# ****MindMate – A Comprehensive Mental Health Assessment & Coping Web App****

## ****1. Title Page****

**Project Name:** MindMate

**Team Name:** MindMates

**Team Members:**

Mohammed Shaik Sahil (Leader)

Mohammed Ameen Ul Haq

Rajapatel Mohammed Sumair

Abdullah Abu Imam

**Hackathon Name:** AI Autonomous Hackathon

Table of Contents

[MindMate – A Comprehensive Mental Health Assessment & Coping Web App 1](#_Toc190407813)

[1. Title Page 1](#_Toc190407814)

[2. Abstract 4](#_Toc190407815)

[3. Introduction 5](#_Toc190407816)

[3.1 Background and Context 5](#_Toc190407817)

[3.2 Problem Statement 5](#_Toc190407818)

[3.3 Objectives 5](#_Toc190407819)

[4. Research and Background 6](#_Toc190407820)

[4.1 Literature Review 6](#_Toc190407821)

[4.2 Existing Solutions and Gaps 6](#_Toc190407822)

[5. Proposed Solution 7](#_Toc190407823)

[5.1 App Concept and Workflow 7](#_Toc190407824)

[5.2 Unique Selling Points 7](#_Toc190407825)

[6. Features and Functionality 8](#_Toc190407826)

[6.1 Mental Health Assessment 8](#_Toc190407827)

[6.2 Personalized Coping Recommendations 8](#_Toc190407828)

[6.3 Additional Mental Health Tools 8](#_Toc190407829)

[6.4 User Authentication and Management 8](#_Toc190407830)

[7. Technology Stack 9](#_Toc190407831)

[Front-End 9](#_Toc190407832)

[Back-End 9](#_Toc190407833)

[Database 9](#_Toc190407834)

[8. System Architecture 10](#_Toc190407835)

[8.1 Architecture Overview 10](#_Toc190407836)

[9. Development Process 11](#_Toc190407837)

[9.1 Methodology 11](#_Toc190407838)

[9.2 Communication 11](#_Toc190407839)

[10. Challenges and Solutions 12](#_Toc190407840)

[10.1 Challenges Encountered 12](#_Toc190407841)

[10.2 How We Overcame Them 12](#_Toc190407842)

[11. Security and Privacy Measures 13](#_Toc190407843)

[12. Impact and Benefits 14](#_Toc190407844)

[13. Future Scope 15](#_Toc190407845)

[13.1 Planned Enhancements 15](#_Toc190407846)

[14. Conclusion 16](#_Toc190407847)

[15. References 17](#_Toc190407848)

[16. Appendices 18](#_Toc190407849)

[Additional Notes 23](#_Toc190407850)

## ****2. Abstract****

MindMate is a forward-thinking web application designed to address the escalating mental health crisis by integrating daily mental health assessments with personalized coping strategies. The application uses a diverse set of 150 mental health-related questions—randomly selecting 21 questions each day—to provide users with a comprehensive view of their mental well-being. Unlike traditional assessments such as DASS21, our approach offers increased diversity and accuracy. By leveraging AI-driven recommendations powered by the Lamma API (accessed via Open Router), MindMate guides users through tailored coping strategies and tracks their progress using visual tools like line graphs, streak calendars, and additional features including journaling, habit tracking, and an interactive chatbot. Our holistic design bridges the gap between mere assessment and ongoing mental health management, making it a compelling tool for today's users.

## ****3. Introduction****

### **3.1 Background and Context**

Mental health issues have reached epidemic proportions globally. Despite this, many available tools address only one aspect of mental health care, such as assessments or coping strategies. This fragmented approach often leaves users without the comprehensive support they need for effective mental health management.

### **3.2 Problem Statement**

The lack of integrated, user-friendly platforms for mental health care leaves a significant gap in support services. Users are forced to switch between different apps to assess their mental state, track progress, and access coping strategies. This inefficiency contributes to under-utilization of mental health resources and delays intervention.

