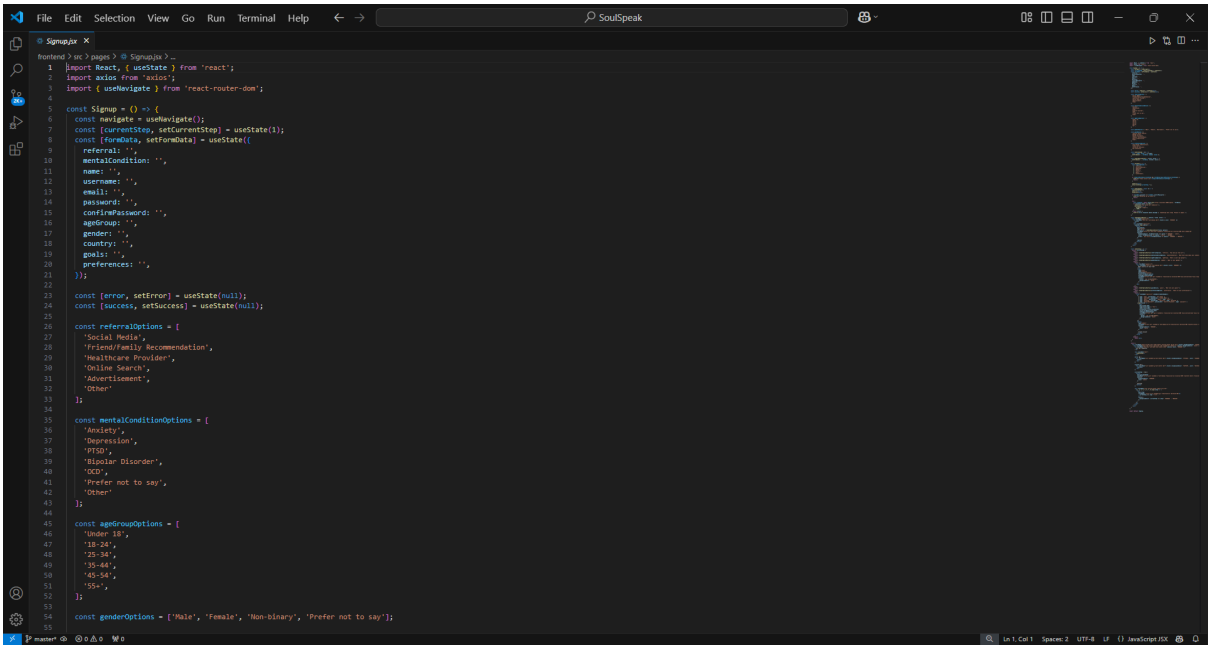
```tsx
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { Plus, X } from 'lucide-react';

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

const Report = () => {
  const [userEmail, setUserEmail] = useState<string>('');
  const [reports, setReports] = useState<Report[]>([]);
  const [showModal, setShowModal] = useState(false);
  const [formData, setFormData] = useState({
    type: 'review',
    details: '',
    profileId: '',
    chatId: ''
  });

  const fetchUserEmail = async () => {
    const token = localStorage.getItem("token");
    try {
      const response = await axios.get("http://localhost:5000/profile", {
        headers: { Authorization: `Bearer ${token}` },
      });
      setUserEmail(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 = formData.details;
    if (formData.type === 'profile-report' && formData.profileId) {
      enhancedDetails = `Profile: ${formData.profileId}\n${formData.details}`;
    } else if (formData.type === 'chat-report' && formData.chatId) {
```

ResetPassword.tsx



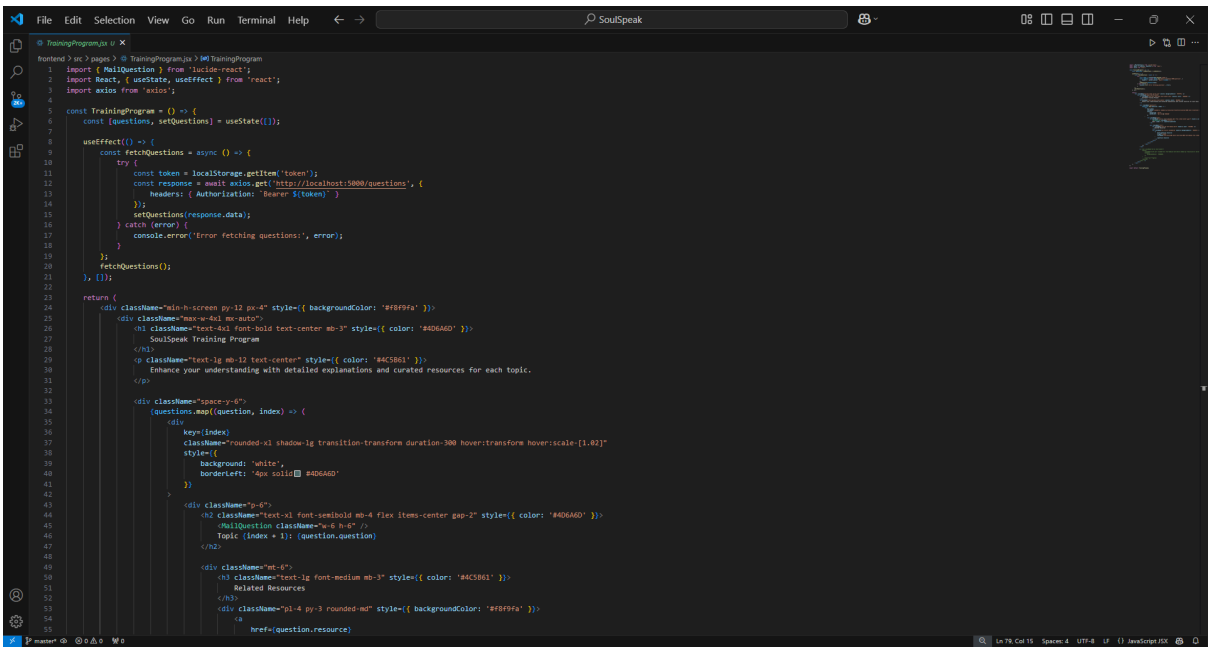
```tsx
const ResetPassword = () => {
  const handleSubmit = async (e: React.FormEvent) => {

    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={{ backgroundColor: '#C5C5C5' }}>
      <div className="max-w-md w-full space-y-8 p-10 rounded-2xl shadow-xl" style={{ backgroundColor: 'white' }}>
        <h2 className="text-3xl font-bold text-center mb-8" style={{ color: '#4D6A6D' }}>
          Reset Password
        </h2>

        {message && (
          <div className="p-4 rounded-xl text-center mb-6" style={{ backgroundColor: '#e8f5e9', color: '#2e7d32' }}>
            {message}
          </div>
        )}

        {error && (
          <div className="p-4 rounded-xl text-center mb-6" style={{ backgroundColor: '#ffebee', color: '#c62828' }}>
            {error}
          </div>
        )}

        <form onSubmit={handleSubmit} className="space-y-6">
          <div>
            <label className="block text-sm font-medium mb-2" style={{ color: '#4D6A6D' }}>
              New Password
            </label>
            <input
              type="password"
              value={password}
              onChange={(e) => setPassword(e.target.value)}
              className="w-full px-4 py-3 rounded-xl transition-all duration-300 focus:outline-none focus:ring-2"
              style={{
                border: '1px solid #829191',
                backgroundColor: 'white'
```

