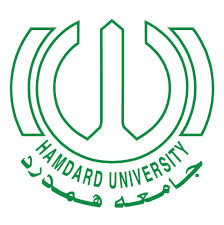
Hamdard University

Department of Computing

Final Year Project



**E-Health**

**(FYP-012/FL24)**

**Software Requirements Specifications**

Submitted by

Muaaz Ahmed Baig (1726-2021)

S.M Hamza Sohail (2271-2021)

Syed Fazalullah (2173-2021)

Supervisor

Sir Mohsin Raza Khan

**Fall 2021**

**Document Sign off Sheet**

### **Document Information**

|  |  |
| --- | --- |
| **Project Title** | E-Health |
| **Project Code** | FYP-012/FL24 |
| **Document Name** | Software Requirements Specifications |
| **Document Version** | 1.0 |
| **Document Identifier** | FYP-012/FL24-SRS |
| **Document Status** | Final Draft |
| **Author(s)** | Muaaz Ahmed, Syed Fazal, S.M Hamza Sohail |
| **Approver(s)** | Sir Mohsin Raza Khan |
| **Issue Date** | 13-01-2025 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Role** | **Signature** | **Date** |
| Muaaz Ahmed Baig | Team Lead |  |  |
| S.M Hamza Sohail | Team Member 2 |  |  |
| Syed Fazalullah | Team Member 3 |  |  |
| Sir Mohsin Raza Khan | Supervisor |  |  |
| --- | Co-Supervisor |  |  |
| Sir Suleman Naz | Project Coordinator |  |  |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 10-01-2025 | 0.1 | First Draft | Muaaz, Hamza |
| 12-01-2025 | 0.2 | Revised Version | Sir Mohsin Khan |
| 16-01-2025 | 0.1 | Final Report | Muaaz, Hamza |
|  |  |  |  |

**Definition of Terms, Acronyms, and Abbreviations**

|  |  |
| --- | --- |
| **Term** | **Description** |
| **Scrum** | It is used to agile methodology in this project |
| **Firebase** | Firebase can help the hosting the mobile and web application |
| **Frontend** | It is part of the User Interface which shows in the screen using the HTML, CSS, JavaScript and the modern language that is React Native. |
| **Backend** | The back side of the application that can not shows the user. It can store the user data and help the functionality. |
| **MongoDB** | It is flexible and scalable data storage in the database system. |
|  |  |
|  |  |
|  |  |

# **Table of Contents**

[**1.** Introduction 8](#_Toc4677804)

[1.1 Purpose of Document 8](#_Toc4677805)

[1.2 Intended Audience 8](#_Toc4677806)

[1.3 Abbreviations 8](#_Toc4677807)

[**2.** Overall System Description 9](#_Toc4677808)

[2.1 Project Background 9](#_Toc4677809)

[2.2 Problem Statement 9](#_Toc4677810)

[2.3 Project Scope 9](#_Toc4677811)

[2.4 Not In Scope 9](#_Toc4677812)

[2.5 Project Objectives 9](#_Toc4677813)

[2.6 Stakeholders & Affected Groups 9](#_Toc4677814)

[2.7 Operating Environment 9](#_Toc4677815)

[2.8 System Constraints 9](#_Toc4677816)

[2.9 Assumptions & Dependencies 9](#_Toc4677817)

[**3.** External Interface Requirements 10](#_Toc4677818)

[3.1 Hardware Interfaces 10](#_Toc4677819)

[3.2 Software Interfaces 10](#_Toc4677820)

[3.3 Communications Interfaces 10](#_Toc4677821)

[**4.** System Functions / Functional Requirements 11](#_Toc4677822)

[4.1 System Functions 11](#_Toc4677823)

[4.2 Use Cases 12](#_Toc4677824)

[4.2.1 List of Actors 12](#_Toc4677825)

[4.2.2 List of Use Cases 12](#_Toc4677826)

[4.2.3 Use Case Diagram 13](#_Toc4677827)

[4.2.4 Description of Use Cases 13](#_Toc4677828)

[**5.** Non - Functional Requirements 16](#_Toc4677829)