### **3.3 Objectives**

* **Comprehensive Assessment:** Provide a daily mental health evaluation using 21 randomized questions from a diverse pool of 150 questions.
* **Personalized Recommendations:** Deliver customized coping strategies based on individual responses through an AI-driven engine (Lamma API via Open Router).
* **Integrated Tools:** Offer additional resources such as journaling, habit tracking, mood tracking via visual graphs, and a chatbot for immediate support.
* **User Engagement:** Encourage regular interaction with features like a streak calendar and progress visualization, fostering a routine that supports mental well-being.
* **Future-Proofing:** Plan for enhanced data security and user experience improvements in future updates.

## ****4. Research and Background****

### **4.1 Literature Review**

Our design and methodology are grounded in established research and recent academic findings:

* **Assessment of Anxiety, Depression and Stress using Machine Learning Models:** This research highlights how machine learning can refine mental health assessments by identifying patterns and predicting trends. ([ResearchGate](https://www.researchgate.net/publication/341906771_Assessment_of_Anxiety_Depression_and_Stress_using_Machine_Learning_Models))
* **JAIT Article on Mental Health Tools:** Provides insights into innovative mental health assessments and the integration of technology in mental health care.
* **ArXiv Study on Mental Health:** Explores how advanced algorithms can enhance the precision of mental health evaluations. ([ArXiv](https://arxiv.org/abs/2110.13710?utm_source=chatgpt.com" \t "_new))
* **Hef Journal Research:** Discusses the importance of holistic mental health tools that combine assessment with ongoing coping strategies. ([Hef Journal](https://hefjournal.org/index.php/HEF/article/view/286?utm_source=chatgpt.com" \t "_new))

### **4.2 Existing Solutions and Gaps**

While many platforms offer mental health assessments or separate coping tools, few combine both into a cohesive, easy-to-use application. MindMate is unique because it integrates:

* **Diverse Assessment Methods:** A large pool of questions ensures that users receive varied and comprehensive evaluations.
* **Continuous Support:** Beyond assessments, the app provides ongoing coping tools, enabling users to manage their mental health actively.
* **Data Visualization:** Tracking progress over time with visual tools helps users see improvements or patterns in their mental health.

## ****5. Proposed Solution****

### **5.1 App Concept and Workflow**

MindMate is designed as a user-friendly web application with a clear, intuitive workflow:

1. **Daily Assessment:** Each day, the user logs in and completes a mental health assessment consisting of 21 randomized questions drawn from a bank of 150. This ensures a fresh and diverse set of queries that capture a wide range of mental health indicators.
2. **Personalized Feedback:** Upon completion, the responses are processed using an AI engine (Lamma API via Open Router) that generates personalized recommendations for coping strategies.
3. **Progress Visualization:** Users can view their progress over time through interactive line graphs that track changes in their mental health metrics.
4. **Additional Tools:** The dashboard also includes a streak calendar to encourage daily participation, a journal for reflective entries, and a habit tracker to build and maintain healthy routines. An integrated chatbot offers real-time support and guidance.
5. **User Authentication:** Secure login and sign-up ensure that personal data is protected and accessible only by the user.

### **5.2 Unique Selling Points**

* **Dynamic, Randomized Assessments:** Our assessment mechanism ensures that users face a diverse set of questions each day, preventing monotony and increasing the accuracy of self-evaluation.
* **Tailored Recommendations:** The use of AI allows the app to offer personalized coping strategies, making the advice relevant to the user's specific mental health state.
* **Holistic Approach:** By merging assessment, visualization, and supportive tools into one platform, MindMate provides a complete ecosystem for mental well-being.
* **User Engagement:** Features such as the streak calendar and progress graphs motivate users to maintain a regular mental health routine.

## ****6. Features and Functionality****

### **6.1 Mental Health Assessment**

* **Dynamic Questionnaire:** Every day, 21 questions are randomly selected from a pool of 150, ensuring that users receive a broad range of insights into their mental state.
* **Daily Evaluations:** Users are encouraged to complete the assessment daily, allowing for continuous monitoring and early detection of any concerning changes.

