

**FINAL DOCUMENTATION**

**10 October 2024**

**GROUP 5 – MediVault**

**LESEGO SENAMELA 202205170**

**ROSEMARY ERAWEMEN 202204102**

**AMOGELANG MATHUMBU 202210641**

**INNOCENTIA LEDIMO 202113328**

Submitted in partial fulfilment of the requirements of **MCOA032 Computer Networks**

**BSc Mathematical Sciences**

in the

**Department of Computer Science and Information Technology**

School of Science and Technology

Sefako Makgatho Health Sciences University

Supervisor:

**Ms. V Sumbana**

Table of Contents

[Introduction 5](#_Toc179235507)

[System Dependencies 5](#_Toc179235508)

[Project report 6](#_Toc179235509)

[Project objectives 6](#_Toc179235510)

[Team roles 6](#_Toc179235511)

[Project timeline 7](#_Toc179235512)

[Challenges and Solutions 8](#_Toc179235513)

[Project Outcomes 8](#_Toc179235514)

[System Vision Documentation 10](#_Toc179235515)

[Problem Description 10](#_Toc179235516)

[System Capabilities 10](#_Toc179235517)

[System Benefits 11](#_Toc179235518)

[User Case Diagram 12](#_Toc179235519)

[Functional Requirements 13](#_Toc179235520)

[Non-Functional Requirements: 18](#_Toc179235521)

[Activity Diagrams 19](#_Toc179235522)

[Business Rules 23](#_Toc179235523)

[Database 24](#_Toc179235524)

[Tables 24](#_Toc179235525)

[Triggers 27](#_Toc179235526)

[Procedures 31](#_Toc179235527)

[Class Diagram 37](#_Toc179235528)

[System Sequence Diagrams 38](#_Toc179235529)

[Testing Plan 43](#_Toc179235530)

[This testing plan should: 43](#_Toc179235531)

[The testing will cover: 43](#_Toc179235532)

[Test types: 43](#_Toc179235533)

[Test Environment: 44](#_Toc179235534)

[Test Cases: 44](#_Toc179235535)

[Test Execution 53](#_Toc179235536)

[Defect Management 53](#_Toc179235537)

[Acceptance Criteria 53](#_Toc179235538)

[Recommendations 54](#_Toc179235539)

[Patient Self-Service Portal 54](#_Toc179235540)

[Mobile App 54](#_Toc179235541)

[Better Data Analysis 54](#_Toc179235542)

[Blockchain for Security 54](#_Toc179235543)

[Support for Multiple Languages 54](#_Toc179235544)

[Integration with Other Systems 55](#_Toc179235545)

[Automated Notifications 55](#_Toc179235546)

[Regular Security Audits 55](#_Toc179235547)

[Table 1 8](#_Toc179232737)

[Table 2 9](#_Toc179232738)

[Table 3 10](#_Toc179232739)

[Table 4 11](#_Toc179232740)

[Table 5 11](#_Toc179232741)

[Table 6 12](#_Toc179232742)

[Table 7 39](#_Toc179232743)

[Table 8 40](#_Toc179232744)

[Table 9 41](#_Toc179232745)

[Table 10 41](#_Toc179232746)

[Table 11 42](#_Toc179232747)

[Table 12 42](#_Toc179232748)

[Table 13 43](#_Toc179232749)

[Table 14 44](#_Toc179232750)

[Table 15 44](#_Toc179232751)

[Table 16 45](#_Toc179232752)

[Table 17 46](#_Toc179232753)

[Table 18 46](#_Toc179232754)

[Table 19 47](#_Toc179232755)

[Figure 1. Use Case 7](#_Toc179232987)

[Figure 2.Activity Diagram for booking an appointment. 14](#_Toc179232988)

[Figure 3.Activity Diagram for registering a new patient. 15](#_Toc179232989)

[Figure 4.Activity Diagram for editing a patient file. 16](#_Toc179232990)

[Figure 5.Activity Diagram for adding a new prescription. 17](#_Toc179232991)

[Figure 6.Activity diagram for viewing a prescription. 17](#_Toc179232992)

