PROBLEM STATEMENT 2

Clinic Management System (FastAPI)

Project Name

clinic

Scenario Description

A clinic wants to build a **Clinic Management System** to manage:

1. Patient records
2. Doctor appointment slots

The system must be implemented using **FastAPI**, must store data **in memory using HashMaps**, and **must not use Pydantic models or databases**.

The application must support **CRUD operations** and validate input using **Path, Query, and Body parameters**.

Mandatory Project Structure

clinic/

│

├── main.py

├── patients.py

└── appointments.py

main.py Requirements

- FastAPI app instance must be created **only in main.py**
- Routers must be included using include_router()
- APIs must be accessed only via main.py
- Application must be started using:
- uvicorn main:app --reload

PATIENTS MODULE

Base Route

/patients

CRUD Functional Requirements

1. **Create Patient (POST)**

- patient_id → Path (int, > 0)
- name → Body (min 3 characters)
- age → Body (≥ 0)
- contact_number → Body (10 digits)
- gender → Query (male, female, other)

2. **Read Patient (GET)**
3. **Update Patient (PUT)**
4. **Delete Patient (DELETE)**

Validation Rules

- Duplicate patient_id not allowed
- Gender must be validated via query parameter
- Contact number must be exactly 10 digits
- Proper error messages required

APPOINTMENTS MODULE

Base Route

/appointments

CRUD Functional Requirements

1. **Create Appointment (POST)**
   - appointment_id → Path (int, > 0)
   - doctor_name → Body (min 3 characters)
   - available_slots → Body (≥ 1)
   - specialization → Query (general, cardiology, orthopedics)
2. **Read Appointment (GET)**
3. **Update Slots (PUT)**
4. **Delete Appointment (DELETE)**

Validation Rules

- Available slots must not go below zero
- Specialization must be validated via query parameter

PROBLEM STATEMENT 3

Bank Management System (FastAPI)

Project Name

bank

Scenario Description

A bank wants to develop a **Bank Management System** to manage:

1. Customer accounts
2. Bank transactions

The application must be developed using **FastAPI**, must store all data **in memory using HashMaps**, and **must not use Pydantic models or databases.**

The system must support **CRUD operations** and validate data using **Path, Query, and Body parameters.**

Mandatory Project Structure

bank/

│

├── main.py

├── customers.py

└── transactions.py

CUSTOMERS MODULE

Base Route

/customers

CRUD Functional Requirements

1. **Create Customer (POST)**
    - customer_id → Path (int, > 0)
    - name → Body (min 3 characters)
    - balance → Body (≥ 0)
    - email → Body (valid format)
    - account_type → Query (savings, current)
2. **Read Customer (GET)**

3. **Update Customer (PUT)**
4. **Delete Customer (DELETE)**

TRANSACTIONS MODULE

Base Route

/transactions

CRUD Functional Requirements

1. **Create Transaction (POST)**
    - transaction_id → Path (int, > 0)
    - customer_id → Query (must exist)
    - transaction_type → Query (deposit, withdraw)
    - amount → Body (> 0)
2. **Read Transaction (GET)**
3. **Delete Transaction (DELETE)**

Validation Rules

- Customer must exist
- Transaction type must be validated using query parameter
- Withdrawal amount must not exceed balance
- Proper error handling required

Data Storage Requirement (ALL PROJECTS)

- All data must be stored using **in-memory HashMaps**
- No database or file system allowed