|  |  |  |
| --- | --- | --- |
| **The Islamic University of Gaza**  **Faculty of Engineering**  **Computer Engineering Department** |  | الجامعة الإسلامية بغزة  كلية الهندسة  قسم هندسة الحاسوب |

**HealthFlow – Medical Point Management System**

**هيلث فلو – نظام إدارة نقطة طبية**

**By**

**Basil Mohammed Alserr 120200665**

**Yaseen ALHassan Abu Zeina 120200291**

**Anas Mohammed Abu Al-Tarabish 120200999**

**Momen Mohammed Issa 120200866**

**Hassan Mooneef Salama 120190478**

**Supervised by**

**Dr. Ahmed Mahdi**

**A Graduation Project Report  
Submitted in Partial Fulfillment of the Requirements  
for the Degree of Bachelor of Engineering**

**July 2025**

**Abstract**

Many healthcare institutions and medical points, particularly in resource-limited environments, continue to rely heavily on paper-based systems to manage critical daily operations such as patient registration, medication dispensing, and inventory tracking. This reliance often results in poor data accuracy, frequent human errors, and delays in both administrative and medical decision-making. In some cases, data is manually recorded throughout the day and then entered into a digital system by a data entry clerk at the end of the day, leading to redundant work and limited data accessibility.

This project aims to address these challenges by developing a centralized, integrated digital system for managing medical points. The HealthFlow systemis initially implemented as a robust desktop application using Java and JavaFX, with a PostgreSQL database hosted on a cloud server to support secure and concurrent data access from multiple client workstations within and across facilities. It automates essential workflows—including patient intake, medication distribution, and inventory control—to ensure real-time data access and improve operational transparency.

Designed with scalability in mind, the HealthFlow system architecture supports future expansion into a fully integrated web-based interface that connects to the existing cloud database without requiring a complete HealthFlow system rebuild. By bridging the gap between manual and digital operations, the project provides a practical and efficient solution to enhance service delivery and optimize performance in targeted healthcare facilities.

**Table of Contents**

[**Chapter 1** 4](#_Toc203316247)

[**Introduction** 4](#_Toc203316248)

[**1.1 General Background** 4](#_Toc203316249)

[**1.2 Problem Statement** 5](#_Toc203316250)

[**1.3 Objectives** 6](#_Toc203316251)

[**1.3.1 Main Objective** 6](#_Toc203316252)

[**1.3.2 Sub Objectives** 6](#_Toc203316253)

[**1.4 Scope and Limitations** 7](#_Toc203316254)

[**1.4.1 Scope** 7](#_Toc203316255)

[**1.4.2 Limitations** 8](#_Toc203316256)

[**1.5 Project Significance & Benefits** 9](#_Toc203316257)

[**1.6 Document Structure** 10](#_Toc203316258)

[**1.7 Proposed Project Methodology & Work Plan** 11](#_Toc203316259)

[**1.7.1 Scrum Implementation** 11](#_Toc203316260)

[**1.7.2 Detailed Timeline** 11](#_Toc203316261)

[**Table 1.1 – Sprint Timeline and Key Deliverables for HealthFlow System** 12](#_Toc203316262)

[**1.7.3 Tracking Project Documentation Across Sprints** 13](#_Toc203316263)

# **Chapter 1**

# **Introduction**

This chapter presents the problem addressed by the project, its goals, defined scope, limitations, and the significance of the proposed solution.

## **1.1 General Background**

Modern healthcare systems are fundamentally reliant on efficient and accurate information management to deliver quality patient care. The advent of information technology has revolutionized various aspects of healthcare, enabling better data organization, streamlined administrative processes, and improved clinical decision-making globally. Digital health solutions, ranging from electronic health records to advanced telemedicine platforms, have become indispensable tools for optimizing service delivery and ensuring public health safety.

However, the continuous reliance on paper-based systems persists in many local healthcare institutions and medical points, particularly in areas affected by conflict and resource limitations such as Gaza. This manual approach to tracking patient records, medication inventory, and daily operations inevitably leads to critical challenges including frequent data loss, duplication, and delayed administrative and medical decision-making. These issues are further exacerbated by the absence of a centralized system for managing aid, medical supplies, and beneficiary information, which compromises accountability and equitable distribution of vital resources. Consequently, the pressing need for an efficient, unified digital solution capable of maintaining accurate records and ensuring fair and transparent service delivery has become acutely evident.

## **1.2 Problem Statement**

Despite the recognized benefits of digital information systems in healthcare, numerous medical points and small health institutions in Gaza continue to operate under a workflow heavily reliant on outdated paper-based records and fragmented digital spreadsheets. This prevalent manual approach introduces severe vulnerabilities, including a high susceptibility to human error, significant risks of data loss, and pervasive inconsistencies across patient records and operational logs.

Critically, the absence of a centralized and unified database specifically impedes real-time visibility and control over vital assets. This manifests as significant challenges in tracking medication stock levels, hindering effective inventory management, and complicating the verification of dispensed medications and the relevant patient encounter data. Furthermore, the practice of manual data recording throughout the day, followed by delayed digital entry by dedicated staff, creates redundant workflows and introduces additional points of potential error and inefficiency.

