MindfulAI Deployment Guide

This guide provides step-by-step instructions for deploying your MindfulAI mental health chatbot application using Render for hosting and Supabase for the database.

Application Overview

MindfulAI is a mental health chatbot that helps users identify and address the root causes of their mental health issues. The application:

1. Provides user authentication with secure password hashing
2. Follows a structured conversation flow:
    - Asks about the specific mental health problem
    - Builds the conversation with targeted questions
    - Identifies the root cause
    - Provides practical solutions to address the root cause
3. Uses GitHub's LLaMA AI model for generating empathetic, helpful responses
4. Stores only user credentials in the database (conversations remain in memory)

Prerequisites

Before deployment, ensure you have:

- A GitHub account with a personal access token (GITHUB_TOKEN) with "models:read" permission
- A Supabase account
- A Render account

Step 1: Set Up Supabase Database

1. Go to the Supabase dashboard

2. Click **New Project** and fill in the project details
3. Once created, go to **Project Settings** → **Database**
4. Locate the **Connection String** section and copy the **URI** value under "Transaction pooler" (you'll need to replace `[YOUR-PASSWORD]` with your actual password)
5. Store this URI as it will be used for your DATABASE_URL in Render

Step 2: Create the Required Tables in Supabase

1. In your Supabase project, go to the **SQL Editor** tab
2. Create a new query and paste the following SQL:

```
CREATE TABLE IF NOT EXISTS public.user (
    id SERIAL PRIMARY KEY,
    user_id VARCHAR(64) UNIQUE NOT NULL,
    name VARCHAR(100),
    email VARCHAR(120) UNIQUE,
    password VARCHAR(256) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

1. Click **Run** to create the table

Step 3: Prepare Your Application for Deployment

1. Create a `requirements.txt` file in your project root containing all dependencies:

```
email-validator
flask
flask-cors
flask-login
flask-sqlalchemy
```

```
flask-wtf
gunicorn
psycopg2-binary
requests
sqlalchemy
supabase
werkzeug
```

1. Create a `render.yaml` file in your project root:

```
services:
  - type: web
    name: mindfulai
    env: python
    buildCommand: pip install -r requirements.txt
    startCommand: gunicorn --bind 0.0.0.0:$PORT --workers=1 main:app
    envVars:
      - key: DATABASE_URL
        sync: false
      - key: GITHUB_TOKEN
        sync: false
      - key: SESSION_SECRET
        generateValue: true
      - key: SUPABASE_URL
        sync: false
      - key: SUPABASE_KEY
        sync: false
```

Step 4: Deploy to Render

1. Push your code to a GitHub repository
2. Log in to your Render account
3. Click **New** and select **Web Service**
4. Link your GitHub repository
5. Fill in the service details:
    - **Name**: mindfulai (or your preferred name)

    - **Environment**: Python
    - **Build Command**: `pip install -r requirements.txt`
    - **Start Command**: `gunicorn --bind 0.0.0.0:$PORT --workers=1 main:app`
6. Add the following environment variables:
    - `DATABASE_URL`: Your Supabase PostgreSQL connection string
    - `GITHUB_TOKEN`: Your GitHub personal access token with "models:read" permission
    - `SESSION_SECRET`: A strong random string for session security
    - `SUPABASE_URL`: Your Supabase project URL (from Project Settings)
    - `SUPABASE_KEY`: Your Supabase service_role key (from Project Settings → API)
7. Click **Create Web Service**

Render will automatically build and deploy your application. The service will be available at a URL like `https://mindfulai.onrender.com`.

Verifying the Deployment

1. Once deployed, visit your Render app URL
2. Create a new account using the registration page
3. Log in and test the chatbot functionality
4. Verify that:
    - User registration and login work correctly
    - The chatbot responds to mental health queries
    - The conversation flow follows the structured approach (problem → conversation → root cause → solutions)
    - The UI is responsive on both mobile and desktop

Troubleshooting

- If the application fails to start, check the Render logs for errors

- If database connection fails, verify your DATABASE_URL and Supabase credentials
- If the AI responses don't work, check that your GITHUB_TOKEN is valid and has the correct permissions

How the Application Works

1. **User Authentication**:
    - Users register with a unique user ID, name, email, and password
    - Passwords are securely hashed before storage in Supabase
    - Sessions are maintained using Flask's session management
2. **Conversation Flow**:
    - The chatbot first asks about the user's specific mental health issue
    - It follows up with targeted questions to understand contributing factors
    - Through careful questioning, it identifies the underlying root cause
    - Finally, it provides actionable solutions tailored to address that root cause
3. **Technical Architecture**:
    - **Frontend**: HTML/CSS/JS with responsive design
    - **Backend**: Flask Python application
    - **Database**: Supabase PostgreSQL (stores user data only)
    - **AI**: GitHub LLaMA model via API
    - **Deployment**: Render web service

Maintenance Recommendations

1. Regularly backup your Supabase database
2. Monitor your Render service for any performance issues
3. Keep your dependencies updated for security patches

4. Consider implementing a more sophisticated session management system for scaling

Resources

- Render Documentation
- Supabase Documentation
- Flask Documentation
- GitHub LLaMA Documentation