Helio - Smart Diet Planner

Helio is an intelligent, full-stack web application designed to help users take control of their health. It leverages a **Neural Network** to provide personalized diet and workout recommendations based on a user's unique health profile.

The application serves as a comprehensive wellness dashboard, allowing users to not only receive AI-driven plans but also track their progress, save their favorite meals, and get daily health tips. The entire frontend is built to be dynamic, fetching and displaying data from the Flask backend without requiring page reloads.

✨ Key Features

1. Core ML & Prediction

- **AI Diet Prediction:** Uses a Keras/TensorFlow Neural Network model to analyze a user's health metrics (age, BMI, BP, cholesterol, etc.) and predict a suitable diet class (e.g., Balanced, Low Carb, High Protein).
- **Dynamic Workout Generation:** Generates custom workout plans from a JSON database based on user's self-selected goal (Weight Loss, Muscle Building), fitness level (Beginner, Active), and available time (15-20 min, 30-40 min).

2. User & Profile Management

- **Secure Authentication:** Full user registration, login, and logout system.
- **Password Validation:** Enforces strong passwords on registration (min 5 characters, 1 number, 1 special character).
- **Password Reset:** Secure "Forgot Password" functionality that sends a timed reset link via email (using Flask-Mail).
- **Comprehensive User Profile:** Users can create and update their health profile with all metrics used for prediction.
- **Profile Completion Bar:** A dynamic progress bar on the profile page that shows users how complete their profile is.
- **Incomplete Profile Warning:** A non-blocking modal warns users to fill in optional fields for more accurate predictions.

3. Interactive Dashboard

- **Dynamic BMI Gauge:** A real-time ApexCharts gauge visualizes the user's current BMI status.
- **Daily Health Tip:** A card that fetches and displays a new random health tip from a JSON file every time the page loads.

4. Dynamic Meal Plan Module

- **One-Click Prediction:** Get a new, random meal plan from the predicted diet class without leaving the dashboard.
- **Smart Swap:** A "swap" icon next to each meal (Breakfast, Lunch, Dinner) allows users to get a new random option for that specific meal without changing the rest of the plan.
- **Save & View Plans:** Save generated diet plans to the user's account.
- **Action Buttons:**
  - **Share:** Share the plan via Copy to Clipboard, WhatsApp, or native device sharing.
  - **Print:** A "Print/Download" button formats the plan for printing.
  - **Order Food (Swiggy):** An "Order Food" button opens a Swiggy search for the specific meal.
  - **Order Groceries (Blinkit):** An "Order Groceries" button intelligently extracts ingredients and opens a Blinkit search.
  - **Find Stores:** A "Find Stores" button uses geolocation to open Google Maps with nearby grocery stores.

5. Workout Planner Module

- **Dedicated Planner Page:** A separate page to generate workout plans based on Goal, Level, and Duration.
- **Action Buttons:**
  - **Save & View Plans:** Save generated workout plans.
  - **Share:** Share the workout via Copy to Clipboard, WhatsApp, or native device sharing.
  - **Print:** A "Print/Download" button formats the plan for printing.
  - **"How-to" Links:** A YouTube icon next to each exercise opens a new tab with a search for a tutorial on that specific workout.

6. Saved Plans Page

- **Tabbed Interface:** A clean UI that separates "Saved Diets" and "Saved Workouts" into two tabs.
- **Delete Functionality:** Users can delete any saved diet or workout with a custom confirmation modal (no `alert()` boxes).

🛠️ Tech Stack

Backend

- **Framework:** Flask
- **Database:** MongoDB (with PyMongo)
- **Machine Learning:** Keras (TensorFlow) for the Neural Network, Scikit-learn (StandardScaler)

- **Authentication:** Flask-Login, Werkzeug (for hashing)
- **Email & Tokens:** Flask-Mail, itsdangerous (for password reset)
- **Core:** Python, JSON, NumPy

Frontend

- **Core:** HTML5, CSS3, JavaScript (ES6+)
- **Framework:** Bootstrap 5 (for layout, components, and icons)
- **Templating:** Jinja2
- **Visualization:** ApexCharts.js (for BMI gauge), Chart.js (for weight chart)

Project Structure

```
Helio-Smart-Diet-Planner/
├── app.py                  # Main Flask application
├── data/
│   ├── diet_plans.json
│   ├── workouts.json
│   └── health_tips.json
├── model/
│   ├── diet_model.keras    # The trained Neural Network
│   ├── scaler.pkl          # The pre-fitted StandardScaler
│   └── meal_encoder.pkl
├── static/
│   └── css/
│       └── main.css
├── templates/
│   ├── base.html
│   ├── index.html
│   ├── login.html
│   ├── register.html
│   ├── forgot_password.html
│   ├── reset_password.html
│   ├── dashboard.html
│   ├── profile.html
│   ├── workout_planner.html
│   └── saved_plans.html
├── .env                    # (Must be created locally)
├── requirements.txt
└── README.md
```

🚀 Setup & Installation

To run this project locally, follow these steps:

1. **Clone the repository:**

```
git clone [https://github.com/your-username/Helio-Smart-Diet-Planner.git](https://g
cd Helio-Smart-Diet-Planner
```

2. **Create and activate a virtual environment:**

```
# Windows
python -m venv venv
venv\Scripts\activate

# macOS/Linux
python3 -m venv venv
source venv/bin/activate
```

3. **Install the required packages:**

```
pip install -r requirements.txt
```

4. **Create your Environment File:** Create a file named `.env` in the root of the project and add your credentials. Use `.env.example` as a template.

```
FLASK_SECRET_KEY='your_super_secret_key_here'
MONGO_URI='your_mongodb_connection_string'

# For password reset emails
MAIL_SERVER='smtp.gmail.com'
MAIL_PORT=587
MAIL_USE_TLS=True
MAIL_USERNAME='your-email@gmail.com'
MAIL_PASSWORD='your-gmail-app-password'
```

5. **Run the application:**

```
python app.py
```

The application will be available at `http://127.0.0.1:5000`.

🔮 Future Enhancements

- **Dietary Preferences:** Implement the "veg" / "non-veg" / "eggetarian" filtering.
- **Nutritional Info:** Add calorie and macronutrient (protein, carbs, fat) data to each meal and display it.
- **Weight Goal Tracking:** Allow users to set a "goal weight" and show progress towards it.