Moreover, this disjointed and non-integrated operational paradigm severely impacts internal coordination among medical and administrative staff, leading to diminished transparency and compromised operational efficiency. The direct consequences include sub-optimal documentation practices, prolonged service delivery times, and an elevated risk of resource mismanagement, ultimately affecting the quality and equity of healthcare provision in these critical environments.

## **1.3 Objectives**

The primary goal of this project is to address the challenges posed by manual and fragmented systems in healthcare facilities in Gaza by providing an efficient, centralized digital solution. This main objective is supported by several specific objectives designed to ensure comprehensive system development and impactful delivery.

### **1.3.1 Main Objective**

To **design and implement** a **centralized and integrated digital system** for managing the essential daily operations of medical points in Gaza, specifically focusing on **patient intake and appointments, medication dispensing, and comprehensive inventory control**, leveraging a robust and accessible database to enhance operational efficiency and data accuracy.

### **1.3.2 Sub Objectives**

* **To analyze and document** the existing administrative and workflow practices in targeted local medical points to identify key pain points and functional requirements.
* **To develop and structure** a PostgreSQL database capable of securely managing patient registration data, detailed medication stock levels, and comprehensive dispensing records.
* **To engineer and implement** a user-friendly desktop application using Java and JavaFX to automate core processes, including patient registration, appointment scheduling, and medication distribution.
* **To integrate** real-time inventory updates directly with medication dispensing processes to ensure immediate and accurate stock reflection.
* **To design the system architecture** with inherent scalability to support seamless future expansion into a fully integrated web-based interface.
* **To significantly reduce** reliance on paper-based and fragmented spreadsheet systems, thereby mitigating data loss, duplication, and human errors.
* **To enhance** the overall accuracy, transparency, and accountability of medication and patient data management within the medical points.
* **To provide** comprehensive reporting and statistical analysis tools to facilitate informed decision-making and efficient record tracking.

## **1.4 Scope and Limitations**

This section delineates the boundaries of the proposed system, specifying what functionalities and technologies are included in the current development phase (Scope) and what aspects are intentionally excluded or represent inherent constraints (Limitations). This clarity ensures a focused development effort and manages expectations for the project's deliverables.

### **1.4.1 Scope**

* **Development Platform:** To develop a robust desktop application utilizing Java and JavaFX for the user interface and backend logic.
* **Database Management:** To implement a PostgreSQL database, centrally hosted on a cloud server, ensuring secure, concurrent, and scalable data storage and retrieval.
* **Core Functionalities:** To automate essential medical point operations, including patient registration, appointment scheduling, medication dispensing, and comprehensive inventory control with real-time stock updates.
* **Target Environment:** To design and implement the HealthFlow system specifically for healthcare institutions and small medical points operating within the unique context of Gaza.
* **Architectural Design for Scalability:** To engineer the HealthFlow system's architecture with inherent flexibility and modularity to facilitate future expansion towards web-based and mobile interfaces without requiring a complete system rebuild.

### **1.4.2 Limitations**

* **Platform Exclusivity:** The system's initial version will be exclusively a desktop application; the development of dedicated web or mobile interfaces falls outside the scope of this project.
* **Clinical Depth:** The system's scope for patient data is limited to administrative and dispensing records (patient intake, appointments, dispensed medications, basic visit details), and does not include comprehensive clinical health records, diagnostic information, or advanced medical imaging integration.
* **External System Integration:** Integration with external platforms, such as national health information systems, insurance databases, or other governmental medical records systems, is beyond the current project's scope.
* **Advanced Data Validation:** While basic input validation will be implemented, the HealthFlow system primarily relies on the accuracy of user-entered data and does not incorporate advanced, AI-driven data anomaly detection or complex external data verification mechanisms.
* **Infrastructure Dependency:** The system's HealthFlow  optimal performance and accessibility are contingent upon the availability of stable internet connectivity and reliable power supply at the medical points, which are external factors beyond the system's control.

## **1.5 Project Significance & Benefits**

The proposed HealthFlow system carries substantial significance, offering a practical and timely solution to the deeply rooted challenges prevalent in medical points within resource-limited environments like Gaza. Its implementation is poised to deliver multifaceted benefits, addressing the critical issues identified in current operational paradigms.

Firstly, by **significantly reducing reliance on inefficient paper-based systems and fragmented spreadsheets**, the project directly mitigates the risks of data loss, duplication, and human errors. This fundamental shift will **enhance the reliability and integrity of patient registration records, appointment scheduling, medication dispensing data, and inventory reporting**, providing a robust digital foundation.

Secondly, through the **centralization of data and digitalization of core operations**, the system will significantly **improve operational transparency and efficiency**. This enhancement is crucial for empowering medical and administrative staff with real-time, accurate information, thereby facilitating **faster and more informed decision-making** regarding patient care, resource allocation, and overall service delivery. It will also foster greater **accountability** in medication management and resource utilization.

Furthermore, this project extends beyond immediate operational improvements. It serves as a **critical foundational step towards the modernization and development of healthcare infrastructure in Gaza**. By proving the viability and benefits of an integrated digital solution, it lays a solid groundwork for future expansion into more comprehensive health information systems, aligning with global trends in digital healthcare transformation and contributing to the long-term resilience and quality of local medical services.

## **1.6 Document Structure**

This report is organized into six main chapters, each addressing a critical aspect of the HealthFlow project.