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



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

const Signup = () => {
  const navigate = useNavigate();
  const [currentStep, setCurrentStep] = useState(1);
  const [formData, setFormData] = useState({
    referral: '',
    mentalCondition: '',
    name: '',
    username: '',
    email: '',
    password: '',
    confirmPassword: '',
    ageGroup: '',
    gender: '',
    country: '',
    goals: '',
    preferences: '',
  });

  const [error, setError] = useState(null);
  const [success, setSuccess] = useState(null);

  const referralOptions = [
    'Social Media',
    'Friend/Family Recommendation',
    'Healthcare Provider',
    'Online Search',
    'Advertisement',
    'Other'
  ];

  const mentalConditionOptions = [
    'Anxiety',
    'Depression',
    'PTSD',
    'Bipolar Disorder',
    'OCD',
    'Prefer not to say',
    'Other'
  ];

  const ageGroupOptions = [
    'Under 18',
    '18-24',
    '25-34',
    '35-44',
    '45-54',
    '55+',
  ];

  const genderOptions = ['Male', 'Female', 'Non-binary', 'Prefer not to say'];
```

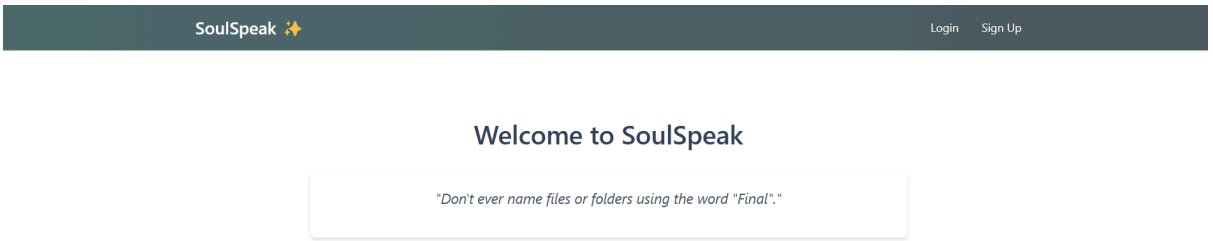
TrainingProgram.jsx (Handles the front end interface of Resource Training page)

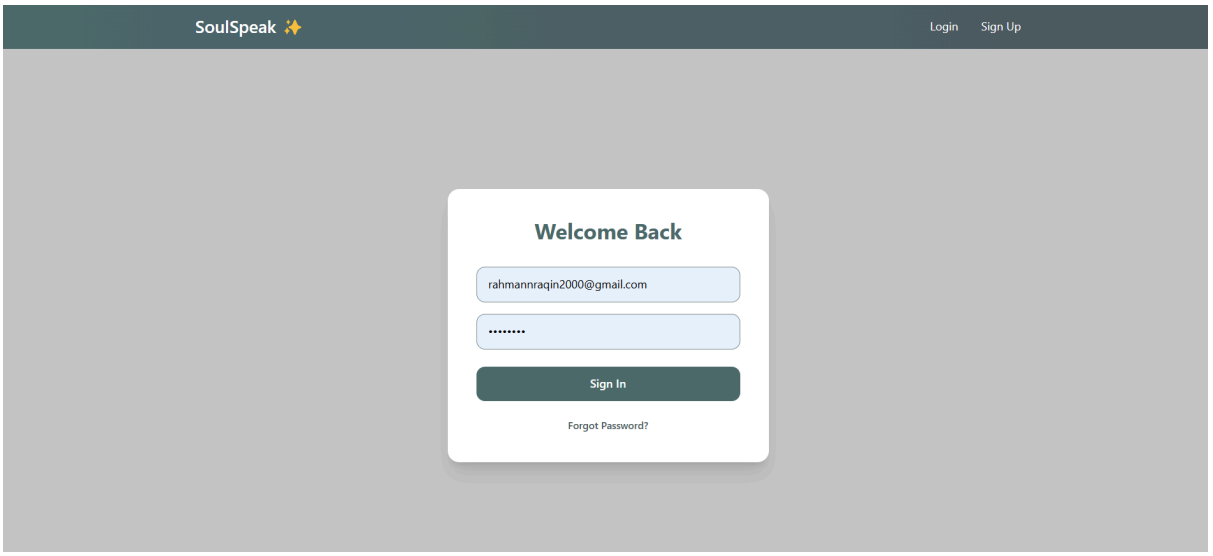


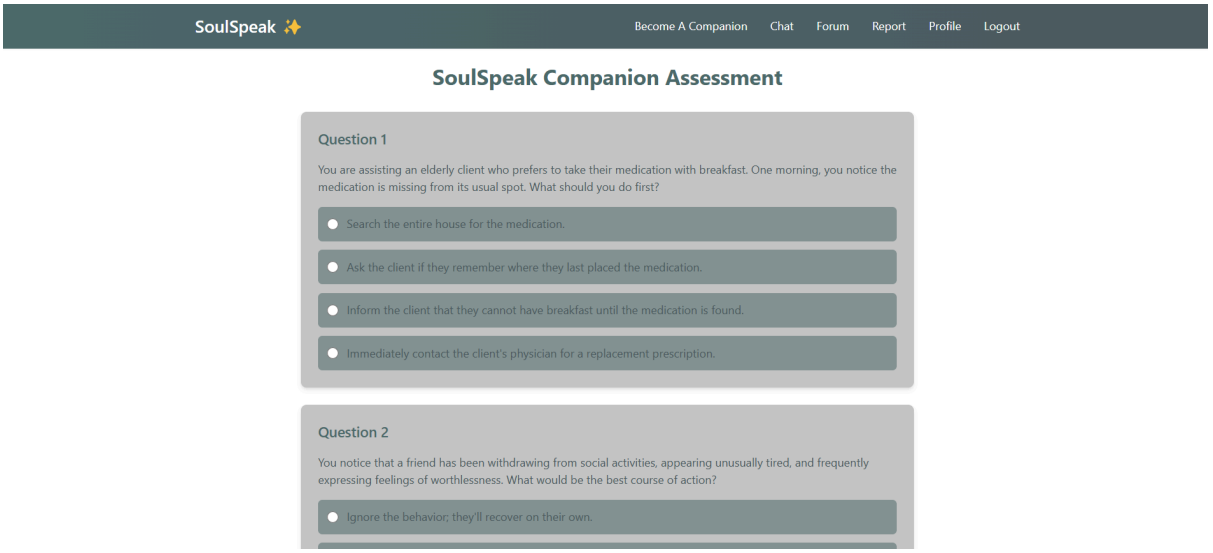
```jsx
import { MailQuestion } from 'lucide-react';
import React, { useState, useEffect } from 'react';
import axios from 'axios';