[Figure 7. Class Diagram for MediVault 32](#_Toc179232993)

[Figure 8. SSD for the Nurse 33](#_Toc179232994)

[Figure 9. SSD for the Pharmacy 34](#_Toc179232995)

[Figure 10. SSD for the Doctor 35](#_Toc179232996)

[Figure 11. SSD for the Patient 36](#_Toc179232997)

[Figure 12.SSD for Administrator 37](#_Toc179232998)

# Introduction

In today's fast-paced world, healthcare systems must adapt to meet growing demands for efficiency and patient care. In our country, the sector faces challenges that compromise care quality and data security. Many facilities still use outdated paper-based systems, leading to lost patient files, long appointment queues, and prescription errors, which slow processes and put personal data at risk.

To address these issues, we developed MediVault, a healthcare management system that streamlines healthcare workers' tasks by digitizing patient records, booking appointments, and automating prescriptions. With robust security features, MediVault ensures data confidentiality and allows providers to focus on delivering quality care.

MediVault also enhances the patient experience by providing quicker access to records, shorter wait times, and secure prescription management. Designed for scalability, the system can evolve with industry needs while ensuring compliance with data protection laws.

Ultimately, MediVault aims to modernize our healthcare system by closing gaps in efficiency and service quality, using technology to create a more connected, secure, and patient-centered environment.

### System Dependencies

To run or be able to view our system, you have to download XAMPP then unzip the MediVault file into the htdocs folder. After doing that you go to your browser then access it using the local host route then specify the path that will direct you to the file you want to access. Make sure that’s when you install XAMPP you configure it in also include php and mySQL so you do not experience problems.

# Project report

### Project objectives

The main goal of MediVault is to tackle the inefficiencies and vulnerabilities in our current healthcare system, such as misplaced patient files, long appointment queues, and prescription errors. Our aim is to develop a secure, efficient, and user-friendly digital platform that streamlines healthcare operations. MediVault will modernize patient record management, improve appointment booking, and ensure secure prescription handling, while also protecting sensitive patient information.

### Team roles

Below is a comprehensive outline of the expertise and responsibilities demonstrated by team members during the project's development phase:

1. Lesego Senamela

- Database Architecture and Optimization: Designed, implemented, and optimized the project's database schema, ensuring high-performance data retrieval and storage.

- Front-end Development: Developed and implemented responsive, user-centric interface components, utilizing HTML5, CSS3, and JavaScript.

- Back-end Development: Spearheaded server-side logic development, focusing on scalable system architecture, database integration, and software development best practices.

2. Innocentia Ledimo

- Database Configuration and Modeling: Collaborated on database schema design, data modeling, and query optimization to ensure efficient data storage and retrieval.

- Back-end Development Support: Contributed to server-side development, enhancing overall system performance and reliability.

3. Rosemary Erawemen

- Front-end Development: Developed and implemented responsive, user-centric interface components, utilizing HTML5, CSS3, and JavaScript.

- Documentation Management: Compiled, edited, and maintained comprehensive project documentation, incorporating inputs from all team members.

4. Amogelang Mathumbu

- Technical Writing and Documentation: Authored and edited technical guides, user manuals, and project reports, ensuring clarity and precision.

- Front-end Development Support: Provided additional expertise on client-side development tasks, enhancing overall project delivery.

Collaborative Efforts

- Cross-functional Documentation: All team members contributed to the documentation process, sharing expertise and insights to ensure a thorough and accurate project record.

### Project timeline

The project cycle was followed in accordance with the Final year project submission and presentation document. The dates provided in that document are how we were able to navigate and plan around the work required in the development of the system. Although we had 19 weeks to develop a fully functional system, we had 11 weeks to design and document the project.

### Challenges and Solutions

Challenges:

* Data Security: Protecting sensitive patient data from unauthorized access.
* Appointment Management: Handling multiple users booking appointments simultaneously.
* Prescription Automation: Ensuring secure and accurate transfer of prescriptions to the pharmacy system.