* **Chapter 1: Introduction** provides an overview of the project, including the general background, problem statement, objectives, scope, limitations, and overall significance. It also introduces the adopted methodology and outlines the planned work phases using the Scrum framework.
* **Chapter 2: Literature Review and Related Work** explores existing research and similar systems, analyzing relevant theories, concepts, and technologies that inform this project.
* **Chapter 3: System Analysis and Design** details the comprehensive requirements analysis, architectural design, database design, and user interface design for the proposed system.
* **Chapter 4: Implementation** describes the development environment, system components, and the technical realization of the HealthFlow system.
* **Chapter 5: Testing and Evaluation** outlines the testing methodologies, test cases, and the results of system performance and functionality evaluations.
* **Chapter 6: Conclusion and Recommendations** summarizes the project's findings, discusses its limitations, and provides recommendations for future work and potential enhancements.

## **1.7 Proposed Project Methodology & Work Plan**

To ensure effective project management and achievement of objectives within the defined timeframe, the **Agile Methodology**, specifically the **Scrum framework**, will be adopted. Scrum provides an iterative and incremental approach well-suited for software projects, particularly those that may experience evolving requirements or require continuous feedback from the supervisor. This methodology offers high flexibility, recurrent delivery of results, and continuous improvement in product quality.

### **1.7.1 Scrum Implementation**

The development of the HealthFlow project will be divided into successive **Sprints**, each lasting **two weeks**. This division will allow for incremental delivery of functional system components, facilitating continuous review with the supervisor and effective incorporation of feedback. Each sprint will commence with a Sprint Planning meeting to define prioritized tasks, followed by brief Daily Scrums to track progress and address impediments. At the end of each sprint, a Sprint Review meeting will be held to demonstrate completed work, and a Sprint Retrospective meeting will be conducted to improve the development process for subsequent sprints.

### **1.7.2 Detailed Timeline**

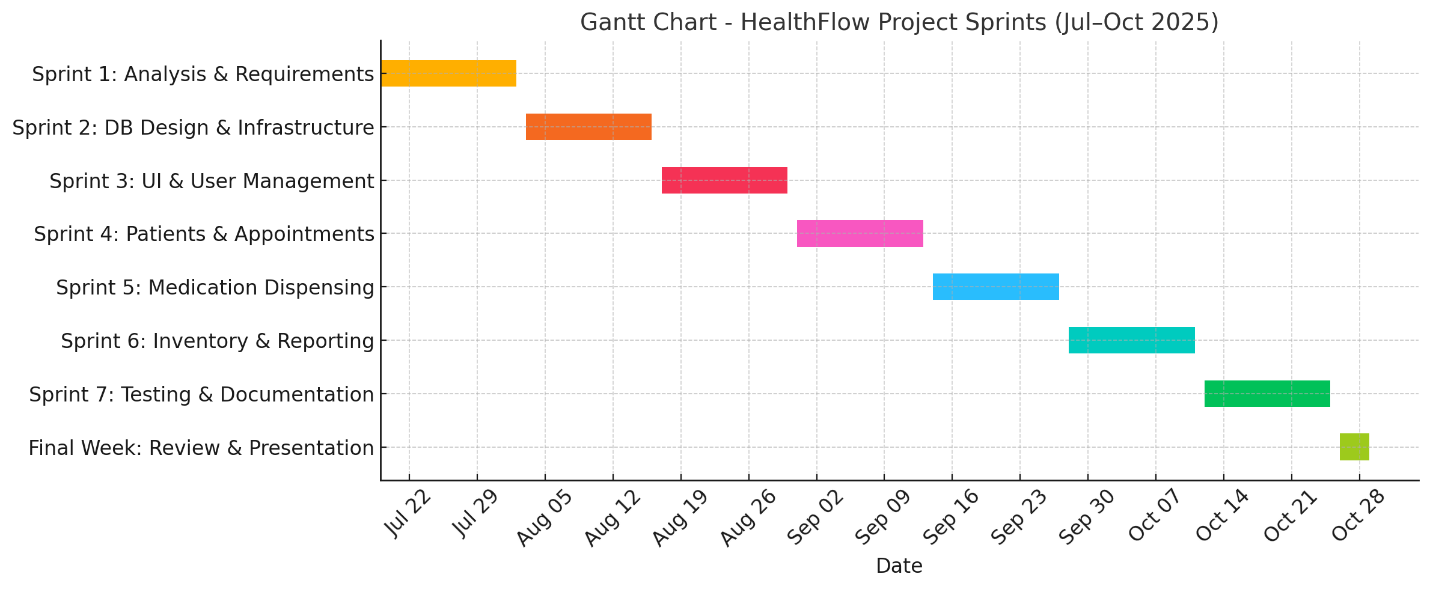
Based on the available timeframe from July 19, 2025, to October 29, 2025 (approximately 15 weeks), the work will be distributed across seven main sprints, in addition to a final week for comprehensive review and presentation preparation:

* **Semester Duration:** July 19, 2025 to October 29, 2025.
* **Total Weeks:** 15 weeks.
* **Sprint Duration:** 2 weeks.
* **Number of Sprints:** 7 Sprints.
* **Final Preparation Week:** 1 week.