const TrainingProgram = () => {
    const [questions, setQuestions] = useState([]);

    useEffect(() => {
        const fetchQuestions = async () => {
            try {
                const token = localStorage.getItem('token');
                const response = await axios.get('http://localhost:5000/questions', {
                    headers: { Authorization: `Bearer ${token}` }
                });
                setQuestions(response.data);
            } catch (error) {
                console.error('Error fetching questions:', error);
            }
        };
        fetchQuestions();
    }, []);

    return (
        <div className="min-h-screen py-12 px-4" style={{ backgroundColor: '#F8F9FA' }}>
            <div className="max-w-4xl mx-auto">
                <h1 className="text-4xl font-bold text-center mb-3" style={{ color: '#4D6A6D' }}>
                    SoulSpeak Training Program
                </h1>
                <p className="text-lg mb-12 text-center" style={{ color: '#4C5861' }}>
                    Enhance your understanding with detailed explanations and curated resources for each topic.
                </p>

                <div className="space-y-6">
                    {questions.map((question, index) => (
                        <div
                            key={index}
                            className="rounded-xl shadow-lg transition-transform duration-300 hover:transform hover:scale-[1.02]"
                            style={{
                                background: 'white',
                                borderLeft: '4px solid #4D6A6D'
                            }}
                        >
                            <div className="p-6">
                                <h2 className="text-xl font-semibold mb-4 flex items-center gap-2" style={{ color: '#4D6A6D' }}>
                                    <MailQuestion className="w-6 h-6" />
                                    Topic {index + 1}: {question.question}
                                </h2>

