Healthcare Appointment Scheduling System

1

Analyst: Could you explain the main purpose of the system?

Client: We want something where patients can easily make appointments, and doctors can know who's coming and when.

Analyst (interpretation): The **core purpose** is to provide a **Healthcare Appointment System** that digitizes the scheduling process.

Extracted Requirements: Understood the problem domain.

- Functional: Patients should be able to book appointments online.
- Non-functional: System should be user-friendly and accessible to patients of all ages.

2

Analyst: What features do you expect for patients?

Client: Patients should find the right doctor, choose a convenient time, and be able to change or cancel if something comes up.

Analyst (interpretation): This means we need features for **searching doctors (by specialty, location, availability)**, **booking**, **rescheduling**, and **cancellation**.

Extracted Requirements:

Functional: Patients can **register, search, book, reschedule, cancel appointments**.

3

Analyst: And what about doctors—what should they be able to do?

Client: Doctors should keep track of their schedules, avoid double-booking, and adjust their working hours.

Analyst (interpretation): So we'll create a **doctor module** where they can **update availability** and **view/manage appointments**.

Extracted Requirements:

Functional: Doctors can **manage schedules and availability**.

Non-functional: **Reliability:** No double-booking, timely reminders.

4

Analyst: How do you see this system organized?

Client: Well, patients do their part, doctors do theirs, and someone needs to make sure everything runs smoothly.

Analyst (interpretation): That's **modularization**. We'll split the system into independent **modules** (patient, doctor, scheduling, notifications), each handling a clear role. This makes the system easier to build, test, and maintain.

Extracted Requirements:

Non-functional:

- **Maintainability: modularization** ensure the system can grow without breaking.
- **Flexibility**

5
Analyst: What should the system look and feel like?
Client: It should be easy, even for older people. And many will use it on their phones.

Analyst (interpretation): So we need **usability** (simple UI), **mobile-first design**, and accessibility.

Extracted Requirements:

Non-functional:

- **Usability:** Simple, mobile-friendly, accessible.
- **Availability:** 24/7 uptime.

6
Analyst: Can you give me an example of how you think tasks should be broken down?
Client: Well, booking an appointment has steps: first find a doctor, then pick a time, then confirm.

Analyst (interpretation): Breaking big processes into smaller tasks. Each function (search, select, confirm) becomes a part under the scheduling module.

Extracted Requirements: Functional decomposition: booking process split into steps.

7
Analyst: Should we think in terms of people using the system, like doctors, patients, and admins?
Client: Yes, each type of user does something different.

Analyst (interpretation): We'll model the system around objects/entities like **Patient, Doctor, Appointment, Admin**, each with its own responsibilities and data.

Extracted Requirements: Object-oriented decomposition: Patient, Doctor, Appointment as objects.

8
Analyst: How many people might use it at the same time?
Client: We expect thousands of patients in the city, maybe more in the future.

Analyst (interpretation): The system must handle **high concurrency** (10,000+ users), so the architecture should be **scalable**.

Extracted Requirements:

Non-functional:

- **Scalability:** Should evolve to microservices.
- **Performance:** <2s response, 10,000+ concurrent users.

9
Analyst: How important is security for you and do you want every user to see all the system details, or only the parts they need?
Client: Very important. We don't want patient details leaking, A patient shouldn't see doctor's notes, and doctors don't need admin settings.

Analyst (interpretation): So we'll implement **data encryption**, and **secured storage.** Also includes **abstraction and encapsulation.**

Extracted Requirements:

Non-functional:

- **Security:** Encryption of user credentials.
- **Authentication: role-based access control** (patient/doctor/admin).

10
Analyst: Do you need reminders or communication features?
Client: Yes, patients often forget appointments. It should send reminders by text or email.

Analyst (interpretation): That means we need a **notification service** integrated with email/SMS.

Extracted Requirements:

Functional: System sends **reminders via email/SMS**.

Non-functional: **Availability:** 24/7 uptime.

11
Analyst: Would you like the system to connect to other hospital systems in the future?
Client: Maybe later. For example, linking it to medical records would save time.

Analyst (interpretation): So we'll plan for **domain modeling** with entities like *Patient, Doctor, Appointment, Notification*. Later we can extend the model to include **EHR integration**.

Extracted Requirements:

Functional: System can **integrate with hospital EHRs** in the future.

12
Analyst: Do you prefer a simple design at first or something that can grow over time?
Client: Start simple, but make sure it won't collapse if we expand.

Analyst (interpretation): Start with a layered architecture (UI, business logic, database). Later evolve into microservices architecture.

Extracted Requirements: Architectural Styles: Layered architecture.

Interview Summary (Main Points)

- Patients can search doctors and book/reschedule/cancel appointments.
- Doctors manage availability and avoid double bookings.
- System sends reminders/notifications.
- Admin oversees users and system settings.
- Security is critical: protect patient data, restrict access by role.
- System should be simple, mobile-friendly, and scalable.
- Future option: connect with hospital records (EHR).

Manuel Youssef Haik Kevorkian	13006600
Omar Marmoush	14001357
Chantal Sherif	13007034
Youstina Raouf	13001755
Ahmed Mohamed	14002120