### **Table 1.1 – Sprint Timeline and Key Deliverables for HealthFlow System**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sprint / Phase** | **Start**  **Date** | **End**  **Date** | **Main Tasks** | **Expected Deliverables at End of Sprint** | **Status** |
| **Sprint 1: Initial Analysis and Requirements Planning** | 07-19 | 08-02 | - Conduct system requirements analysis - Draft use cases - Outline initial database structure | - Preliminary requirements document - Use case list - Initial database design | Planned |
| **Sprint 2: Database Design and Infrastructure Setup** | 08-03 | 08-16 | - Finalize the ERD - Develop relational schema - Set up the PostgreSQL environment - Begin model class implementation | - Completed ERD diagram - Implemented database schema - Initial backend model classes | Planned |
| **Sprint 3: User Interface Development (Login and User Management)** | 08-17 | 08-30 | - Design login/user management interfaces - Implement authentication and user CRUD operations | - Fully functional login interface - Complete user management module | Planned |
| **Sprint 4: Patient and Appointment Management** | 08-31 | 09-13 | - Implement patient records and appointment scheduling features - Integrate with the database - Develop the doctor's interface | - Operational patient and appointment modules - Database-integrated doctor interface | Planned |
| **Sprint 5: Medication Dispensing and Inventory Update (Part 1)** | 09-14 | 09-27 | - Design medication dispensing interface - Implement dispensing logic with initial inventory update integration | - Medication dispensing module - Initial inventory update logic | Planned |
| **Sprint 6: Comprehensive Inventory Management and Reporting (Part 1)** | 09-28 | 10-11 | - Develop inventory management interface and functions - Design reporting features - Implement basic report viewing | - Complete inventory management module - Functional basic reporting system | Planned |
| **Sprint 7: Testing, Enhancements, and Documentation** | 10-12 | 10-25 | - Conduct full system testing - Apply supervisor feedback - Enhance performance - Finalize technical documentation | - Stable and optimized system - Finalized documentation | Planned |
| **Final Week: Review and Presentation** | 10-26 | 10-29 | - Perform final review - Prepare and deliver the project presentation | - Finalized project and documentation ready for presentation | Planned |

**Gantt Chart - HealthFlow Project Sprints (Jul–Oct 2025)**



### **1.7.3 Tracking Project Documentation Across Sprints**

This section outlines where to integrate both technical and functional documentation throughout the development sprints of the **HealthFlow** project. The goal is to ensure that documentation progresses in parallel with implementation, rather than being delayed to the final stages.

**Sprint 1: Initial Analysis and Requirement Planning**  
• Functional and non-functional requirements → Chapter 3, Section 3.1: Requirements Analysis  
• Use Case list and diagram → Chapter 3, Section 3.1.3: Use Cases  
• Preliminary database concept → Chapter 3, Section 3.3.1: Initial Database Design

**Sprint 2: Database Design and Infrastructure Setup**  
• Final Entity-Relationship Diagram (ERD) → Chapter 3, Section 3.3.2  
• Relational schema design → Chapter 3, Section 3.3.3  
• UML Class Diagrams → Chapter 3, Section 3.2: System Architecture

**Sprint 3: Main Interface and Receptionist Module**  
• Main user interface design → Chapter 3, Section 3.4.1  
• Role-based access control (RBAC) system → Chapter 3, Section 3.4.2  
• GUI screen flow/navigation → Chapter 3, Section 3.4.3

**Sprint 4: Completion of Receptionist Tasks and Doctor Module**  
• Patient and appointment management interfaces → Chapter 3, Section 3.4.4  
• Use case updates → Chapter 3, Section 3.1.3  
• Doctor dashboard and related features → Chapter 3, Section 3.4.5

**Sprint 5: Pharmacist Interface and Prescription Approval**  
• Pharmacist interface → Chapter 3, Section 3.4.7  
• Integration of approval system with inventory → Chapter 3, Section 3.5  
• Updates to Prescription and PrescriptionApproval models → Chapter 3, Section 3.3.5

**Sprint 6: Inventory Management and Reporting**  
• Inventory management interfaces → Chapter 3, Section 3.4.6  
• Inventory-related database tables → Chapter 3, Section 3.3.4  
• Report types and templates → Chapter 4, Section 4.1: Outputs and Reports

**Sprint 7: Testing, Optimization, and Final Documentation**  
• System testing plan → Chapter 4, Section 4.2  
• Improvement logs and results → Chapter 4, Section 4.3  
• Final documentation review and integration → Chapter 4, Section 4.4

**Final Week: Project Presentation**  
• Presentation materials → Chapter 5, Section 5.1  
• Final review of all deliverables → Chapter 5, Section 5.2

# **Chapter 3**

# **System Analysis and Design**

This chapter covers the analytical and design aspects of the **HealthFlow** system. It begins with a comprehensive analysis of the system's requirements—both functional and non-functional—followed by a presentation of the main use cases that represent expected user interactions. The proposed architectural structure is then detailed through structural diagrams and a database schema design. The design of the Graphical User Interfaces (GUI) is also presented to show how different roles interact with the system. This chapter serves as the foundation for the system's development process, translating user needs into a clear, executable model that guides the system's construction.

## **3.1 Requirements Analysis**

This section provides a detailed analysis of the features and services the proposed software solution must offer to meet the needs of a health center operating in a resource-constrained environment. The analysis includes identifying **functional requirements** (what the system must do) and **non-functional requirements** (focusing on quality attributes like security, efficiency, and usability). Additionally, key use cases that embody realistic user interaction scenarios with the system are presented.