                                <div className="mt-6">
                                    <h3 className="text-lg font-medium mb-3" style={{ color: '#4C5861' }}>
                                        Related Resources
                                    </h3>
                                    <div className="pl-4 py-3 rounded-md" style={{ backgroundColor: '#F8F9FA' }}>
                                        <a
                                            href={question.resource}
```

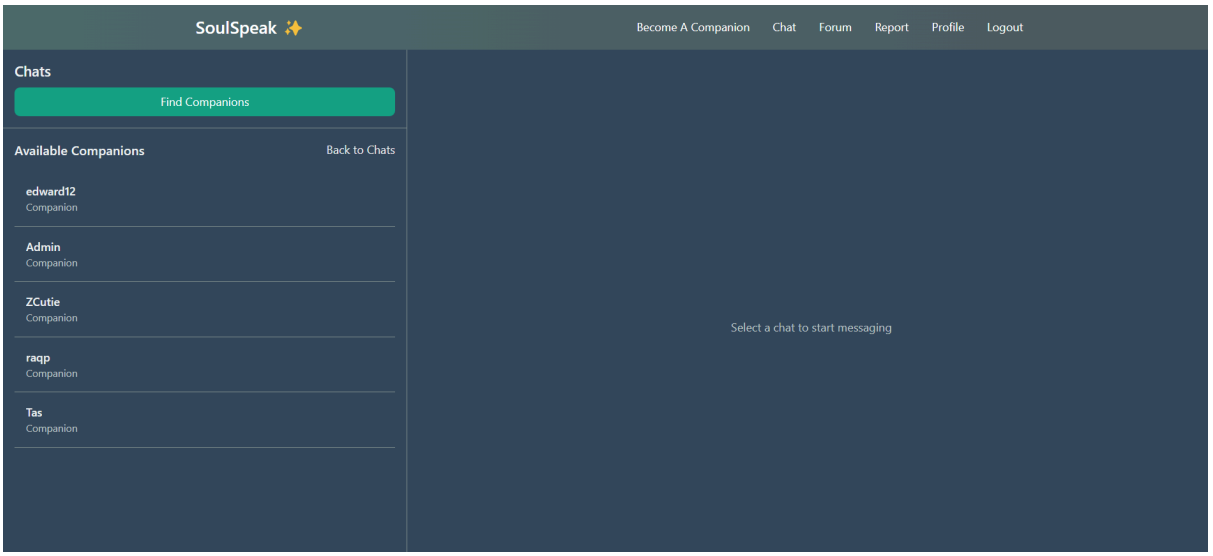
4. User Manual

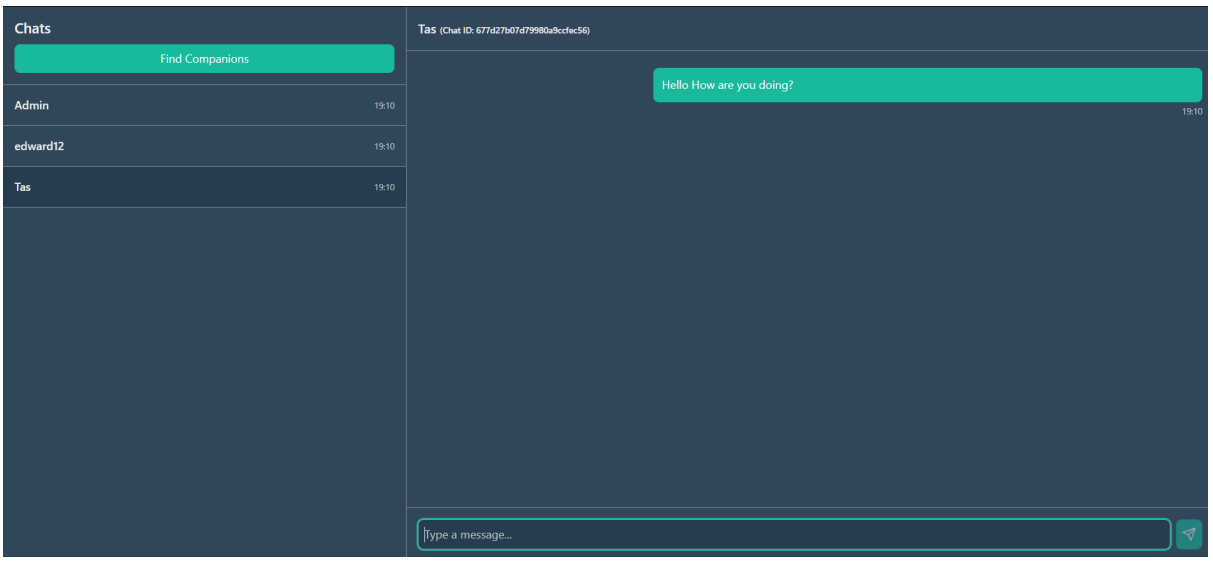
How a new user will navigate or use the system with screenshots.

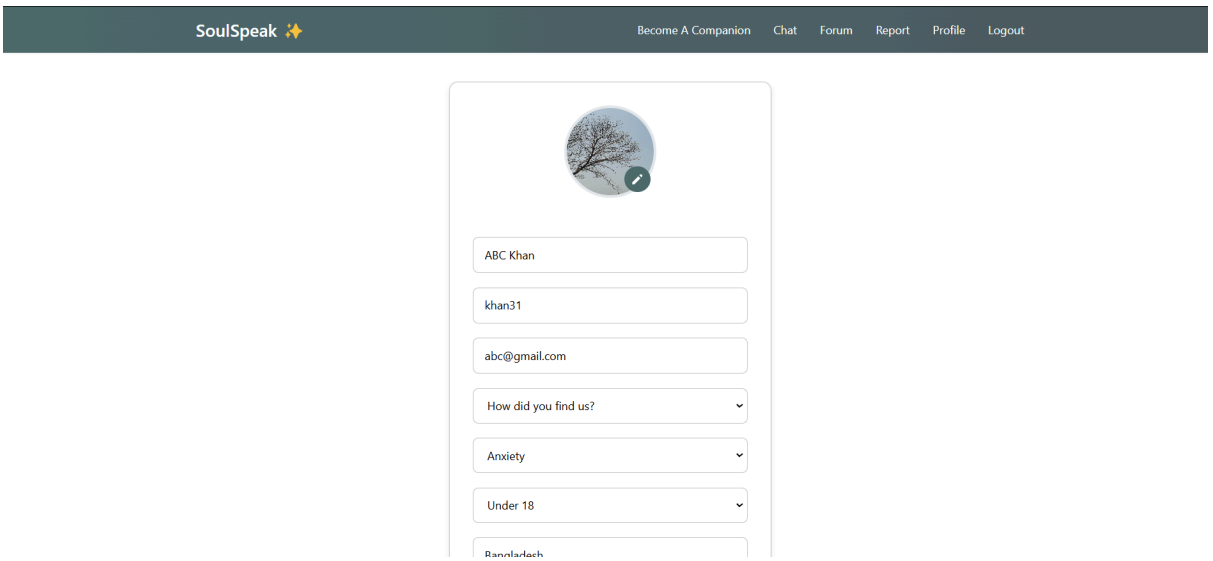


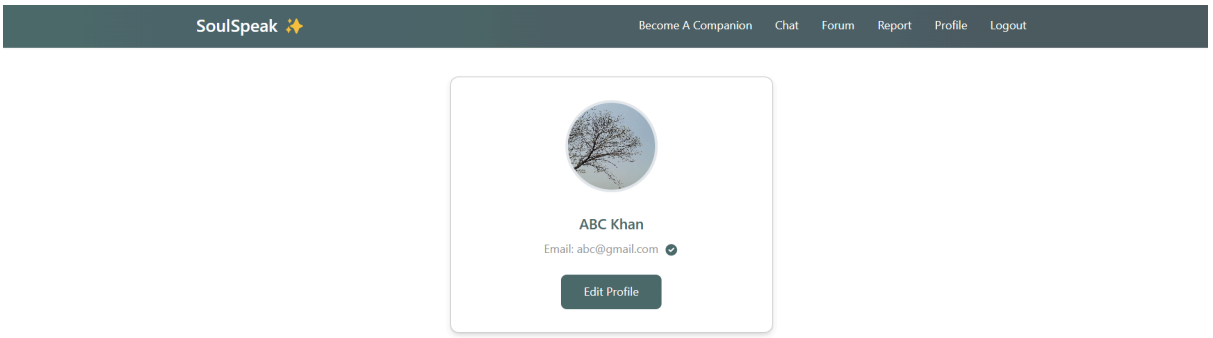


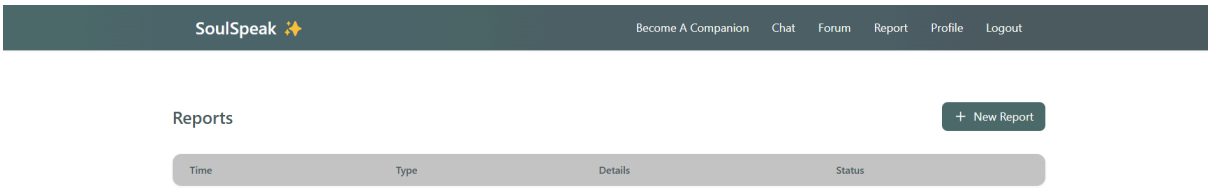

SoulSpeak
Become A Companion
Chat
Forum
Report
Profile
Logout
SoulSpeak Companion Assessment
Question 1