Solutions:

* Security: Implemented strong encryption protocols, user role-based access, password encryption when submitting to the database and secure login credentials.
* Concurrency Handling: Managed appointment slots using locking mechanisms to prevent double bookings.
* Automation: Integrated automated checks to ensure secure transfer of prescriptions to the pharmacy system and updated medication inventory

### Project Outcomes

MediVault successfully achieves the following:

* Efficient Patient Record Management: Provides quick access to patient data and ensures information is always up to date.
* Appointment Scheduling: Streamlines the booking process, reducing patient wait times and ensuring accurate scheduling.
* Prescription Management: Allows doctors to send prescriptions directly to the hospital pharmacy, and patients can track their medication.

MediVault has proven to be scalable and adaptable to future requirements, with robust security measures in place to ensure patient confidentiality.

# System Vision Documentation

### Problem Description

Our current healthcare system, is in critical need for modernization and efficiency improvement to address several challenges. First, important patient files are often paper-based, making them prone to getting lost. Additionally, booking appointments is a hassle, leading to long queues at hospitals. Patient history is not efficiently stored and accessible, affecting the quality of healthcare. Moreover, traditional prescriptions can be lost or misread, causing errors in medication. To fix this, we need a digital system. It will digitize files to prevent loss, offer an easy appointment booking system, store patient history in one place, and introduce digital prescriptions for accurate medication. This system aims to make healthcare smoother, more efficient, and safer for patients.

### System Capabilities

The system should:

* Enable quick and accurate retrieval of patient records using search functionalities.
* Digitalizes patient information and medical history.
* Include implementations of security measures to protect against unauthorized access and data breaches.
* Allow the patient to book an appointment for the specific date and time.
* Enable doctors to electronically send medication lists for patients directly to the hospital pharmacy.
* Provide real-time information on medication availability in the hospital pharmacy's inventory.
* Implement secure login credentials for doctors and pharmacists to ensure patient confidentiality and data security.
* Remind the patient of their upcoming appointments and to schedule their next appointments and or checkups

### System Benefits

* Reduced paper usage, printing costs, and the need for physical storage space contribute to a more cost-effective operational model.
* Improved data accuracy and integrity, ensuring that healthcare professionals rely on precise and up-to-date information for diagnosis and treatment.
* With instant access to patient records and relevant documents, healthcare professionals can make quicker and more informed decisions.
* Patient information is safeguarded from unauthorized access, reducing the risk of data breaches and maintaining trust in the healthcare institution.
* Streamline the medication prescribing process, reducing manual errors and administrative burdens for healthcare providers.
* Ensure timely access to prescribed medications, leading to improved patient satisfaction and health outcomes.
* Reduce medication processing times and minimize delays in medication collection, optimizing resource utilization and healthcare delivery costs.
* Provide patients with convenient access to prescribed medications by offering information on alternative pharmacy options if medications are unavailable at the hospital pharmacy.
* Ensure compliance with data privacy regulations and maintain patient confidentiality through secure login credentials and data encryption.