### **3.1.1 Functional Requirements**

The following functional requirements represent the core operations the **HealthFlow** system must support to manage critical tasks within the health center and meet the needs of various user categories. Each requirement is tied to a specific user role and describes the system's expected behavior.

**FR-01: User Account Management**  
The system must allow the administrator to perform comprehensive user account management operations, including the following:

1. **Creating New User Accounts**  
   The administrator must be able to add new user accounts to the system for various roles such as doctors, receptionists, and pharmacists. The required data fields include:

* **National ID**: Stored as a primary key to ensure uniqueness and facilitate table relationships.
* **Username**: Unique login credentials, preferably an email address.
* **Password**: Set during account creation and stored in a hashed format using a secure algorithm.
* **Full Name**
* **Gender**: A value selected from a list (e.g., Male / Female).
* **Date of Birth**: Stored using the LocalDate data type.
* **Mobile Number**: Optional field for communication purposes.
* **Address**: For contact and record-keeping purposes.
* **Role Assignment**: A specific role must be assigned to each user (e.g., Administrator, Doctor, Receptionist, Pharmacist).

1. **Modifying Existing User Accounts**  
   The system must enable the administrator to update user information when needed, including the full name, username (email), assigned role, and password reset.
2. **Deleting User Accounts**  
   The system must allow the administrator to delete user accounts from the system.

**Business Rule:**  
Before executing a permanent deletion, the system must display a confirmation message to the administrator, especially if the account is linked to sensitive data such as past appointments or prescriptions.

**Organizational Note:**  
It is recommended to address policies regarding data associated with deleted accounts in a later section of the technical documentation.  
Some suggested policies include:

* **Deactivating** the account instead of permanently deleting it.
* **Reassigning** associated data to another active user.
* **Archiving** the data to ensure compliance with regulatory requirements or institutional policies.

### **FR-02: User Authentication**

The system must allow users created by the administrator to log in using their email (username) and password.  
Upon successful login, the system should display a customized interface based on the user's role, with permissions determined accordingly.

**FR-03: Patient Data Management**

The system must allow reception staff to register new patients visiting the clinic for the first time, in order to create a profile for each patient upon arrival.  
This process includes collecting the following information:

* **National ID**: Stored as a primary key to ensure record uniqueness and easy table linking.
* **Full Name**
* **Date of Birth**
* **Gender**
* **Mobile Number**
* **Address**

### **FR-04: Medical Appointment Scheduling**

The system must allow the receptionist to **schedule and manage medical appointments** for patients with specific doctors, as well as modify existing appointments. This function is intended to organize patient flow, ensure adequate time allocation for consultations, and effectively manage changes.

**1. Creating a New Medical Appointment**

The receptionist must be able to enter the following information to book an appointment:

* **Patient ID**: The unique identifier for the previously registered patient.
* **Doctor ID (User ID)**: The unique identifier of the doctor assigned to the appointment.
* **Appointment Date**: The specific date of the appointment.
* **Appointment Time**: The specific time of the appointment.
* **Additional Notes**: Optional field for any relevant details (e.g., reason for the appointment, main complaint).
* **Appointment Status**: (SCHEDULED, COMPLETED, CANCELED)

**2. Modifying an Existing Appointment**

The system must allow the receptionist to search for an existing appointment and update its details. Possible modifications include:

* Changing the **appointment date**
* Changing the **appointment time**
* Reassigning the **doctor**
* Editing the **additional notes**
* Updating the **appointment status** (e.g., SCHEDULED, COMPLETED, CANCELED)

**Business Rules**

* The **appointment status** must be automatically set to **"SCHEDULED"** upon creation by the system.
* The system must ensure that **no scheduling conflicts** occur a doctor cannot have more than one appointment at the same time during appointment creation or modification.

**FR-05: Doctor Appointment Management**

The system must provide **a dedicated interface for doctors** to manage their scheduled appointments. This interface allows the doctor to **view, filter, and update the status** of each appointment after completion.

**Required Features:**

**1. Viewing Scheduled Appointments:**

* The doctor must be able to view a **list of all their scheduled appointments**.
* The list must include:
  + **Patient information**
  + **Appointment date and time**
  + **Current appointment status**
  + **Notes** entered by the receptionist (if any)
* The doctor must be able to **filter appointments** by:
  + **Date**
  + **Status** (SCHEDULED, COMPLETED, CANCELED)

**2. Updating Appointment Status:**

* The doctor must be able to **update the status** of an appointment after it is completed.
* This should be done using clear options next to each appointment (e.g., buttons or dropdown menu).
* Available status options:
  + **COMPLETED** – if the appointment was fulfilled
  + **CANCELED** – if the appointment was canceled

**Business Rules:**

* The **appointment status is automatically set to "SCHEDULED"** when created by the receptionist.
* Once the status is changed to **"COMPLETED" or "CANCELED"**, it **cannot be reverted back** to "SCHEDULED".
* **All status changes** must be **logged** in a dedicated audit trail for tracking and review purposes.

### **FR-06: Creating and Sending a Prescription to the Pharmacy**

The system must allow the **doctor to create a new prescription** for a specific patient using their **National ID**, with the option to **link it to a medical appointment** (optional).