You are assisting an elderly client who prefers to take their medication with breakfast. One morning, you notice the medication is missing from its usual spot. What should you do first?
Search the 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.

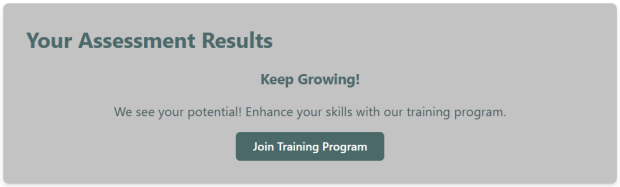

SoulSpeak
Become A Companion
Chat
Forum
Report
Profile
Logout
Chats
Find Companions
Available Companions
Back to Chats
edward12
Companion
Admin
Companion
ZCutie
Companion
raqp
Companion
Tas
Companion
Select a chat to start messaging

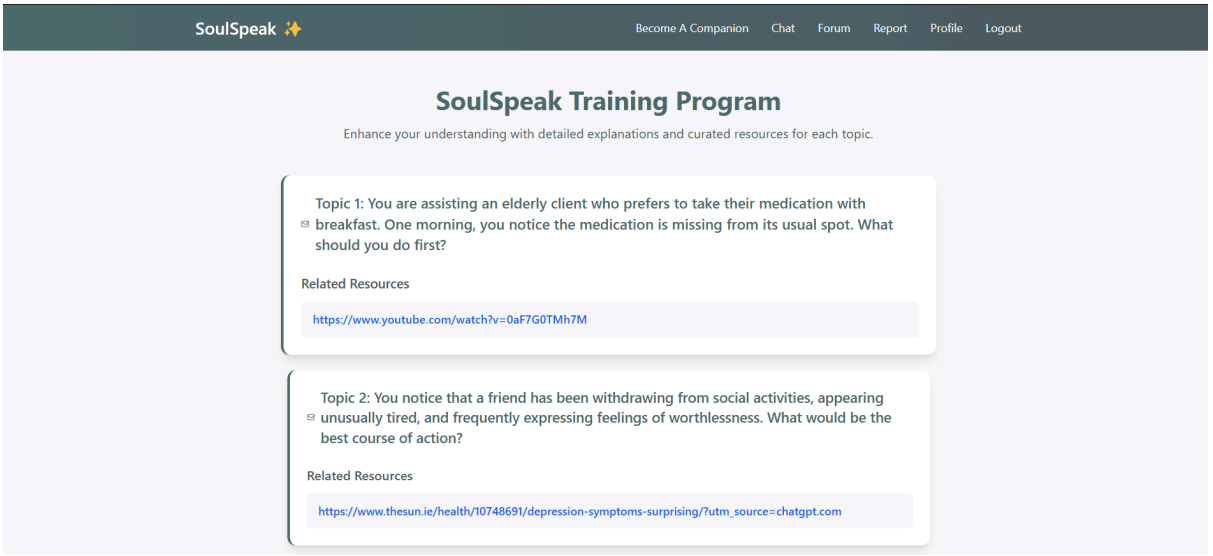

Chats
Find Companions
Admin
19:10
edward12
19:10
Tas
19:10
Tas (Chat ID: 677d27b07d79980a9ccfec56)
Hello How are you doing?
19:10
Type a message...

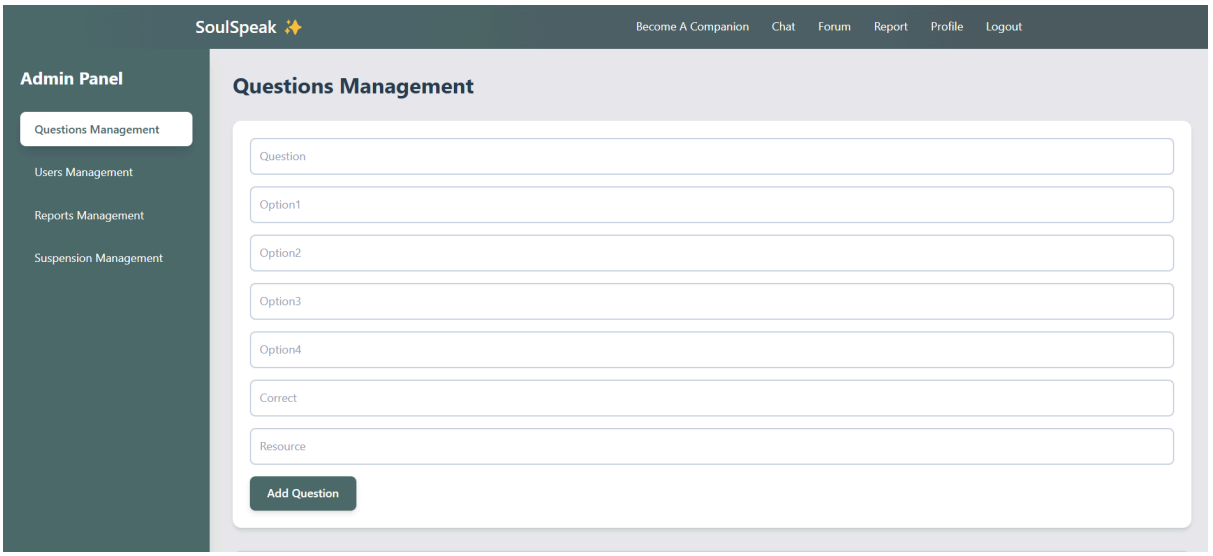

SoulSpeak
Become A Companion
Chat
Forum
Report
Profile
Logout
ABC Khan
khan31
abc@gmail.com
How did you find us?
Anxiety
Under 18

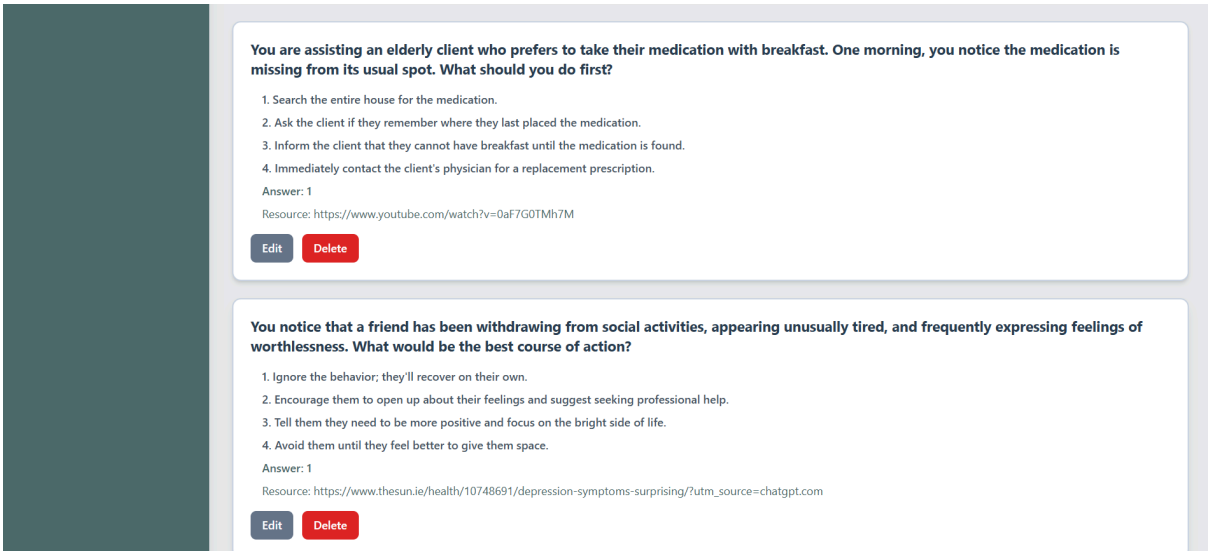

SoulSpeak
Become A Companion
Chat
Forum
Report
Profile
Logout
ABC Khan
Email: abc@gmail.com
Edit Profile