[5.1 Performance Requirements 16](#_Toc4677830)

[5.2 Safety Requirements 16](#_Toc4677831)

[5.3 Security Requirements 16](#_Toc4677832)

[5.4 Reliability Requirements 16](#_Toc4677833)

[5.5 Usability Requirements 16](#_Toc4677834)

[5.6 Supportability Requirements 16](#_Toc4677835)

[5.7 User Documentation 16](#_Toc4677836)

[**6.** References 17](#_Toc4677837)

# **Introduction**

Through this E–Health System, Patients can search clinic or doctors based on specialization, name, or availability. Patients can receive a token number online for their appointment. Displays the current token number being served and an estimated wait time. A laboratory management system will work in the same way as a clinic management system. When scheduling a test, you can see a list of laboratories, their prices, and any other relevant details. This application will provide online medical services and patients can buy their prescribed medicines.

## **Purpose of Document**

The Software Requirements Specification (SRS) document for the Vulnerability Assessment and Penetration Testing (VAPT) application serves as a comprehensive guide for its development. Created using Scrum methodology, this document details the application's design, architecture, and functionality. It acts as a blueprint for the development team, clearly defining the technical requirements and specifications crucial for successful implementation. By establishing a shared understanding of user needs, the SRS provides clear direction for development, facilitates progress tracking, and helps ensure the project stays on course.

## **Intended Audience**

* **Developers:** Those responsible for creating and maintaining the software.
* **Testers:** Individuals who verify the software's functionality and quality.
* **End-users:** The people who will ultimately use the software.
* **Project Managers:** Those overseeing the project's progress and ensuring its success.
* **Stakeholders:** Anyone with an interest in the project's outcome, such as investors or clients.

# Overall System Description

## **Project Background**

The current management of clinics, laboratories, and medical stores often relies on manual processes, leading to inefficiencies, errors, and delays in patient care and administrative tasks. With the increasing patient load and the complexity of healthcare management, the limitations of these manual systems have become more evident. These limitations include difficulties in managing patient records, scheduling appointments, tracking lab results, managing inventory in medical stores, and coordinating communication between doctors, patients, and labs. The lack of automation can result in misplaced or lost records, scheduling conflicts, delays in diagnosis and treatment, and inefficient inventory management. This can negatively impact patient experience and the overall efficiency of healthcare delivery. To address these issues, there is a clear need for an automated E-Health system that streamlines patient data management, appointment scheduling, lab result tracking, medical store inventory management, and communication. The proposed E-Health application will improve accuracy, reduce administrative burdens on healthcare staff, enhance efficiency in healthcare delivery, and provide a better experience for both patients and healthcare providers, while supporting the growing demand for accessible and efficient healthcare services.

## **Problem Statement**

The problem statement for the application is made are that when a patient go to the clinic so he has to be waited so long for the doctor and also have to take the appointment by going to that clinic. As same for the laboratory the patient have to wait for long to make conduct his test. And for medical store people can’t get the medicine from their nearby medical store.

## **Project Scope**

* Through this E – Health System, Patients can search clinic or doctors based on specialization, name, or availability.
* Patients can receive a token number online for their appointment. Displays the current token number being served and an estimated wait time.
* A laboratory management system will work in the same way as a clinic management system.
* When scheduling a test, you can see a list of laboratories, their prices, and any other relevant details.
* This application will provide online service of medical store and patients can buy their prescribed medicines.

## **Not In Scope**

The scope of this E-Health project does not include manual processes beyond the initial migration of existing patient data, medical records, and inventory data into the system. Additionally, the system will not integrate with hospital-wide or national healthcare platforms or portals other than the core E-Health application itself. Post-treatment management, such as long-term care tracking or public health surveillance integration, is also outside the project scope. Furthermore, advanced features such as AI-based diagnostic tools, predictive analytics for disease outbreaks, or machine learning for personalized medicine recommendations are excluded from the current development. This project focuses on core clinic, laboratory, and medical store management functionalities.