**The prescription data includes:**

* Doctor ID (automatically determined from the session)
* Patient ID.
* Related appointment ID (if available)
* Prescription creation date
* Optional notes
* A list of prescribed medications saved through an intermediate table *PrescriptionMedication*, which includes:

♣ Medication ID (*medicationId*)  
♣ Required quantity  
♣ Dosage instructions

After the prescription is created, it is automatically sent to the pharmacist’s interface.

**FR-07: Reviewing, Approving, and Dispensing Medications**

The pharmacist must be able to review prescriptions sent by doctors and decide to approve or reject each medication item independently. This allows for full or partial dispensing of the prescription based on medication availability. The goal of this approach is to simplify the workflow, ensure flexibility in handling stock shortages, and enable real-time inventory updates while fully documenting the process.

**Required Functions:**

**1. Display List of Prescriptions Awaiting Processing:**

The pharmacist should have an interface to view all prescriptions created by doctors and classified as "Pending Approval."  
Each prescription in the list should include the following details:

* Prescription ID
* Patient ID
* Doctor’s Name
* Date Creation
* Doctor’s notes
* **List of prescribed medications**, for each medication:
  + - Name
    - Required quantity
    - Dosage instructions
    - Stock availability indicator (✅ Available – ❌ Not Available)

1. **Take Approval/Reject Action at the Medication Level:**

* **Approve item:** If the medication is available, the pharmacist can approve it for dispensing, and the quantity is immediately deducted from inventory.
* **Reject item:** If the medication is unavailable or rejected for another reason.

1. **Record Dispensing Operations:**  
   The system must log all dispensing operations in a dedicated record  
   (*Dispensing Log/History*) with the following details:

* Prescription ID
* Patient ID
* Doctor’s name
* Date and time of the operation
* List of dispensed medications (for each medication: name, dispensed quantity)
* Rejection reason (if any)

**Business Rules:**

* **State Immutability:** A prescription cannot be modified or reprocessed after all its items have been fully handled.
* **Pre-validation:** Before enabling the approval option for the pharmacist, the system verifies the available quantity for each medication.
* **Immediate Deduction:** Dispensed medication quantities are deducted from inventory immediately upon item approval confirmation.

**FR-08: Supply Entry Registration (Supply Entry)**  
The system must allow an authorized user (such as the manager or the responsible pharmacist) to register the entry of new medication batches received from suppliers. This process ensures accurate inventory updates and enables tracking of each batch individually by its expiry date.

**Required Functions:**

1. **Register a New Batch:**  
   The system must prompt the user to enter the following data for each new batch:
   * Order number (*if available*)
   * Supplying or donating entity (*optional; may be selected from a dedicated supplier list*)
   * Receipt date (*automatically recorded as the entry date*)
   * **Batch Number:** Very important for tracking batches
   * **Expiry Date:** *Must be a future date*
   * **Received Quantity:** *Must be a positive numeric value*
2. **Automatic Inventory Update:**
   * After successfully registering the batch, the system must automatically update the total inventory quantity for that medication.

**Business Rules:**

* Supply registration must be restricted to authorized users only.
* Each batch's data must be stored separately to ensure accurate expiry tracking.
* All supply operations must be logged in a dedicated record for auditing and review purposes.

**FR-09: System Reports**

The system must provide a set of automated reports that assist the manager or responsible user in monitoring clinic performance, inventory status, and workflow. These reports should be customizable and filterable.

**Required Functions:**

1. **Administrative Reports:**

* **Patient Activity Report:** A daily or monthly report showing the number of patients and the number of appointments that were booked, completed, or canceled.

1. **Inventory Reports:**

* **Low Stock Report:** A list of medications that have reached a predefined low stock threshold. This report is essential for planning supply activities.
* **Most Dispensed Medications Report:** A report showing medications that were dispensed in large quantities over a specified time period, helping in inventory management and identifying high-demand items.
* **Expiring Medications Report:** A list of batches approaching their expiry date, helping to take necessary action in time.
* **Dispensed Medications Report (Within a Specified Period):** A report displaying all medications dispensed during a user-defined time period, with details including dispensed quantity and dispensing dates, supporting follow-up and analysis.

**Business Rules:**

* All reports must be filterable by time period (daily, weekly, monthly, custom).
* Reports must be exportable to common formats (e.g., Excel).
* Reports should only be accessible to users with administrative or reporting privileges.
* Reports should be presented graphically (charts/graphs) when possible to facilitate data understanding.

### **3.1.2 Non-Functional Requirements**

In addition to the core functionalities that the system must provide, there is a set of non-functional requirements that define the system’s qualitative attributes, including the following:

**1. Performance**

Related to the system’s speed and efficiency in responding to user actions.

* **NR-01 Response Time:** The system must respond to user requests (e.g., login, searching for a patient, viewing the appointment schedule) in under 3 seconds under normal conditions
* **NR-02 Scalability:** The system should be capable of handling an increase in the number of concurrent users (e.g., 10 users at the same time) and data volume without a noticeable decline in performance

#### **2. Security**

Concerns the protection of data and the system from unauthorized access.

* **NR-03 User Authentication:** Passwords must be encrypted and stored as hashed values in the database
* **NR-04 Access Control:** The system must assign specific access permissions to each user role; for example, a receptionist should not have access to administrative reports.
* **NR-05 Data Protection:** Sensitive data (such as patient personal information) must be encrypted during transmission between the desktop application and the cloud database, and also encrypted at rest in the database**.**

#### **3. Usability**

Concerns how easy and efficient the system is for users to operate.