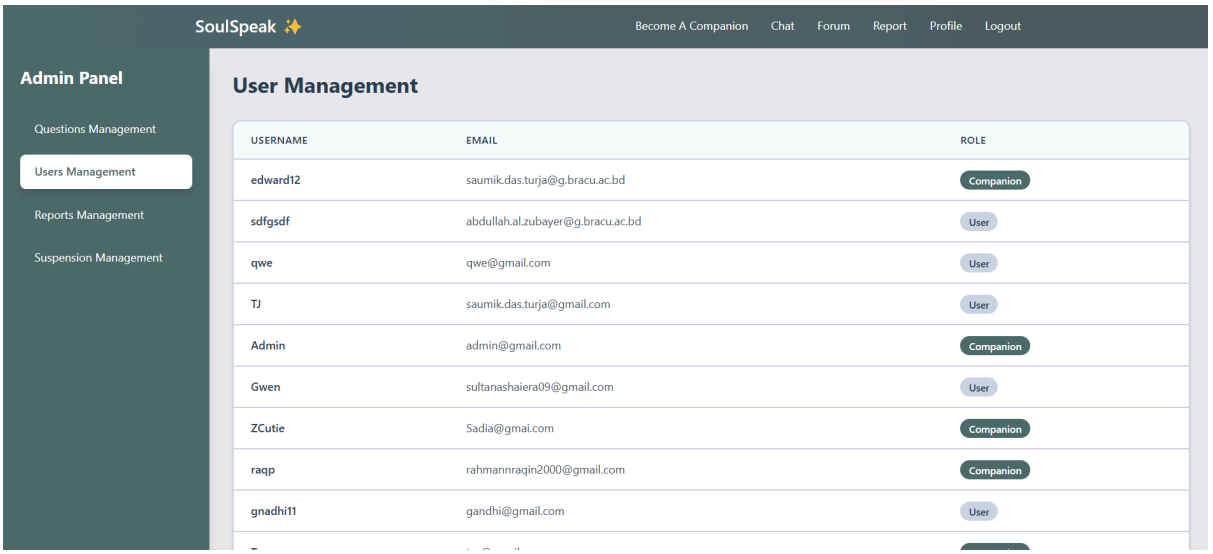

SoulSpeak
Become A Companion
Chat
Forum
Report
Profile
Logout
Reports
New Report
Time
Type
Details
Status

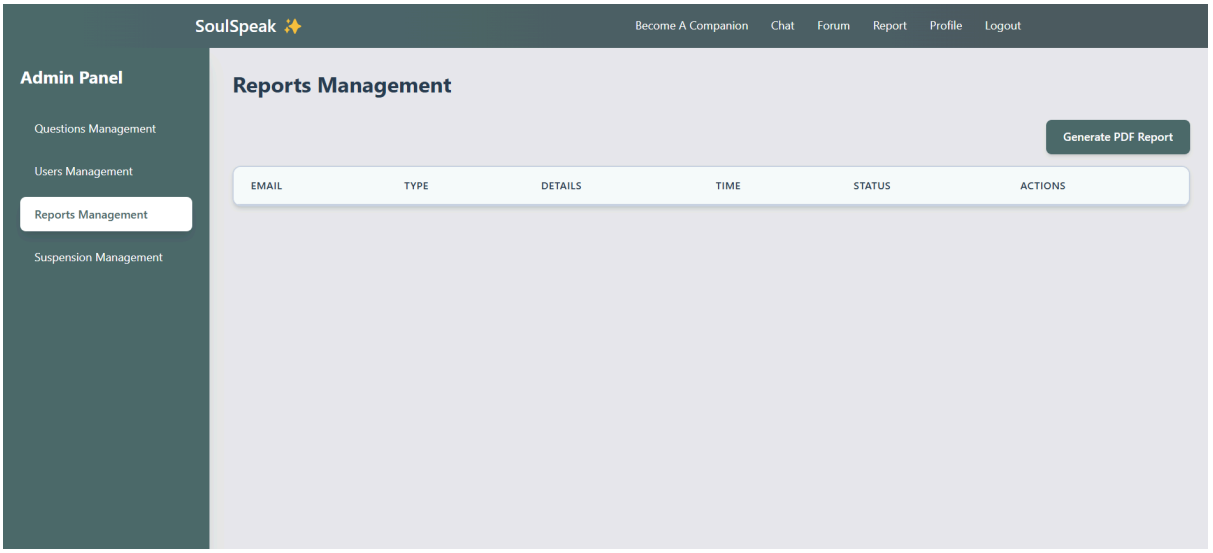

Your Assessment Results
Keep Growing!
We see your potential! Enhance your skills with our training program.
Join Training Program

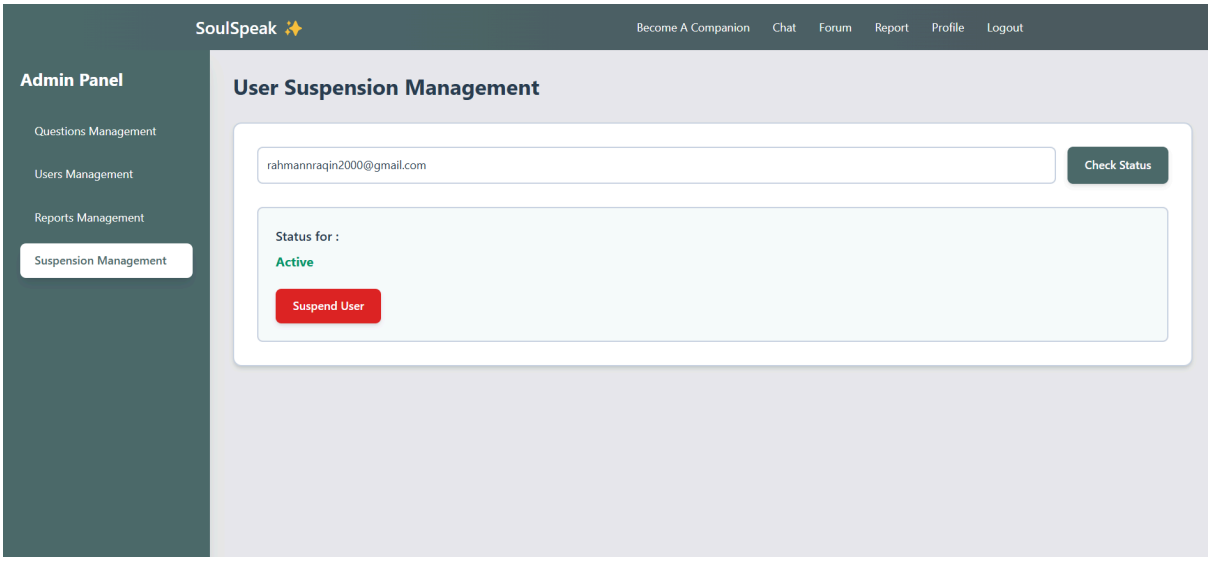

SoulSpeak
Become A Companion
Chat
Forum
Report
Profile
Logout
SoulSpeak Training Program
Enhance your understanding with detailed explanations and curated resources for each topic.
Topic 1: You are assisting an elderly client who prefers to take their medication with breakfast. One morning, you notice the medication is missing from its usual spot. What should you do first?
Related Resources
https://www.youtube.com/watch?v=0aF7G0TMh7M
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


SoulSpeak
Become A Companion
Chat
Forum
Report
Profile
Logout
Admin Panel
Questions Management
Users Management