## **Project Objectives**

* The primary objectives of the E-Health project are to design a secure and user-friendly messaging system for efficient communication, develop an intuitive appointment scheduling feature to reduce wait times and improve patient engagement, and create a secure and easily accessible platform for patients to view their medical records and test results.
* Additionally, the project aims to improve patient engagement by providing educational resources and health tips, ensure the application meets industry standards for data security and privacy, design an intuitive and user-friendly interface for both doctors and patients, and develop a scalable and flexible application for future updates and feature additions. The overall goal is to create a comprehensive and user-friendly application that improves healthcare outcomes, enhances patient engagement, and streamlines communication between doctors and patients.

## **Stakeholders & Affected Groups**

 **Patients:** The primary users of the E-Health system who will benefit from easier appointment scheduling, access to medical records, and improved communication with healthcare providers.

 **Doctors/Healthcare Providers:** Medical professionals who will experience reduced administrative workload, improved access to patient data, and streamlined workflows for consultations, diagnoses, and treatment plans.

 **Laboratory Staff:** Personnel who manage laboratory operations and will benefit from automated test requests, result entry, and reporting.

 **Medical Store Staff:** Employees responsible for managing inventory and dispensing medications, who will experience streamlined stock management, automated ordering, and reduced errors.

 **Administrative Staff (Clinic):** Employees who manage the administrative aspects of the clinic and will experience reduced manual workload related to patient registration, billing, and reporting.

 **Project Managers:** Individuals overseeing the project’s execution and ensuring its alignment with timelines and objectives.

 **Security Analysts:** Personnel responsible for ensuring the safe handling and storage of sensitive patient data and compliance with relevant regulations.

## **Operating Environment**

The E-Health application will operate as a secure web-based system accessible via both desktop and mobile devices, ensuring accessibility for patients and healthcare providers regardless of location. The environment is designed for high availability and reliability, crucial for continuous healthcare operations. Robust security measures will be implemented to safeguard sensitive patient health information and ensure compliance with relevant regulations. The backend will leverage a robust and scalable database solution such as MongoDB for efficient and secure data storage, including patient records, lab results, and inventory data. The frontend will utilize a modern framework like React Native with a responsive CSS framework like Tailwind CSS or Material UI to provide an intuitive and user-friendly interface across different devices. The system will be hosted on secure cloud infrastructure to ensure scalability, reliability, and disaster recovery, accommodating fluctuating patient volumes and evolving healthcare needs. This cloud-based architecture also allows for easier maintenance and updates.

## **System Constraints**

**System Constraints:**

* **Time Constraints:** The project must be completed within the academic deadlines set for final year projects, imposing a strict timeframe for development, testing, and deployment.
* **Budget Constraints:** Limited funding restricts the purchase of additional hardware, premium software services, or extensive third-party integrations. This necessitates efficient use of available resources and potentially the use of open-source or free alternatives where appropriate.
* **Regulatory and Compliance Constraints:** The system must adhere to relevant healthcare regulations and standards, such as HIPAA, GDPR, and other local data privacy laws. These regulations may impose limitations on data storage, access control, security measures, and system functionality.
* **Resource Constraints:** The project is developed by a small team with specific skillsets. This limits the scope and complexity of features that can be implemented within the given timeframe and budget. It also requires careful task management and efficient utilization of team members' expertise.
* **Data Migration Constraints:** Migrating existing patient data, medical records, and inventory data from potentially disparate systems into the new E-Health application can be a complex and time-consuming process, posing a constraint on project timelines and requiring careful data validation and cleansing.

## **Assumptions & Dependencies**

The E-Health project assumes reliable internet connectivity and basic digital literacy for all users, availability of necessary hardware, accurate data entry, active participation of healthcare staff, and access to relevant APIs. The project's success depends on reliable cloud hosting, successful data migration from existing systems, stable third-party integrations (payment gateways, SMS gateways, lab equipment interfaces), active stakeholder cooperation, availability of development tools and frameworks, and adherence to healthcare regulations like HIPAA and GDPR.

# External Interface Requirements

**3.1 Hardware Interfaces**

The E-Health application is designed to be accessible on a variety of devices, including desktops, laptops, tablets, and smartphones, ensuring broad accessibility for patients and staff. These devices should be capable of running modern web browsers and/or native mobile applications (built with React Native).

