MindfulAI Documentation

Project Overview

MindfulAI is a mental health chatbot application designed to provide support through structured conversations. The chatbot uses the OpenRouter API with Google's Gemini 2.0 Flash model to deliver intelligent responses focused on mental health topics.

The application features a sleek dark-themed UI with glassmorphism effects and animations, making it visually appealing and easy to use. It guides users through a structured conversation flow: identifying problems, exploring issues, finding root causes, and offering solutions.

Core Features

1. User Authentication
   - Secure registration and login system using Supabase
   - Password hashing and secure session management
   - User-specific chat history storage

2. Mental Health Focused Chat
   - Structured conversation flow (problem > explore > root cause > solutions)
   - Can briefly respond to general knowledge questions while redirecting to mental health focus
   - Context-aware responses based on conversation history

3. Responsive UI
   - Dark theme with glassmorphism effects (frosted glass appearance)
   - Smooth animations and transitions
   - Mobile and desktop-friendly design
   - Enter key support for easy message sending

4. Conversation Management
   - Ability to start new conversations
   - History tracking for context-aware assistance
   - Real-time response with typing indicators

Technical Architecture

Frontend:
- HTML5, CSS3, and vanilla JavaScript
- Responsive design with flexbox and CSS variables
- Custom animations and transitions

Backend:
- Python Flask web framework
- RESTful API endpoints for chat functionality
- Session-based authentication

Database:
- PostgreSQL via Supabase
- Tables for users, chat sessions, and messages

AI Integration:
- OpenRouter API as a gateway
- Google's Gemini 2.0 Flash model
- Structured system prompts for mental health focus

Running Locally

Prerequisites:
- Python 3.11 or higher
- PostgreSQL database (via Supabase)
- Required API keys (OpenRouter, Supabase)

Step 1: Environment Setup
- Clone or download the repository
- Install dependencies: pip install -r requirements.txt

Step 2: Environment Variables
Create a .env file with these variables:
- SESSION_SECRET=your_session_secret_key
- DATABASE_URL=your_supabase_connection_string
- SUPABASE_URL=your_supabase_project_url
- SUPABASE_KEY=your_supabase_service_role_key
- OPENROUTER_API_KEY=your_openrouter_api_key

Step 3: Database Setup
- Create a Supabase account at https://supabase.com
- Create a new project
- Get connection string: Project Settings > Database > Connection String > URI
- Replace [YOUR-PASSWORD] with your database password
- Run setup script: python create_supabase_tables.py

Step 4: Start the Application
- Run: gunicorn --bind 0.0.0.0:5000 --reuse-port --reload main:app
- Or simply: python main.py

Step 5: Access the Application
- Open browser and go to: http://localhost:5000

Deploying on Render

Step 1: Create a Render Account
- Sign up at https://render.com

Step 2: Create a New Web Service
- Click "New" and select "Web Service"
- Connect your repository or use "Public Git repository" option

Step 3: Configure the Web Service
- Name: Choose a name (e.g., "mindful-ai")
- Environment: Python
- Build Command: pip install -r requirements.txt
- Start Command: gunicorn main:app
- Instance Type: Free or paid tier

Step 4: Environment Variables
Add these key-value pairs:
- SESSION_SECRET=your_session_secret_key
- DATABASE_URL=your_supabase_connection_string
- SUPABASE_URL=your_supabase_project_url
- SUPABASE_KEY=your_supabase_service_role_key
- OPENROUTER_API_KEY=your_openrouter_api_key

Step 5: Deploy
- Click "Create Web Service"
- Render will build and deploy your application
- Access your application at the provided URL

Step 6: Using render.yaml (Optional)
- In Render dashboard, click "Blueprints" > "New Blueprint Instance"
- Connect your repository
- Update the environment variables
- Deploy the blueprint

Project Structure

Key Files and Directories:

main.py - Entry point for the application
app.py - Core Flask application with routes
openrouter_client.py - Client for AI API integration
create_supabase_tables.py - Database setup script

```
static/ - Static assets
  css/style.css - Main stylesheet
  js/ - JavaScript files
      auth.js - Authentication handling
      chat.js - Chat functionality
```

main.js - General utilities

templates/ - HTML templates
  index.html - Homepage
  login.html - Login page
  register.html - Registration page
  chat.html - Chat interface

Troubleshooting

Database Connection Issues:
- Verify your Supabase connection string and credentials
- Make sure the database is accessible from your deployment environment

API Limits:
- OpenRouter API has usage limits
- Check your usage and consider upgrading your plan if needed

Environment Variables:
- Ensure all required environment variables are set correctly
- Check both local and deployment environments

Session Management:
- If sessions aren't persisting, check your SESSION_SECRET
- Confirm your cookies are being set correctly

Deployment Issues:
- Check Render logs for any build or runtime errors
- Verify that all dependencies are properly installed

Future Enhancements

1. Add more specialized mental health conversation flows
2. Implement message typing animations
3. Create an admin dashboard for monitoring usage
4. Add export functionality for conversation history
5. Implement customizable themes or UI preferences
6. Add voice input/output capabilities
7. Integrate with additional AI models for comparison
8. Create a mobile app version