# User Case Diagram

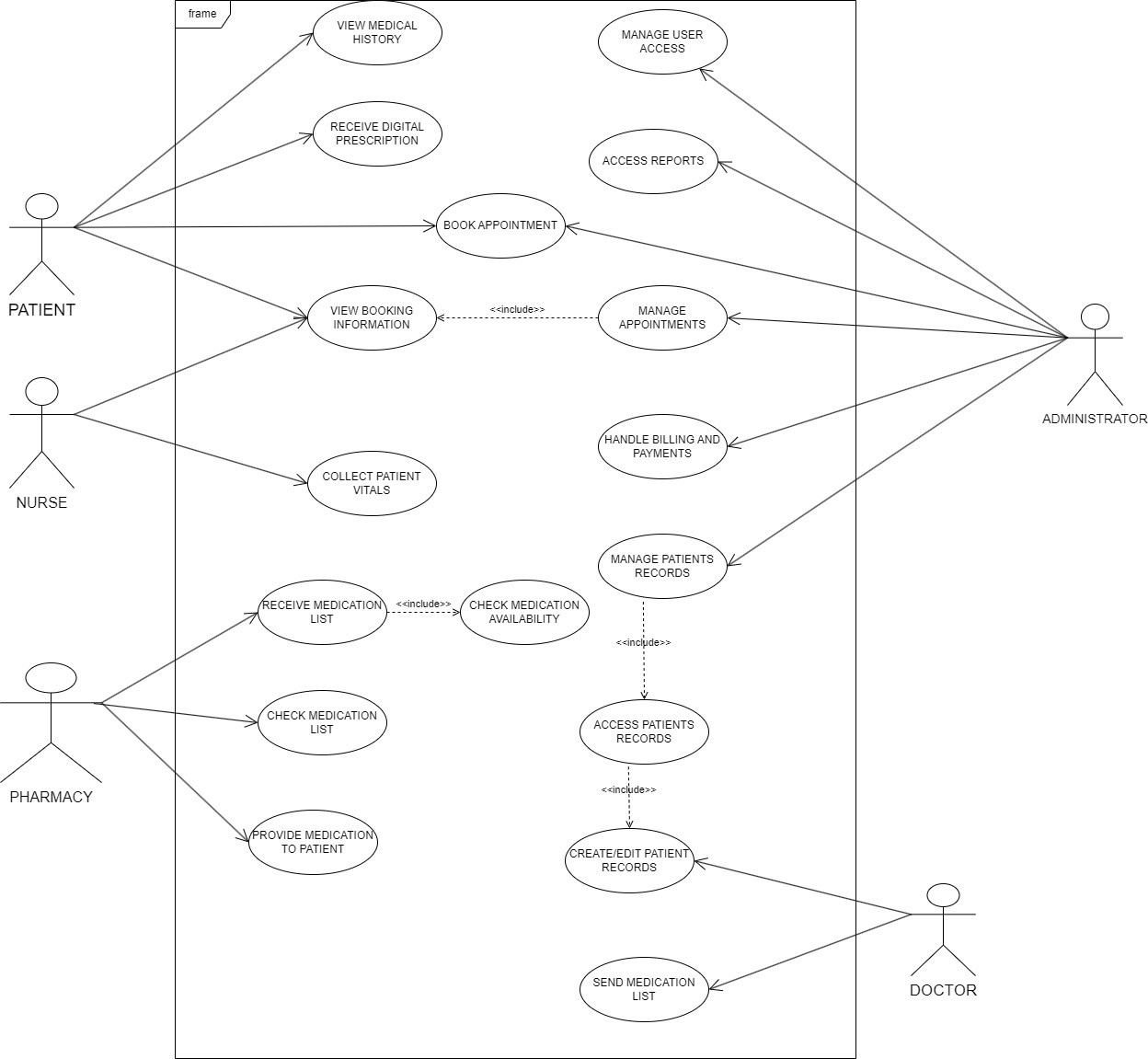


Figure 1. Use Case

# Functional Requirements

1. The system should be able to book an appointment.

|  |  |
| --- | --- |
| Element | Description |
| Functional requirement | Book an appointment |
| Functional requirement ID | FR001 |
| Priority | High |
| Actor(s) | Patient(s) and Administrator(s) |
| Description | Users should be able to choose their desired date and time slots from the given calendar dates. Once appointment is booked, users should get a confirmation message. |
| Pre-Condition(s) | The user has logged in to their account. |
| Post-Condition(s) | Gets an appointment date. |
| Flow of events | |  |  | | --- | --- | | Actor Action | System Response | | 1. Clicks **BOOKING APPOINTMENT** |  | |  | 1. Shows appointment slots in dates and times | | 1. Selects their desired date and timeslot |  | |  | 1. Saves the chosen date and time slot | | 1. Confirms the chosen date and timeslot |  | |  | 1. Displays confirmation message and saves it in the database | |
| Alternative flows | N/A |
| Exceptions | 1. The user can choose to cancel booking. 2. The user can choose another date of their choosing |