### **6.2 Personalized Coping Recommendations**

* **AI-Driven Guidance:** Utilizing the Lamma API (accessed via Open Router), the app processes responses to generate personalized recommendations that address the user’s specific mental health needs.
* **Visual Feedback:** Recommendations are accompanied by visual representations, such as line graphs that display trends over time, helping users understand their progress and areas needing improvement.

### **6.3 Additional Mental Health Tools**

* **Journaling:** Users can keep a daily journal to record their thoughts and feelings, providing a space for self-reflection and emotional expression.
* **Habit Tracking:** The app includes a habit tracker that enables users to set and monitor personal goals, much like fitness or diet tracking apps.
* **Streak Calendar:** A streak calendar motivates users to consistently engage with the app by displaying the number of consecutive days they have completed the assessment.
* **Chatbot Support:** An integrated chatbot offers instant conversational support, answering queries, providing encouragement, and guiding users toward appropriate resources.

### **6.4 User Authentication and Management**

* **Secure Login/Sign-Up:** The app features a secure authentication system, ensuring that user data remains private and that only registered users can access their personal information.
* **Data Integrity:** Although full encryption is planned for future iterations, current measures ensure that sensitive data is managed responsibly.

## ****7. Technology Stack****

### **Front-End**

* **Languages:** HTML, CSS, JavaScript
* **Design:** A responsive design that adapts to different screen sizes, ensuring that the app is accessible on desktops, tablets, and smartphones.
* **User Interface:** Interactive charts, clear navigation menus, and intuitive controls enhance user experience.

### **Back-End**

* **Framework:** Python Django
* **Server-Side Logic:** Django handles the core application logic, user authentication, and API interactions efficiently.
* **API Integration:** Seamless integration with the Lamma API via Open Router enables the app to process user assessments and generate tailored recommendations.

### **Database**

* **Database System:** NoSQL MongoDB
* **Data Handling:** MongoDB’s flexible schema design allows for efficient storage and retrieval of diverse data types including user profiles, daily assessment records, journal entries, and habit tracking data.

## ****8. System Architecture****

### **8.1 Architecture Overview**

The architecture of MindMate is structured to ensure scalability, performance, and ease of maintenance:

* **User Interface Layer:**
  + Developed using HTML, CSS, and JavaScript to provide a responsive and engaging front-end experience.
* **Application Logic Layer:**
  + Powered by Python Django, this layer manages user sessions, processes assessments, and communicates with external APIs.
* **Data Storage Layer:**
  + MongoDB is used to store all user data securely, including assessment records and additional tool usage (journals, habits, etc.).
* **External API Integration:**
  + The Lamma API (accessed via Open Router) is integrated to analyze assessment results and provide personalized coping recommendations.

## ****9. Development Process****

### **9.1 Methodology**

Our development process is iterative and user-focused, incorporating feedback at every stage:

* **Agile/Sprint-Based Development:**
  + We divided the project into short sprints, each targeting specific features such as the assessment module, AI integration, and additional tools.
* **Task Breakdown:**
  + **Front-End Development:** Creating the user interface with HTML, CSS, and JavaScript.
  + **Back-End Development:** Implementing the business logic and API integration using Django.
  + **Testing and Refinement:** Conducting regular testing sessions to identify issues and incorporate feedback.
* **Milestones:**
  + **Prototype Development:** Establishing basic UI/UX and assessment logic.
  + **AI Recommendation Integration:** Connecting the Lamma API to generate personalized feedback.
  + **Full Feature Integration:** Combining journaling, habit tracking, visualization, and chatbot functionality.
  + **Final Testing and Debugging:** Ensuring all features work seamlessly before submission.

### **9.2 Communication**

* **Tools Used:**
  + **Slack:** For organized, real-time communication among team members.
  + **WhatsApp:** For quick updates and immediate problem-solving discussions.
* These tools ensured that the team remained in constant contact, allowing for rapid feedback and collaborative problem solving throughout the project lifecycle.

