Mental Health Classifier - Project Documentation

1. Project Overview

This web application allows users to take a self-assessment survey for mental health. The backend uses Flask, and a custom logistic regression model helps determine the likelihood of symptoms such as depression.

2. Features

- User registration and login with hashed passwords
- Session persistence using Flask's session and `g` object
- Mental health survey with PHQ-9-style questions
- Survey result analysis using a logistic regression classifier
- Email notifications with results (if email is provided)
- User profile editing and history tracking

3. Technology Stack

- Flask
- SQLAlchemy (SQLite database)
- Flask-Mail (SMTP email)
- NumPy (for logistic regression math)
- Bootstrap 5 (UI/UX)
- Python's built-in `os` and `session` modules

4. Application Structure

- `app.py` – Main Flask app with routes and logic
- `templates/` – HTML templates for all views
- `static/` – Static CSS and JavaScript (if used)
- `site.db` – SQLite database for storing user and survey data

5. Agile User Stories

ID	As a...	I want to...	So that I can...
US01	Visitor	Register an account	access the survey and save my results
US02	Registered User	Log in to my profile	track and manage my survey results
US03	Registered User	Fill out the mental health survey	receive personalized AI feedback
US04	Registered User	View past survey results	reflect on my mental health over time
US05	Registered User	Edit my profile information	keep my data up to date
US06	Admin	View and manage user accounts	moderate and support user activity
US07	System	Send confirmation or results via email	keep users informed about their status
US08	Developer	Use a custom ML model	avoid using black-box libraries for learning purposes
US09	Developer	Structure code with Blueprints and modular files	keep the app scalable and maintainable
US10	User	Access the app on phone or desktop	use it comfortably anywhere thanks to responsive design

6. How Session Persistence Works

Instead of storing just the username in the session, the app stores the user's ID (`session['user_id']`).
Then, Flask's `g` object is used to load the user from the database before every request. This makes the user globally accessible in routes and templates, improving usability and maintainability.

7. Final Notes

This project is suitable for a school environment to demonstrate full-stack development, AI model integration,
and user interaction with persistent data storage and form-based workflows.