* **NR-06 Intuitive User Interface:** User interfaces must be clear, easy to navigate, and consistently designed
* **NR-07 Visual Feedback:** The system should provide immediate visual feedback to users, such as success indicators when data is saved or clear error messages for incorrect input.
* **NR-08 Operating System Compatibility:** The application must run reliably on any desktop operating system (Windows, macOS, Linux) that has the appropriate version of the Java Runtime Environment (JRE) installed**.**

#### **4. Reliability**

Related to the system's ability to operate continuously and accurately without failure.

* **NR-09 Availability:** The system must be available for operation 99% of the time (e.g., 24/7 excluding scheduled maintenance periods).
* **NR-10 Error Handling:** The system must handle errors properly and provide helpful error messages to the user instead of displaying vague technical errors.

#### **5. Maintainability**

#### Related to how easily the system can be modified or repaired in the future.

* **NR-11 Ease of Maintenance:** The system's source code must be well-structured and documented to make it easier for new developers to understand and apply updates.

## **3.2 Use Cases**

## Use cases represent key scenarios in which users interact with the HealthFlow system. This section aims to illustrate the core operations from the end-user’s perspective, helping to practically understand the system requirements and clarify the boundaries of interaction with the system.

**UC-01: User Login**  
**Description:**  
This use case allows any system user (Admin, Doctor, Receptionist, Pharmacist) to log into the system using their credentials.  
**Actors:**  
User (Admin / Doctor / Receptionist / Pharmacist).  
**Preconditions:**  
The user must have an active account in the system.  
**Trigger:**  
Opening the login screen and entering credentials.  
**Inputs:**  
Username, Password.  
**Outputs:**  
Login success or failure message.

**Main Flow:**

**1.** The user opens the login screen.

**2**. The system displays the email and password fields.

**3**. The user enters the required credentials.

**4.** The system verifies the credentials.

**5.** If valid, the system redirects the user to the interface corresponding to their role

**Alternative Flows:**

* **Incorrect email or password → Display an error message and allow retry.**
* **Account is locked or disabled → Display a warning message and halt the login process.**

**Exceptions:**

* **Failure to connect to the database server → Display the message: "Connection failed, please try again later."**

**Final Result:**

* **The user is successfully logged in and granted access to their designated interface.**

**UC-02: Managing User Accounts via Database**

**Description:**

This use case allows the admin to directly manage user accounts through the database by performing insert, update, and delete operations on user-related tables.

**Actors:**

Admin – System Administrator or Database Administrator.

**Preconditions:**

* **The admin must have direct access permissions to the database.**
* **Knowledge of table schema and field names is required.**

**Trigger:**

A need to create a new account, modify existing account details, or delete an account for administrative or security reasons.

**Inputs:**

* User data: Full name, username, initial (encrypted) password, role, etc.

**Outputs:**

* Record inserted, updated, or deleted in the Users Table.

**Main Flow:**

1.The admin opens a database management tool such as DBeaver or PgAdmin.

2.Executes an SQL command to insert, update, or delete user data in the users table.

3.The database validates constraints and data format.

4.Changes are saved in the database.

**Alternative Flows:**

* Invalid or duplicate data entry → Operation is rejected, and a database error (Constraint Violation) is shown.

**Exceptions:**

* Loss of connection to the database during the operation.

**Final Result:**

User data is directly updated in the database, which is reflected in the system during subsequent use.

**UC-03: Add New Patient**

**Description:**

Enables the receptionist to add a new patient to the database, storing their personal and basic medical information.

**Actors:**

Receptionist.

**Preconditions:**

 The user is logged in and has permission to add patients.

* The patient is not already registered in the system (verify via national ID or patient ID)

**Trigger:**

The need to register a patient for the first time before providing any services.

**Inputs:**

Full name, gender, date of birth, national ID number, contact information.

**Outputs:**

Confirmation of successful patient registration or an error message in case of duplicate data.

**Main Flow:**

1. The user selects the "Add New Patient" option.
2. Enters the required information.
3. The system checks for an existing patient record.
4. The system saves the data into the database.
5. A conformation message is displayed upon successful registeration.

**Alternative Flows:**

* Incomplete data entered → Show a warning message and highlight the required fields.
* Patient already registered → Display their existing record instead of creating a new one.

**Exceptions:**

* Failure to save data in the database due to connection issues or data conflicts.

**Final Result:**

A new patient record is added to the database, making it ready for use in appointments or other services.

**UC-04: Book Medical Appointment**

**Description:**

Allows the receptionist to book a medical appointment for a patient with a specific doctor at a specified date and time.

**Actors:**

Receptionist.

**Preconditions:**

Patient data must already exist in the system.

**Trigger:**

The receptionist selects the "Book New Appointment" option.

**Inputs:**

Patient ID, Doctor ID, date and time, additional notes (if any).

**Outputs:**

Appointment confirmation or an error message.

**Main Flow:**

1. The receptionist enters the appointment details.
2. The system checks the doctor’s availability at the requested time.
3. The system saves the appointment data.
4. A conformation message is displayed.

**Alternative Flows:**

* Doctor not available at the requested time → Suggest alternative time slots.
* Patient data not found → Prompt to add patient record first.