## ****10. Challenges and Solutions****

### **10.1 Challenges Encountered**

* **Question Randomization:**
  + **Challenge:** Ensuring that the randomized selection of 21 questions from a pool of 150 provides balanced and comprehensive coverage.
  + **Solution:** Developed and refined algorithms to randomize questions while maintaining coverage across different mental health indicators.
* **AI Integration:**
  + **Challenge:** Fine-tuning the Lamma API (via Open Router) to generate recommendations that are both empathetic and accurate.
  + **Solution:** Iterative testing and adjustments to the API parameters, ensuring that feedback is relevant to individual assessments.
* **Data Security:**
  + **Challenge:** Securing sensitive user data without full encryption in the current iteration.
  + **Solution:** Implementing robust authentication and planning future updates for enhanced encryption.
* **User Engagement:**
  + **Challenge:** Creating an interface that motivates daily usage without overwhelming the user.
  + **Solution:** Incorporating interactive elements such as streak calendars, progress visualizations, and a supportive chatbot.

### **10.2 How We Overcame Them**

* **Algorithm Optimization:** Ensured comprehensive and balanced question selection through continuous testing.
* **API Tuning:** Refined AI recommendations by analyzing user feedback and making necessary adjustments.
* **Planned Enhancements:** Prioritized security improvements and plan to implement full data encryption in upcoming releases.
* **User Testing:** Conducted sessions with target users to gather feedback and make iterative improvements to the UI/UX.

## ****11. Security and Privacy Measures****

* **Current Measures:**
  + Utilized Django’s secure authentication system to protect user data.
  + Established secure data handling practices within MongoDB.
* **Future Enhancements:**
  + Implement full data encryption (both in transit and at rest) to further secure user information.
  + Regular security audits and updates will be conducted to maintain robust data protection standards.

## ****12. Impact and Benefits****

MindMate offers a transformative approach to mental health management by:

* **Providing a Comprehensive Ecosystem:** Combining assessments with continuous support tools, users receive a full spectrum of mental health care in one app.
* **Enhancing Self-Awareness:** Interactive graphs and visual tools allow users to monitor trends and recognize triggers, empowering them to make informed decisions about their mental health.
* **Promoting Regular Engagement:** Features such as daily assessments, journaling, and habit tracking encourage a routine that supports overall mental well-being.
* **Increasing Accessibility:** A web-based solution ensures that anyone with an internet connection can access high-quality mental health tools, addressing a critical need in today’s digital age.
* **Driving Future Innovation:** The planned enhancements (security, UI/UX, expanded functionalities) position MindMate to continuously evolve and meet user needs.

## ****13. Future Scope****

### **13.1 Planned Enhancements**

* **UI/UX Improvements:**
  + Refining the design for a more intuitive and engaging user experience.
  + Conducting additional user testing to optimize navigation and usability.
* **Expanded Toolset:**
  + Incorporating additional tracking features similar to those found in diet and fitness apps, to further personalize the user experience.
* **Advanced AI Functionalities:**
  + Enhancing predictive analysis capabilities to better forecast mental health trends and offer proactive recommendations.
* **Mobile Application Development:**
  + Exploring native iOS and Android applications to broaden accessibility and user engagement.
* **Enhanced Data Security:**
  + Implementing full encryption and regular security updates to ensure data integrity and privacy.