Table 1

1. The system should be able to register a new patient.

|  |  |
| --- | --- |
| Element | Description |
| Functional requirement | Register new patients |
| Functional requirement ID | FR002 |
| Priority | High |
| Actor(s) | Administrator(s) |
| Description | Users should be able to register new patients by providing necessary personal information such as name, ID number, contact details, and relevant medical history. Users should be able to assign a unique identifier to each new registration for easy identification and tracking. Once registered, the system should store the patient information securely and allow authorized users to retrieve and update it as needed. |
| Pre-Condition(s) | The user has logged in to their account. |
| Post-Condition(s) | Gets a patient file number. |
| Flow of events | |  |  | | --- | --- | | Actor Action | System Response | | 1. Clicks **NEW PATIENT** |  | |  | 1. Shows form to input patients’ information | | 1. Fills in the form |  | | 1. Clicks **SAVE** |  | |  | 1. Saves the patient information and displays the patients file number on the screen | |
| Alternative flows | N/A |
| Exceptions | 1. You don’t need to book an appointment to register as a new patient. 2. Only save when the user clicks **SAVE**. |

Table 2

1. The system should allow patients to view booking history.

|  |  |
| --- | --- |
| Element | Description |
| Functional requirement | Allow patient to view booking history |
| Functional requirement ID | FR003 |
| Priority | Medium |
| Actor(s) | Patient(s) and Administrator(s) |
| Description | The patient should be able to view their past booking information to be able to track their next checkup date so they can be able to book an appointment on time |
| Pre-Condition(s) | 1. The user logged into their account. 2. Clicked **BOOKING APPOINTMENT** |
| Post-Condition(s) | See past booking information |
| Flow of events | |  |  | | --- | --- | | Actor action | System response | | 1. Clicks **BOOKING INFORMATION** |  | |  | 1. Shows booking history | |
| Alternative flows | N/A |
| Exceptions | When no data is found the system should show NO DATA FOUND. |

Table 3

4. The system should allow the user to view the prescription.

|  |  |
| --- | --- |
| Element | Description |
| Functional requirement | Allow the user to view the prescription |
| Functional requirement ID | FR004 |
| Priority | Medium |
| Actor(s) | Patient(s) |
| Description | The user must be able to view the prescription so that they can be able to show it to the pharmacist should they choose to not use the hospital pharmacy |
| Pre-Condition(s) | 1. The user should have logged into the system. 2. The user clicked **PRESCRIPTION.** |
| Post-Condition(s) | Show the prescription to the user |
| Flow of events | |  |  | | --- | --- | | Actor action | System response | | 1. Clicks **PRESCRIPTION HISTORY** |  | |  | 1. Shows prescription history | | 1. Clicks the prescription they want to view |  | |  | 1. Shows prescription | |
| Alternative flows | N/A |
| Exceptions | If the user doesn’t have prescriptions, the system will display NO DATA FOUND |

Table 4

1. The system should allow the doctor to edit the patient’s file.

|  |  |
| --- | --- |
| Element | Description |
| Functional requirement | Allow the doctor to edit the patients file. |
| Functional requirement ID | FR005 |
| Priority | Medium |
| Actor(s) | Doctor(s) |
| Description | The doctor should be the only one who can edit the patients file to increase security. |
| Pre-Condition(s) | 1. The doctor has logged into the system. 2. The doctor has clicked **PATIENT FILE.** |
| Post-Condition(s) | The doctor has successfully edited the patients file. |
| Flow of events | |  |  | | --- | --- | | Actor action | System response | | 1. Clicks **EDIT FILE** |  | |  | 1. Shows form to enter information | | 1. Inputs the modifications |  | | 1. Clicks **SAVE** |  | |  | 1. Saves Modifications | |  | 1. Updates patients file | |
| Alternative flows | N/A |
| Exceptions | Non-essential data can be left empty. |