Reports Management
Suspension Management
Questions Management
Question
Option1
Option2
Option3
Option4
Correct
Resource
Add Question


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?
1. Search the entire house for the medication.
2. Ask the client if they remember where they last placed the medication.
3. Inform the client that they cannot have breakfast until the medication is found.
4. 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?
1. Ignore the behavior; they'll recover on their own.
2. Encourage them to open up about their feelings and suggest seeking professional help.
3. Tell them they need to be more positive and focus on the bright side of life.
4. 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

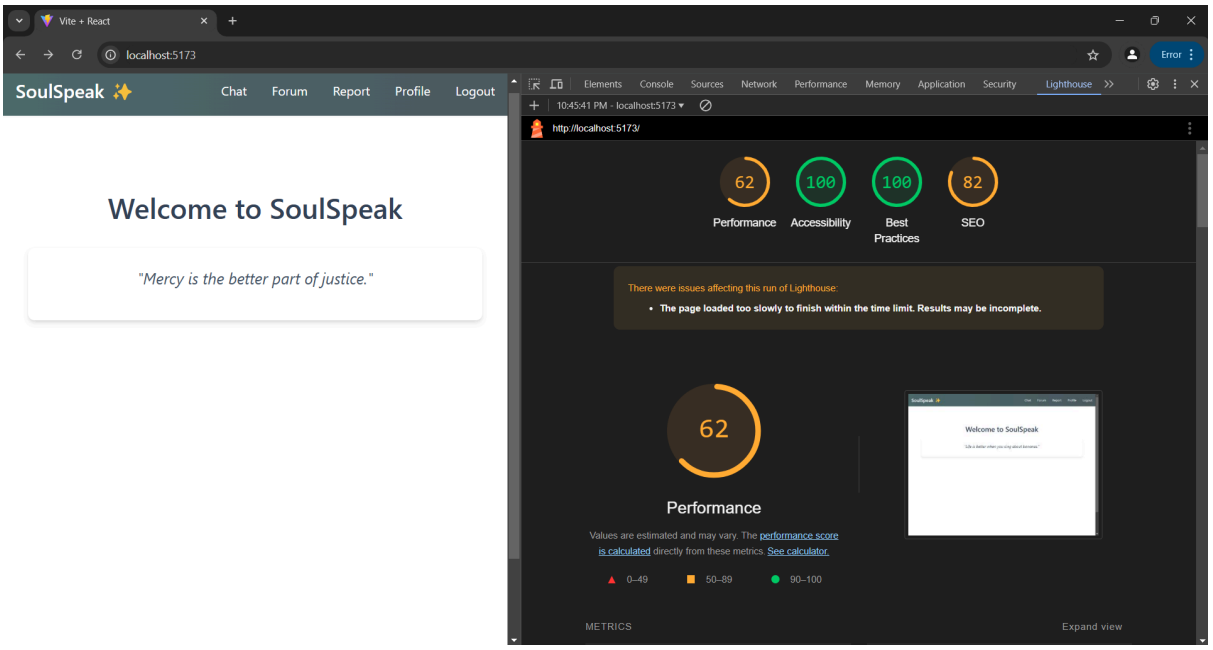

SoulSpeak
Become A Companion
Chat
Forum
Report
Profile
Logout
Admin Panel
Questions Management
Users Management
Reports Management
Suspension Management
User Management
USERNAME
EMAIL
ROLE
edward12