For server-side infrastructure, robust servers are required to handle data storage, processing, and application logic. These servers should have sufficient processing power, memory, and storage capacity to handle anticipated user loads and data volumes. Cloud-based solutions are recommended for scalability, reliability, and high availability. Specific hardware requirements will depend on the chosen cloud provider and anticipated usage. Backup and disaster recovery mechanisms, including offsite backups, will be implemented to ensure data security and business continuity.

* **Client Devices:** Standard desktops, laptops, tablets, and smartphones with up-to-date operating systems and browsers.
* **Servers:** Cloud-based virtual machines or dedicated servers with appropriate specifications for CPU, RAM, and storage.
* **Peripheral Devices (Optional):** Printers (for reports, prescriptions), barcode scanners (for inventory management in the medical store), and potentially specialized medical devices for data input (depending on the specific functionality of the application).

**3.2 Software Interfaces**

The E-Health application interfaces with several key software components:

1. **Database:** MongoDB will be used for persistent data storage, providing scalability and flexibility for managing patient records, medical data, and inventory information.
2. **Frontend Framework:** React Native will be used to develop cross-platform mobile applications (iOS and Android) and potentially web interfaces.
3. **Backend Framework:** A suitable backend framework (e.g., Node.js with Express.js) will handle API requests, business logic, and database interactions.
4. **APIs:** RESTful APIs will facilitate communication between the frontend and backend components.
5. **Third-Party Integrations (Potential):**
   * **Payment Gateways:** Integration with secure payment gateways for online payments (e.g., for consultations, medications).
   * **SMS/Email Services:** For appointment reminders, notifications, and other communications.
   * **Laboratory Information Systems (LIS):** If integrating with external labs, APIs or other interfaces may be required.
   * **Electronic Health Record (EHR) Systems (Potential):** For data exchange with other healthcare providers.

**3.3 Communications Interfaces**

The E-Health application relies on secure communication protocols for data transmission and interaction:

* **Network Connectivity:** Reliable internet access is essential for both client devices and servers.
* **HTTPS:** All communication between clients and servers will use HTTPS to ensure data encryption and security.
* **APIs:** RESTful APIs will be used for communication between different software components.
* **Communication Protocols:** Standard internet protocols (TCP/IP, HTTP/HTTPS) will be used.
* **Security Measures:** Firewalls, intrusion detection/prevention systems, and regular security updates will be implemented to protect against unauthorized access and cyber threats.

This revised version focuses specifically on the needs of an E-Health application, including considerations for patient data security, integration with other healthcare systems (potential), and the use of React Native for mobile development. It also emphasizes the importance of cloud-based infrastructure for scalability and reliability.

# System Functions / Functional Requirements

**4.1 System Functions**

The E-Health application is designed to streamline clinic/laboratory operations, manage patient information, and facilitate efficient healthcare delivery.

1. **Patient Functions:**
   * **Registration and Profile Management:** Patients can create accounts, manage personal information, and update contact details.
   * **Appointment Scheduling:** Patients can book, reschedule, and cancel appointments with doctors or for lab tests.
   * **Medical Record Access:** Patients can view their medical history, lab results, prescriptions, and other relevant medical information (with appropriate security and privacy measures).
   * **Token Management:** Patients can view their token number and estimated waiting time.
   * **Online Payments:** Patients can make online payments for consultations, lab tests, or medications.
2. **Clinic/Laboratory Staff Functions:**
   * **Patient Management:** Staff can register new patients, update patient information, and manage medical records.
   * **Appointment Management:** Staff can schedule appointments, manage doctor schedules, and track patient visits.
   * **Lab Test Management:** Staff can order lab tests, record results, and generate reports.
   * **Token Management:** Staff can generate and manage token numbers for patients.
   * **Billing and Invoicing:** Staff can generate invoices, process payments, and manage billing records.
   * **Inventory Management (Medical Store):** Staff can manage drug inventory, track stock levels, and generate reports.
3. **Administrative Functions:**
   * **User Management:** Administrators can manage user accounts, roles, and permissions.
   * **Reporting and Analytics:** Administrators can generate reports on various aspects of the system, such as patient demographics, appointment statistics, and financial data.
   * **System Configuration:** Administrators can configure system settings, such as appointment schedules, lab test prices, and user roles.
4. **Integration Functions:**
   * **Payment Gateway Integration:** Secure integration with payment gateways for online transactions.
   * **SMS/Email Integration:** Automated appointment reminders, notifications, and other communications.
   * **Potential Integration with External Systems:** Future integration with Laboratory Information Systems (LIS) or Electronic Health Record (EHR) systems.