**Exceptions:**

* Error occurred while saving the data.

**Final Result:**

The appointment is added to both the doctor’s and patient’s appointment schedule.

**UC-05: Manage Appointments**

**Description:**

Allows an authorized user (receptionist or doctor) to modify or cancel existing appointments.

**Actors:**

Receptionist, Doctor.

**Preconditions:**

The user is logged in and has the necessary permissions to manage appointments.

**Trigger:**

Selecting an appointment from the schedule and choosing to modify or cancel it.

**Inputs:**

Modified appointment data.

**Outputs:**

Confirmation of modification or cancellation.

**Main Flow:**

1. The user selects the desired appointment.
2. Enters modifications or chooses to cancel.
3. The system saves the changes.

**Alternative Flows:**

* Attempting to modify an appointment that is in the past → Display rejection message.

**Exceptions:**

* Update failure due to server error.

**Final Result:**  
The appointment is updated or deleted from the system

**UC-06: Add Medical Prescription**

**Description:**Allows the doctor to create a medical prescription for the patient after the medical examination

**Actors:**Doctor

**Preconditions:**The patient is registered in the system

**Trigger:**The doctor selects the "Add Prescription" option from the patient’s file

**Inputs:**Required medications, dosages, additional notes

**Outputs:**Confirmation of prescription addition

**Main Flow:**

1. The doctor enters the prescription details.
2. The system saves the prescription with the status "Pending Approval

**Alternative Flows:**

* Incomplete data entry → Display an error message.

**Exceptions:**

* Failure to save the prescription.

**Final Result:**  
The prescription is added and linked to the patient’s file

**UC-07: Review and Approve Medical Prescriptions**  
**Description:**  
Allows the pharmacist to review prescriptions sent by doctors and decide to approve or reject each medication individually, enabling full or partial dispensing based on drug availability, with immediate inventory updates and documentation of the transaction in the dispensing record.

**Actors:**  
Pharmacist.

**Preconditions:**  
There is a prescription in the "Pending Approval" status.

**Trigger:**  
The pharmacist opens the list of new prescriptions.

**Inputs:**  
Pharmacist’s decision, notes.

**Outputs:**  
Prescription status update, inventory update, transaction recorded in the dispensing log

**Main Flow:**

1. The system displays the prescription details and a list of medications with an inventory availability indicator for each drug.
2. The pharmacist decides for each medication to either approve or reject it, providing a reason if rejected.
3. For approved medications, the system immediately deducts the quantity from the inventory.
4. The system saves the decisions.

**Alternative Flows:**

* Partial dispensing when only some medications are available.

**Exceptions:**

* Failure to update inventory or an attempt to reprocess a completed prescription.

**Final Result:**  
The prescription status is updated, available medications are dispensed, and the transaction is documented in the dispensing log with details of missing medications.

**UC-08: Manage Medication Inventory**  
**Description:**  
Allows the pharmacist or administrator to add, edit, or delete medication data.

**Actors:**  
Pharmacist, Administrator.

**Preconditions:**  
The user has inventory management permissions.

**Trigger:**  
Opening the inventory management screen.

**Inputs:**  
Medication data (name, quantity, expiration date, etc.).

**Outputs:**  
Operation confirmation.

**Main Flow:**

1. Enter or edit medication data.
2. The system validates the input data (e.g., positive quantity, valid expiration date).
3. Save the changes.

**Alternative Flows:**

* Negative quantity or invalid expiration date entered → Operation is rejected.

**Exceptions:**

* Failure to save data to the database.

**Final Result:**  
Inventory data is successfully update

**UC-09: Generate Reports**  
**Description:**  
Allows the administrator to generate administrative or medical reports.

**Actors:**  
Administrator.

**Preconditions:**  
The administrator is logged in.

**Trigger:**  
Selecting the "Reports" option from the menu.

**Inputs:**  
Report type, time period.

**Outputs:**  
Report file (PDF or Excel).

**Main Flow:**

1. The administrator selects the report type and time period.
2. The system displays or saves the report

**Alternative Flows:**

Invalid time period selected → Display an error message.

**Exceptions:**

* Error occurred while generating the report.

**Final Result:**  
The requested report is successfully generated.

**UC-10: Record Drug Supply Batch**  
**Description:**  
Allows an authorized user (such as the manager or responsible pharmacist) to record batches of medications received from suppliers, with all batch details saved and inventory updated automatically.

**Actors:**  
Manager or Responsible Pharmacist.

**Preconditions:**

* User has permission to record supply transactions.
* The medication exists in the database.

**Trigger:**  
Receiving a new batch of medications from the supplier.

**Inputs:**  
Medication ID, invoice or delivery note number, supplier (optional), batch number, receipt date, expiration date, received quantity.

**Outputs:**  
Confirmation of batch registration and update of the total inventory quantity.

**Main Flow:**

1. The user selects the "Register New Batch" option.
2. Enters the batch details.
3. The system validates the data (e.g., future expiration date, positive quantity).
4. The system saves the batch data and updates the total inventory quantity.

**Alternative Flows:**

* Invalid data entered (negative quantity or expired date) → Display an error message and reject the operation.

**Exceptions:**

* Failure to save data to the database.

**Final Result:**  
Batch data is added to the supply record and the inventory is successfully updated