## ****14. Conclusion****

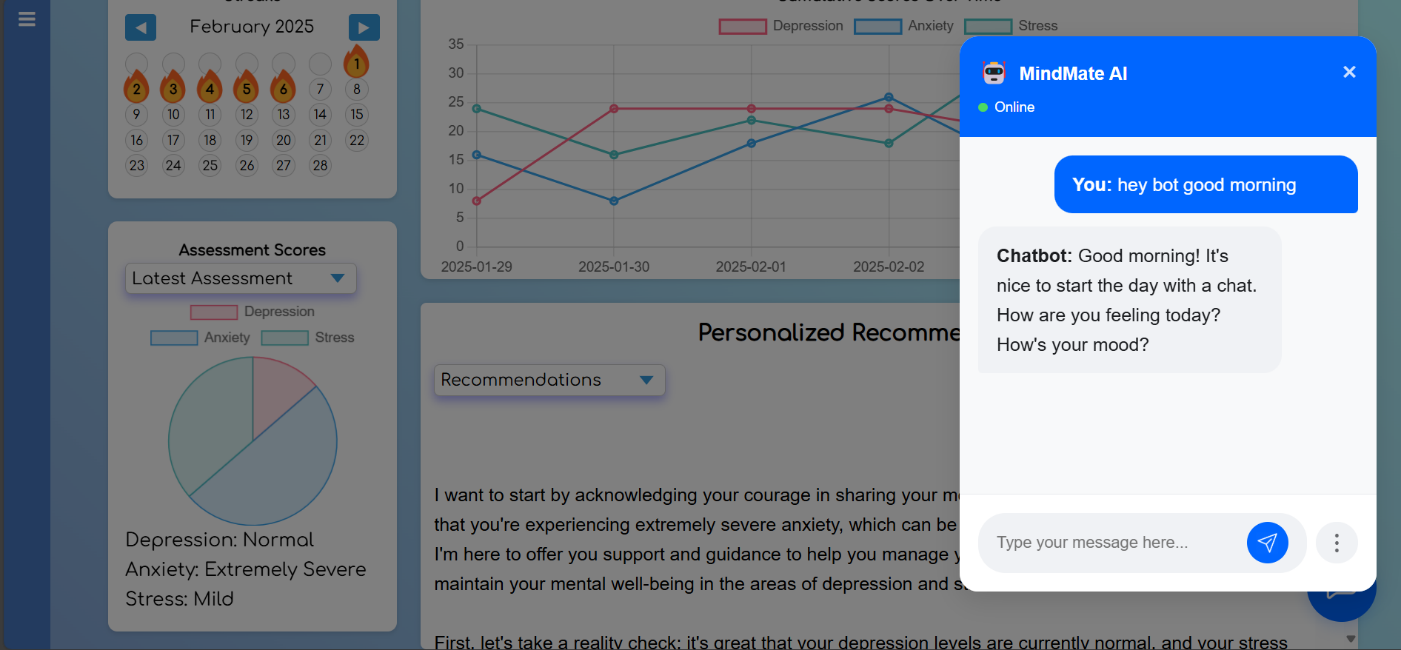
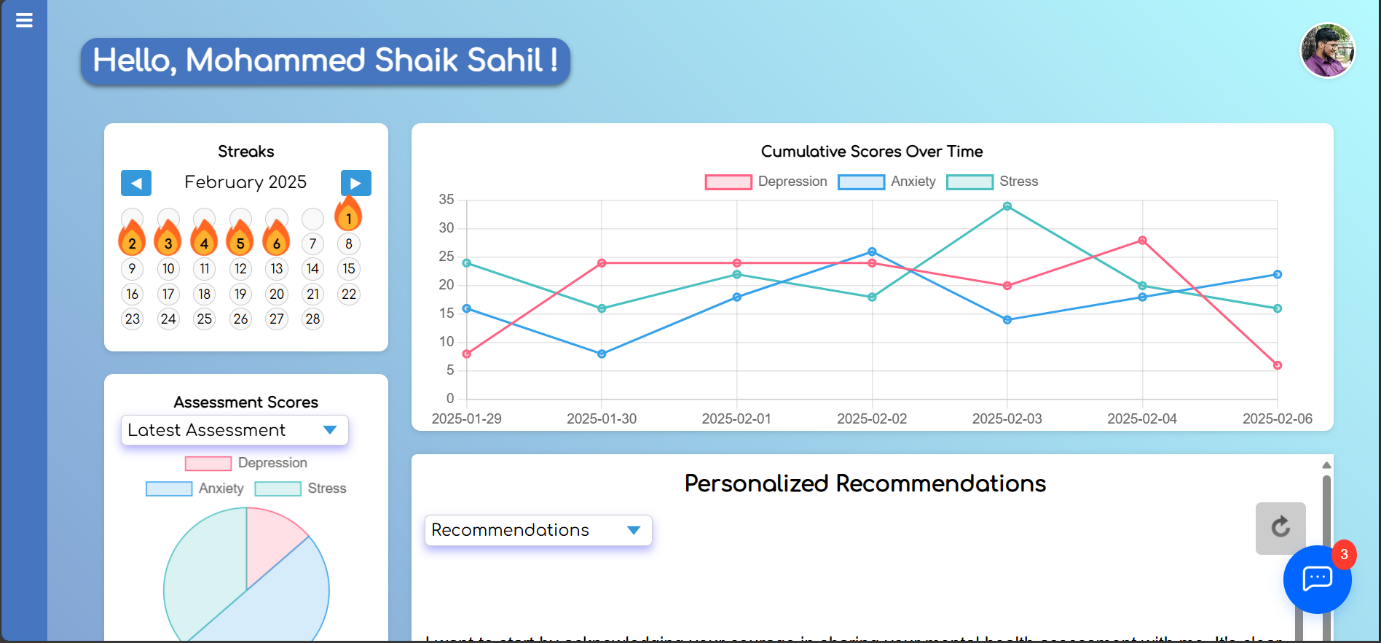
MindMate is not just another mental health app—it is a comprehensive, research-backed platform designed to empower users in managing their mental well-being holistically. By integrating dynamic daily assessments with AI-driven, personalized coping strategies and supporting tools such as journaling, habit tracking, and interactive visualization, MindMate offers a complete solution for mental health management. Our commitment to continuous improvement, enhanced user engagement, and planned security enhancements positions MindMate as a vital resource for addressing the global mental health crisis. We are confident that MindMate will make a significant impact in helping users achieve better mental health and overall well-being.