**4.2 Performance Requirements**

* The system should be able to handle a reasonable number of concurrent users (to be defined based on expected usage).
* Key operations, such as appointment booking and lab result retrieval, should complete within a reasonable timeframe (e.g., within a few seconds).
* The system should be responsive and provide a smooth user experience.

**4.3 Design Constraints**

* **Data Privacy and Security:** The system must comply with relevant data privacy regulations (e.g., HIPAA or local regulations).
* **Usability and Accessibility:** The system should be user-friendly and accessible to users with disabilities.
* **Platform Compatibility:** The system should be compatible with various devices and browsers.

**4.4 Technology Stack**

* **Frontend:** React Native (for mobile and potentially web).
* **Backend:** Node.js with Express.js (or similar).
* **Database:** MongoDB.

**4.5 Interface Requirements**

* **User Interfaces:** Intuitive user interfaces for patients, clinic/lab staff, and administrators.
* **APIs:** RESTful APIs for communication between the frontend and backend.
* **Security:** Secure authentication and authorization mechanisms.

This version is much more focused on the functionalities expected in an E-Health application, covering patient interactions, staff workflows, administrative tasks, and critical aspects like data privacy and security. It also reflects the technology stack you've chosen.

**System Function Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Functions** | **Category** | **Attribute** | **Details & Boundary Constraints** |
| **E1.1** | Display Patient Dashboard | Evident | Interface Availability | Ensure the patient dashboard is accessible after login. |
| **E1.2** | Patient Registration | Evident | Input Validation | Validate required fields (e.g., name, DOB, contact info). |
| **E1.3** | Schedule Appointment | Evident | Dynamic Calendar | Display available appointment slots based on doctor/lab availability. |
| **E1.4** | Save Appointment Data | Hidden | Data Persistence | Ensure all appointment details are stored reliably in the database. |
| **E1.5** | View Medical History | Evident | Data Retrieval | Display patient's past medical records, lab results, and prescriptions. |
| **E1.6** | Manage Concurrent Users | Hidden | Data Retrieval | Support multiple concurrent users without performance degradation. |
| **E1.7** | Secure Login/Authentication | Evident | Security Policy | Enforce strong password rules and secure authentication. |
| **E1.8** | Generate Patient ID | Hidden | Unique ID Gen. | Ensure each patient has a unique identifier. |
| **E1.9** | Patient Profile Page | Evident | Form Fields | Includes personal details, contact information, and medical history summary. |
| **E1.10** | View Lab Results | Evident | Data Display | Display lab results in a clear and understandable format. |
| **E1.11** | Manage Inventory (Medical Store) | Evident | Inventory Control | Track stock levels, manage orders, and generate reports. |
| **E1.12** | Generate Token Number | Hidden | Token Generation | Generate unique token numbers for patients waiting for services. |
| **E1.13** | Display Token Status | Evident | Real-time Updates | Display current token number being served and estimated waiting time. |
| **E1.14** | Admin Dashboard | Evident | Admin Options | Provides access to system configuration, user management, and reporting. |
| **E1.15** | Generate Reports | Evident | Data Analysis | Generate reports on patient demographics, appointments, and financial data. |
| **E1.16** | Online Payment Processing | Evident | Payment Gateway | Securely process online payments for services. |
| **E1.17** | Manage Doctor/Staff Profiles | Evident | User Management | *Allow creation, editing, and management of doctor/staff profiles including specialization and availability.* |