saumik.das.turja@g.bracu.ac.bd
Companion
sdfgsdf
abdullah.al.zubayer@g.bracu.ac.bd
User
qwe
qwe@gmail.com
User
TJ
saumik.das.turja@gmail.com
User
Admin
admin@gmail.com
Companion
Gwen
sultanashaiera09@gmail.com
User
ZCutie
Sadia@gmai.com
Companion
raqp
rahmannraqin2000@gmail.com
Companion
gnadhi11
gandhi@gmail.com
User

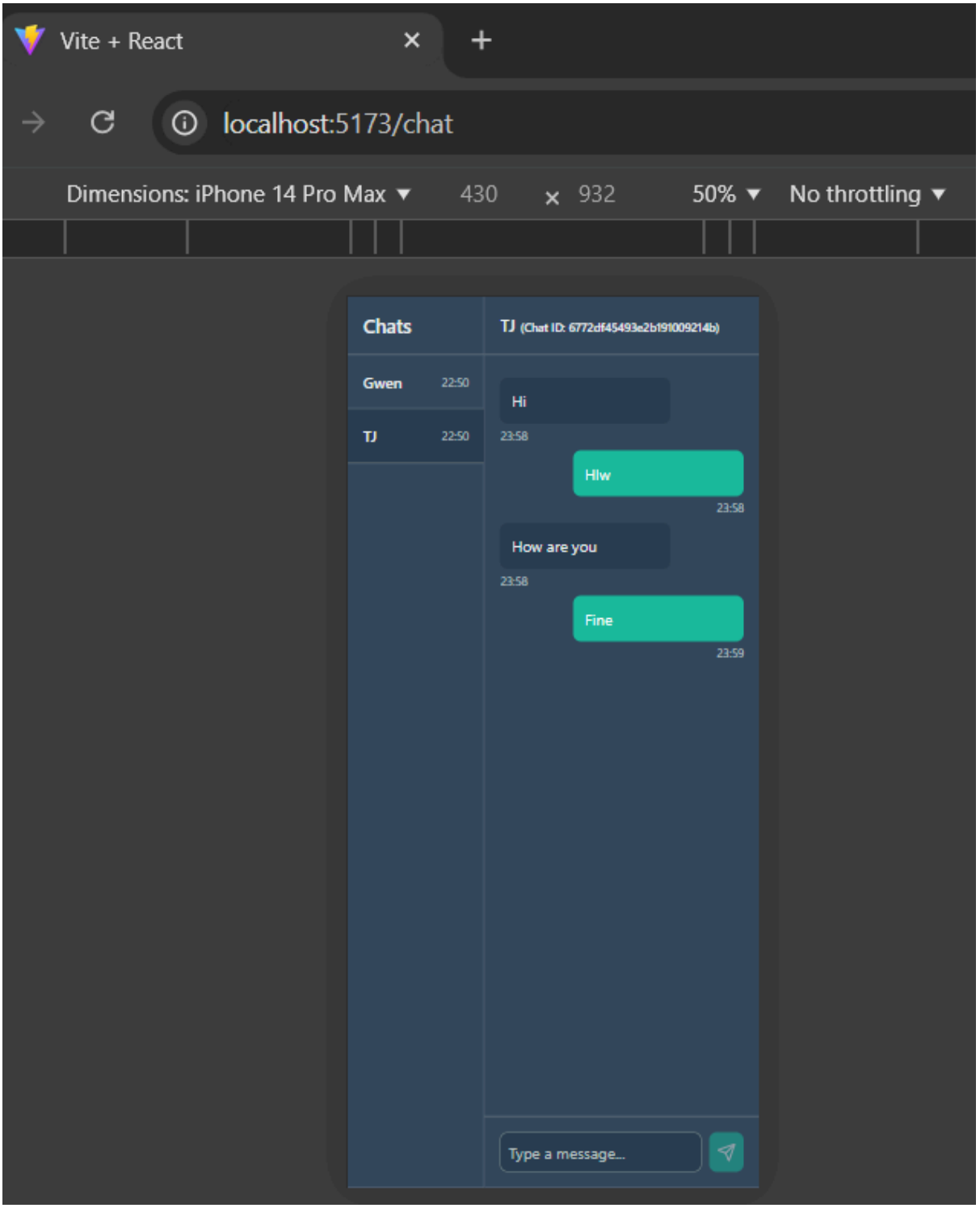

SoulSpeak
Become A Companion
Chat
Forum
Report
Profile
Logout
Admin Panel
Questions Management
Users Management
Reports Management
Suspension Management
Reports Management
Generate PDF Report
EMAIL
TYPE
DETAILS
TIME
STATUS
ACTIONS


SoulSpeak
Become A Companion
Chat
Forum
Report
Profile
Logout
Admin Panel
Questions Management
Users Management
Reports Management
Suspension Management
User Suspension Management
rahmannraqin2000@gmail.com
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:	
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:	
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.	