GoLang Microservices Assignment – Doctor Appointment Booking System

Objective:

Design and implement a secure Doctor Appointment Booking System using GoLang and a microservices architecture**, with JWT-based authentication, proper documentation, and test coverage. A frontend interface is optional but encouraged.

Functional Requirements:

1. **JWT-Based User Authentication**
    - Users (patients and doctors) must register and log in using a secure mechanism.
    - Upon login, issue a **JWT token** to authenticate subsequent requests.
    - Role-based access control:
        - Patients: Can book/view/edit appointments.
        - Doctors: Can view their appointments (optional).
2. **Doctor Listing**
    - Patients can view a list of available doctors.
3. **Slot Availability**
    - Patients can view available time slots for a selected doctor.
4. **Appointment Booking**
    - Patients can book appointments with doctors based on available slots.
5. **Appointment Management**
    - Patients can view appointments with status (Upcoming, Completed, Cancelled).
    - Patients can **edit**, **reschedule**, or **cancel** appointments.

Architecture Requirements:

- Use **GoLang** with a **microservice-based architecture**.
- Minimum required services:
    - **Auth Service:** Handles registration, login, and JWT token management.
    - **User Service:** Manages user profiles (doctors/patients).
    - **Schedule Service:** Manages slot availability per doctor.
    - **Appointment Service:** Manages bookings and appointment operations.
- Secure all service-to-service communication using **JWT token verification** or internal service credentials.

Non-Functional Requirements:

- Clear and modular **code structure**
- **Documentation**:
    - Project README with setup steps
    - Swagger/OpenAPI for each service
- **Test Coverage**:
    - Unit and integration tests with **≥80% coverage**
    - Use tools like go test, cover, or ginkgo
- **Dockerized Deployment**:
    - Dockerfile for each service
    - Docker Compose setup for local testing

Bonus (Good to Have):

- A basic **web frontend** (e.g., React/Next.js):
  - Login form (token-based auth)
  - Doctor search
  - Slot selection
  - Appointment management UI

Deliverables:

- A GitHub repository containing:
  - Video Recordings for all the implementations and features
  - Source code for all microservices
  - README with setup, architecture diagram, and usage instructions
  - Swagger documentation or Postman collection
  - Test coverage reports (HTML or CLI output)
  - Sample .env and docker-compose.yml