**System Attributes/ Nonfunctional Requirements**

|  |  |  |
| --- | --- | --- |
| **System Attribute** | **System Details and Boundary Constraints** | **System Type** |
| **Response Time** | The system shall respond to user interactions (e.g., button clicks, form submissions, data retrieval) within 2 seconds under normal load conditions. During peak usage, the response time should not exceed 5 seconds. | Mandatory |
| **Concurrent User Load** | The system must be able to simultaneously handle at least 1,500 concurrent users without significant performance degradation. This includes supporting concurrent data access, updates, and user interactions. | Mandatory |
| **Interface Design** | The user interface (UI) shall be: <br> - **Intuitive and user-friendly:** Employing clear and consistent navigation, easily understandable labels, and minimal cognitive load. <br> - **Visually appealing:** Adhering to modern design principles and using appropriate branding elements. <br> - **Accessible:** Complying with WCAG guidelines to ensure usability by users with disabilities. | Mandatory |
| **Accessibility** | The system shall adhere to the following accessibility standards: <br> - **WCAG 2.1 Level AA:** Ensuring compatibility with assistive technologies (screen readers, keyboard navigation) and providing alternative text for images. | Mandatory |
| **Scalability** | The system architecture and infrastructure should be designed to accommodate a gradual increase in user load and data volume without requiring significant hardware upgrades or system redesigns. | Optional (Highly Recommended) |
| **Security** | The system shall implement appropriate security measures to protect sensitive user data and system integrity, including: <br> - **Data encryption:** Both in transit and at rest. <br> - **Secure authentication:** Using strong passwords and multi-factor authentication. <br> - **Regular security audits and penetration testing:** To identify and address vulnerabilities. | Mandatory |
| **Reliability** | The system shall exhibit high availability and minimize downtime. This includes: <br> - **Regular backups and disaster recovery procedures:** To ensure data integrity and system resilience. <br> - **Error handling and logging mechanisms:** To detect and address issues proactively. | Mandatory |
| **Maintainability** | The system should be easy to maintain and update, including: <br> - **Well-documented code and infrastructure:** To facilitate future development and troubleshooting. <br> - Modulardesign**:** To enable independent component upgrades and maintenance. | |  | | --- | | Optional (Highly Recommended) | |

**4.6 Use Cases**

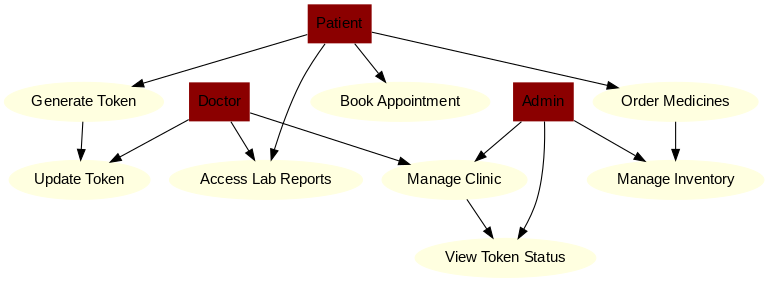
**4.6.1 List of Actors**

1. **Patient:** Schedules appointments, views medical records, pays bills, and communicates with healthcare providers.
2. **Doctor/Healthcare Provider:** Schedules appointments, views patient records, prescribes medications, and communicates with patients.
3. **Lab Technician:** Schedules and conducts tests, enters test results, and manages lab inventory.
4. **Pharmacist:** Fills prescriptions, manages medication inventory, and provides medication information to patients.
5. **Administrator:** Manages user roles, monitors system performance, configures system settings, and generates reports.
6. **IT Administrator:** Manages system infrastructure, performs maintenance tasks, ensures system security, and provides technical support.
7. **Payment Gateway:** Processes and verifies payments for consultations, tests, and medications.