## ****15. References****

1. [Assessment of Anxiety, Depression and Stress using Machine Learning Models](https://www.researchgate.net/publication/341906771_Assessment_of_Anxiety_Depression_and_Stress_using_Machine_Learning_Models)
2. JAIT Article
3. [ArXiv Paper on Mental Health](https://arxiv.org/abs/2110.13710?utm_source=chatgpt.com)
4. [Hef Journal Research Article](https://hefjournal.org/index.php/HEF/article/view/286?utm_source=chatgpt.com)

## ****16. Appendices****

* **Appendix A:**





* **Appendix B:**

#### **1. Generating Personalized Chat Responses with OpenRouter**

This function builds a dynamic prompt based on the user’s conversation history and assessment data, then uses the Lamma API (via OpenRouter) to generate a response. It also saves the conversation for future reference.

python

def generate\_chat\_with\_openrouter(user\_id, user\_message, user\_conversation, user\_data=None, severity=None, include\_helplines=False):

"""

Generate text using the DeepSeek model via OpenRouter API.

"""

client = openai.OpenAI(

base\_url="https://openrouter.ai/api/v1",

api\_key=key, # Replace with your OpenRouter API key

)

try:

# Build the prompt dynamically using helper function

full\_prompt = build\_prompt(user\_message, user\_conversation, user\_data, severity, include\_helplines)

completion = client.chat.completions.create(

model="meta-llama/llama-3.2-90b-vision-instruct:free", # Specifying the DeepSeek model

messages=[{"role": "system", "content": full\_prompt}],

max\_tokens=2000

)

if completion and hasattr(completion, 'choices') and len(completion.choices) > 0:

response\_content = completion.choices[0].message.content

# Save the assistant's response in the conversation history

save\_conversation(user\_id, "assistant", response\_content, datetime.now())

return response\_content

else:

print("Error: Invalid or empty response from OpenRouter API.")

return None

except Exception as e:

print(f"Error generating text with OpenRouter: {e}")

return None

**Explanation:**

* The function uses the OpenRouter API to generate a response based on a dynamically built prompt.
* It handles API responses and errors gracefully while saving the conversation history for future context.

