**1. Components (Modules) of the Front-End and Their Relationships**

**Modules**

1. **Login and Authentication Module**
   * Handles user login via Google Authentication
   * Validates user credentials and creates a user session
2. **Dashboard Module**
   * Central hub displaying an overview of logged metrics, progress on goals, and daily reports
   * Presents visualizations like graphs and charts for health data trends
   * The Dashboard Module in the **HealthPal** app is a real-time, front-end interface that serves as the primary point of interaction for users after logging in. It is designed to display dynamically updated data without requiring explicit user requests to generate a dashboard view
   * The dashboard is displayed as a default home screen after the user logs into the app.
   * It fetches user data (health metrics, calorie logs, and goal progress) from the backend and automatically displays it in a clear, graphical format
   * Data updates seamlessly when user logs new metrics, or calorie information, which ensures the view remains up-to-date

**Backend and Data Flow of Dashboard**

* The app automatically queries the backend back end database after the user logs in to retrieve the latest data
* The user can manually refresh the dashboard trigging a new API call to update data

1. **Health Metrics Logging Module**
   * Interface for users to log data
   * Users input their daily health metrics via a simple text/dropdowns
2. **Calorie Tracker Module**
   * Fetches meal information via the Open Food Facts API.
   * Allows users to input consumed meals and calculates calorie intake
3. **AI Recommendation Module**
   * Displays AI-driven (Open AI) tips based on user health data (water intake reminders, sleep suggestions, food consumption)
   * Provide actionable, personalized tips to help users improve health and meet their goals

**How the Module Works**

* User’s health metrics, logged goals, calorie data, and historical trends are fetched for the SQL database
* Send user data as a structured input prompt to OpenAI GPT API for generating advanced recommendations
* GPT processes and displays the recommendation on the app’s dashboard or as a notification

**Dependencies**

* If the tip doesn't rely on real-time calorie data from the Open Food Facts API, GPT-4 can generate recommendations using existing user data.
* For calorie-related suggestions, the app fetches data from the Open Food Facts API first, stores it, and then sends the complete dataset to GPT-4.

1. **Settings and Reports Module**
   * Users can export their data, configure preferences, and view detailed health reports.

**Relationships**

* **Login and Authentication Module** interacts with the **Dashboard Module** to grant or deny access to the app.
* **Dashboard Module** connects with all other modules to display relevant data and recommendations.
* **Health Metrics Logging Module** transmits logged data to the backend for storage and retrieval.
* **Calorie Tracker Module** interacts with the Open Food Facts API and integrates the fetched data into the dashboard.
* **Settings and Reports Module** retrieves data from the backend to generate reports and supports user configurations.

**2. APIs/Technology for Front-End**

**APIs and Technologies**

1. **Open Food Facts API**
   * Fetches detailed nutritional information about meals.
2. **Google Authentication API**
   * Secure login and logout functionality.
3. **OpenAI GPT**

* Tips recommendation

1. **Graphing and Visualization Library**
   * Visualizes user data trends using charts and graphs.

**3. Database System for Back-End**

**Database Structure**

* **Database Type**: MySQL
* **Tables**:
  1. **Users**: Stores user information ( user id, name, email, and login credentials)
  2. **Health Metrics**: Logs user metrics (user id, date, weight, blood pressure, heart rate, temperature, breathing rate, daily steps, sleep hours)
  3. **Calorie Logs**: Tracks calorie intake (user id, date, food item, calories)

**Implementation**

1. **Input Data**:
   * User logs health metrics or calorie data via the front-end
   * Data is transmitted to the backend
2. **Database Insertion**:
   * Backend receives data and inserts it into the corresponding tables in the MySQL database
3. **Output Data**:
   * Data queries retrieve health metrics, calorie logs, and AI recommendations for display on the front-end
4. **External Data Handling**:
   * Open Food Facts API responses are processed by the backend and stored in the **Calorie Logs** table for future reference

**4. Data Flow Diagram**

**Data Flow Diagram**

1. **Components**:
   * Front-end Modules (e.g., Dashboard, Health Metrics Logging, Calorie Tracker)
   * Backend API Server
   * MySQL Database
2. **Process**:
   * **User Inputs Data**: Front-end sends health metrics/calorie information to the backend
   * **Backend Processing**: Backend validates and stores the data in the database
   * **Data Retrieval**: Backend queries data for dashboard visualization and sends responses to the front-end
   * **API Interaction**: Backend fetches meal data from the Open Food Facts API