**4.6.2 List of Use Cases**

1. **Schedule Appointment:** Patients can schedule appointments with doctors or book lab tests online.
2. **View Medical Records:** Patients can access their medical history, prescriptions, and test results.
3. **Make Payments:** Patients can pay for consultations, tests, and medications online through the integrated payment gateway.
4. **View Doctor Profiles:** Patients can search for and view doctor profiles, including specializations, qualifications, and patient reviews.
5. **Communicate with Healthcare Providers:** Patients can communicate with doctors through secure messaging within the application.
6. **Prescribe Medications:** Doctors can prescribe medications electronically and send them directly to the pharmacy.
7. **Enter Test Results:** Lab technicians can enter and record test results electronically.
8. **Manage Inventory:** Pharmacists can manage medication inventory, track stock levels, and order new supplies.
9. **Generate Reports:** Administrators can generate reports on patient demographics, appointment schedules, test results, and other key metrics.
10. **User Role Management:** Administrators can assign and manage user roles and permissions within the system.
11. **System Monitoring:** IT administrators can monitor system performance, identify and resolve technical issues, and ensure system security.

### Use Case Diagram



**4.1.3 Description of Use Cases**

|  |  |
| --- | --- |
| **Section** | **Main** |
| Name | |  | | --- | | Book Appointment |  |  | | --- | |  | |
| Actors | |  | | --- | | Patient, Admin |  |  | | --- | |  | |
| Purpose | |  | | --- | | Allow patients to book an appointment with a doctor. |  |  | | --- | |  | |
| Description | |  | | --- | | A patient logs into the system to schedule an appointment. The system displays available time slots, and the patient selects a preferred slot. The system confirms the booking and updates the schedule. |  |  | | --- | |  | |
| Cross Reference | |  | | --- | | Functions: R2.1, R2.2. Use Cases: Patient must complete the Login use case before accessing this feature. |  |  | | --- | |  | |
| |  | | --- | | Pre-Conditions |  |  | | --- | |  | | |  | | --- | | The patient must be registered in the system. The doctor’s availability must already be updated. |  |  | | --- | |  | |
| |  | | --- | | Successful Post-Conditions |  |  | | --- | |  | | |  | | --- | | Appointment is successfully booked, and the schedule is updated. |  |  | | --- | |  | |
| |  | | --- | | Failure Post-Conditions |  |  | | --- | |  | | No appointment is booked, and the system remains unchanged. |

**Typical Course of Events**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | **Actor Action** | | | **System Response** | | --- |  |  | | --- | |  | |
| |  | | --- | | 1. The patient logs into the system and navigates to the appointment booking page. |  |  | | --- | |  | | |  | | --- | | 2. Displays a list of available doctors and their schedules. |  |  | | --- | |  | |
| |  | | --- | | 3. The patient selects a doctor and an available time slot. |  |  | | --- | |  | | |  | | --- | | 4. Confirms the appointment and updates the system schedule. |  |  | | --- | |  | |
| |  | | --- | | 5. The patient receives a booking confirmation notification. |  |  | | --- | |  | |  |

**Alternative Course**

|  |  |
| --- | --- |
| **Step** | **Description** |
| **3.** | The patient selects a time slot that is no longer available. The system notifies the patient of the error and prompts them to choose a different slot. |

|  |  |
| --- | --- |
| **Section** | **Main** |
| Name | |  | | --- | | Order Medicines |  |  | | --- | |  | |
| Actors | |  | | --- | | Patient, Admin |  |  | | --- | |  | |
| Purpose | |  |  |  | | --- | --- | --- | | |  | | --- | | Allow patients to order medicines from the medical store. |  |  | | --- | |  | |  |  | | --- | |  | |
| Description | |  |  |  | | --- | --- | --- | | |  | | --- | | The patient searches for medicines in the online store and adds them to the cart. After confirming the order, the system processes payment and generates an order ID. |  |  | | --- | |  | |  |  | | --- | |  | |
| Cross Reference | |  |  |  | | --- | --- | --- | | |  | | --- | | Functions: R3.1, R3.2. Use Cases: Patient must complete the Login use case before placing an order. |  |  | | --- | |  | |  |  | | --- | |  | |
| |  | | --- | | Pre-Conditions |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | The patient must be registered and logged into the system. The medicines must be available in the inventory. |  |  | | --- | |  | |  |  | | --- | |  | |
| |  | | --- | | Successful Post-Conditions |  |  | | --- | |  | | |  | | --- | | The order is confirmed, and the inventory is updated. |  |  | | --- | |  | |
| |  | | --- | | Failure Post-Conditions |  |  | | --- | |  | | No order is placed, and the cart remains unchanged. |

