HEALTH INFORMATION SYSTEM

APPROACH, DESIGN, AND SOLUTION

PROBLEM STATEMENT

- Doctors need a system to manage clients and health programs efficiently.
- Key requirements:
  - Create health programs.
  - Register and manage clients.
  - Enroll clients in programs.
  - Search and view client profiles.
  - Expose client data via an API for integration with other systems.

Approach

- API-First Design: Define all endpoints before starting backend logic.
- Folder Structure: Organized code into routes, models, services, and utils.
- Modular Code: Each feature separated into service functions for maintainability.
- RESTful API: Clean and predictable endpoints.
- Testing: Used Postman to manually test and verify API responses.

System Design

Architecture:

- Flask-based backend with RESTful API endpoints.
- implement file based storage

Key Components:

- routes.py: Handles API endpoints.
- models.py: Defines data structures for clients and programs.
- storage.py: File based storage.

API Endpoints:

- CRUD operations for clients and programs.
- Search and enrollment functionalities.

Solution

- **Features:**
  - Create health programs.
  - Register clients with unique IDs.
  - Enroll clients in one or more programs.
  - Search for clients by name.
  - View and delete client profiles.
- **Example API Usage:**
  - POST /programs: Create a program.
  - POST /clients: Register a client.
  - PUT /clients/<id>/enroll: Enroll a client in programs.
  - GET /clients/search: Search for clients.

Implementation

- **Technologies Used:**
  Python, Flask, and RESTful API principles.
- **Setup:**
  Install dependencies: pip install -r requirements.txt.
  Run the application: python run.py.
- **Testing:**
  Use Postman for testing the endpoints.