#### **2. Handling User Assessments**

This snippet outlines how the system checks for an existing assessment for the current day, processes user responses, calculates scores, determines severity, and stores the results.

python

@login\_required

def assessment(request):

today = timezone.now().strftime('%Y-%m-%d')

username = request.user.username

# Check if today's assessment already exists for the user

existing\_assessment = db.assessments.find\_one({

'username': username,

'timestamp': {'$gte': timezone.now().replace(hour=0, minute=0, second=0, microsecond=0)}

})

if existing\_assessment:

questions = existing\_assessment.get('questions', [])

selected\_options = existing\_assessment.get('selected\_options', {})

return render(request, 'assessment.html', {'questions': questions, 'selected\_options': selected\_options})

if request.method == 'POST':

scores = {'depression': 0, 'anxiety': 0, 'stress': 0}

# Mapping question numbers to categories

question\_categories = {

'depression': [3, 5, 10, 13, 16, 17, 21],

'anxiety': [2, 4, 7, 9, 15, 19, 20],

'stress': [1, 6, 8, 11, 12, 14, 18]

}

selected\_options = {} # Collect user responses

questions = request.session.get('assessment\_questions', [])

for index, question in enumerate(questions):

question\_key = f'q{index + 1}'

selected\_option = request.POST.get(question\_key, 0)

selected\_options[question\_key] = selected\_option

for category, q\_list in question\_categories.items():

if (index + 1) in q\_list:

scores[category] += int(selected\_option)

# Multiply scores for scaling

scores = {k: v \* 2 for k, v in scores.items()}

# Determine severity based on score ranges (simplified logic)

def get\_severity(score, category):

# ... (implementation for severity determination)

pass

severity = {

'depression': get\_severity(scores['depression'], 'depression'),

'anxiety': get\_severity(scores['anxiety'], 'anxiety'),

'stress': get\_severity(scores['stress'], 'stress'),

}

# Save the assessment with questions and selected options

db.assessments.insert\_one({

'username': username,

'depression': scores['depression'],

'anxiety': scores['anxiety'],

'stress': scores['stress'],

'severity': severity,

'timestamp': timezone.now(),

'questions': request.session.get('assessment\_questions', []),

'selected\_options': selected\_options

})

return redirect('user\_view')

# ... (code to fetch and display new questions)

**Explanation:**

* The function handles both GET (displaying questions) and POST (processing responses) methods.
* It checks if an assessment has already been taken today to avoid duplication.
* It computes scores for depression, anxiety, and stress, then categorizes the severity of each.

#### **3. Chatbot Integration for Real-Time Support**

This function enables real-time conversational support by integrating user messages, conversation history, and assessment data. It calls the response generation function and returns the model's output.

python

@csrf\_exempt

def chatbot(request):

if request.method == "POST":

user\_id = request.user.username

user\_message = request.POST.get("message")

# Save the user message to conversation history

save\_conversation(user\_id, "user", user\_message, datetime.now())

user\_conversation = get\_conversation(user\_id)

user\_data = db.users.find\_one({'username': user\_id})

latest\_assessment = db.assessments.find\_one({'username': user\_id}, sort=[('timestamp', -1)])

severity = latest\_assessment['severity'] if latest\_assessment else None

# Determine prompt settings based on the message

include\_helplines = "helpline" in user\_message.lower()

include\_user\_info = "report" in user\_message.lower()

response\_text = generate\_chat\_with\_openrouter(

user\_id,

user\_message,

user\_conversation,

user\_data if include\_user\_info else None,

severity if include\_user\_info else None,

include\_helplines

)

if response\_text:

return JsonResponse({"response": response\_text})

else:

return JsonResponse({"error": "Failed to generate a response from the model."})

elif request.method == "GET":

user\_id = request.user.username

conversation\_history = [{"role": msg["role"], "message": msg["message"]} for msg in get\_conversation(user\_id)]

return JsonResponse({"conversation\_history": conversation\_history})

return JsonResponse({"error": "Invalid request method."})

**Explanation:**

* This endpoint accepts POST requests to interact with the chatbot.
* It uses conversation history and optional user data to generate a context-aware response.
* GET requests are handled to display the conversation history.

### Additional Notes

* **Helper Functions:**  
  The code references several helper functions such as build\_prompt, save\_conversation, and get\_conversation. These functions support prompt creation and conversation management but are omitted here for brevity.
* **Database Operations:**  
  MongoDB is used for storing assessments, conversation history, and user data, ensuring persistence and easy retrieval of information.
* **Security Considerations:**  
  The @login\_required and @csrf\_exempt decorators help secure the endpoints, ensuring that only authenticated users can access sensitive operations.