**Typical Course of Events**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | **Actor Action** | | | **System Response** | | --- |  |  | | --- | |  | |
| |  |  |  | | --- | --- | --- | | |  | | --- | | 1. The patient logs into the system and searches for medicines. |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | | 2. Displays the list of available medicines. |  |  | | --- | |  | |
| |  |  |  | | --- | --- | --- | | |  | | --- | | 3. The patient adds medicines to the cart and proceeds to checkout. |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | | 4. Processes the order and generates an order ID. |  |  | | --- | |  | |
| |  |  |  | | --- | --- | --- | | |  | | --- | | 5. The patient receives an order confirmation notification. |  |  | | --- | |  | |  |  | | --- | |  | |  |

**Alternative Course**

|  |  |
| --- | --- |
| **Step** | **Description** |
| **3.** | The patient adds an unavailable medicine to the cart. The system notifies the patient and suggests alternatives. |

# **Non - Functional Requirements**

**5.1 Performance Requirements**  
• The system must handle up to 500 concurrent users without noticeable performance degradation.  
• Key operations, such as appointment booking, token generation, and payment processing, should complete within 3 seconds.  
• The database should support efficient queries for large medical records, including patient history and lab results, without significant delay.  
• Reports generation, such as daily sales or appointment summaries, should not exceed 5 seconds.

**5.2 Safety Requirements**  
• The system must prevent data loss during critical operations like booking appointments, generating tokens, and processing payments using robust fail-safe mechanisms.  
• Daily automated backups of patient records, doctor schedules, and inventory data should be performed to ensure recovery in case of system failure.  
• The system must maintain a log of errors, notify administrators of critical failures, and provide options to resume failed operations seamlessly.

**5.3 Security Requirements**  
• All data transmissions between users and the system must be encrypted using SSL/TLS protocols to prevent unauthorized access.  
• Secure login mechanisms must include hashed passwords, two-factor authentication, and secure session management.  
• Role-based access control (RBAC) must restrict users to only the data and functions relevant to their roles, such as admin, doctor, pharmacist, or patient.  
• The system must monitor and prevent unauthorized access attempts, including brute-force attacks and suspicious activity, by locking accounts temporarily after multiple failed login attempts.

**5.4 Reliability Requirements**  
• The system must maintain 99.9% uptime to support uninterrupted patient bookings, lab reports, and medical store transactions.  
• Critical features, such as appointment booking, payment processing, and token generation, must remain operational under peak load conditions.  
• The system should include automated failover mechanisms to ensure continued operation in case of server or hardware failure.

**5.5 Usability Requirements**  
• The user interface must be simple, intuitive, and responsive for all users, including doctors, patients, and admin staff, without prior technical knowledge.  
• The system must comply with Web Content Accessibility Guidelines (WCAG) to ensure usability for patients with disabilities.  
• Real-time form validation and error messages must guide users in completing tasks such as form submissions, payments, and appointment bookings.

**5.6 Supportability Requirements**  
• The system should allow for seamless updates, feature enhancements, and bug fixes without disrupting active user sessions.  
• A modular architecture must enable integration with future features like third-party health monitoring apps or external APIs for advanced diagnostics.  
• Detailed logs and monitoring tools should assist IT teams in identifying and resolving issues promptly.

**5.7 User Documentation**  
• A comprehensive user manual should guide patients, doctors, and admin staff through core functionalities such as booking appointments, managing schedules, and ordering medicines.  
• Admin-level documentation should include instructions for system setup, user role management, and troubleshooting common errors.  
• FAQs, video tutorials, and in-app help should be available to assist users in resolving common issues independently.

# References

* https://www.dvago.pk/
* https://www.cbc.ca/news/health/hospital-patients-may-worry-less-after- seeing-their medical-records
* https://www.cbc.ca/news/health/patients-gain-when-doctors-notes- are%20online
* https://doi.org/10.1145/2379776.2379779
* https://doi.org/10.1258/1357633001934113
* Online Health Records (2011) https://www.cbc.ca/news/canada/toronto/online-health-records-popular-with