Table 5

1. The system should allow the user to edit booked appointments.

|  |  |
| --- | --- |
| Element | Description |
| Functional requirement | Allow the user to edit booked appointments |
| Functional requirement ID | FR006 |
| Priority | Low |
| Actor(s) | Patient(s) |
| Description | The user should be able to edit the appointment booking had they made an incorrect booking or want to change the booking. |
| Pre-Condition(s) | 1. The user has logged into their account. 2. The user has clicked **BOOKING INFORMATION** |
| Post-Condition(s) | Show new appointment information |
| Flow of events | |  |  | | --- | --- | | Actor action | System response | | 1. Click **EDIT BOOKING** |  | |  | 1. Allows user to edit or change the booking | | 1. Edits the booking then clicks **SAVE** |  | |  | 1. Saves and displays edited booking | |
| Alternative flows | N/A |
| Exceptions | If the user doesn’t save the edited booking, then it is discarded. |

Table 6

# Non-Functional Requirements:

* **Security:** Ensure compliance with healthcare data protection regulations, access controls, and auditing mechanisms to safeguard patient information.
* **Scalability:** Design the system to handle increasing volumes of patient data and user requests as the healthcare facility grows.
* **Usability:** Develop an intuitive user interface for both healthcare providers and patients, minimizing the learning curve and facilitating efficient navigation.
* **Reliability:** Ensure the system's uptime and reliability to provide uninterrupted access to critical healthcare services.
* **Performance:** Optimize system performance to handle concurrent user requests efficiently, minimizing response times for appointment bookings and data retrieval.
* **Maintainability:** Design the system to facilitate easy updates, modifications, bug fixes, and enhancements without significant overhauls or downtime. This includes creating a modular architecture, using well-documented and standardized coding practices, and implementing comprehensive logging and monitoring tools to quickly identify and address issues.
* **Adaptability:** The system should support efficient deployment of new features and be open to changing healthcare regulations and technologies.

# Activity Diagrams

A diagram of a patient administrator

Description automatically generated

Figure 2.Activity Diagram for booking an appointment.

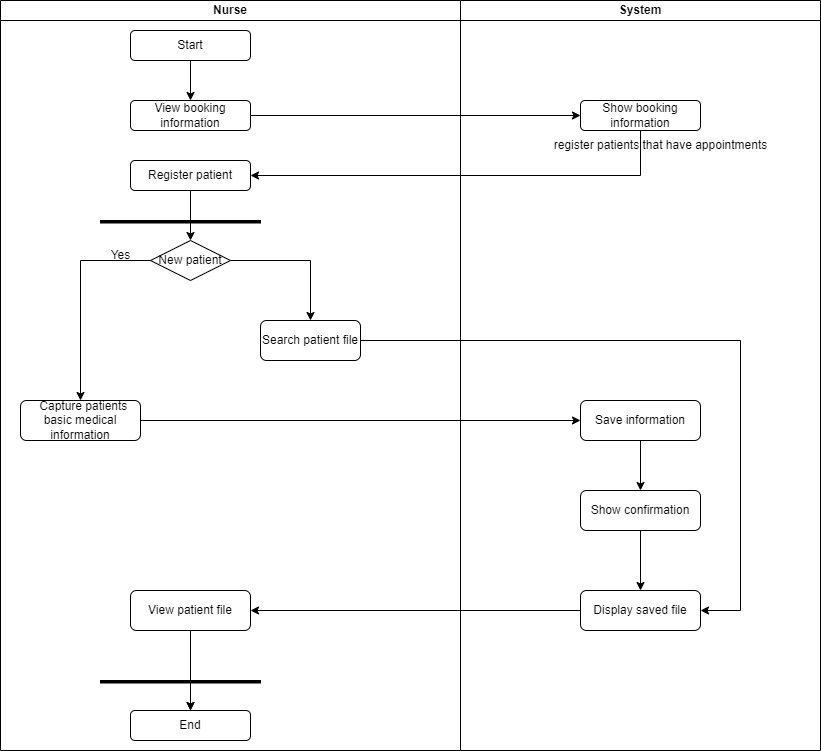


Figure 3.Activity Diagram for registering a new patient.

A diagram of a medical procedure

Description automatically generated

Figure 4.Activity Diagram for editing a patient file.

A diagram of a medical procedure

Description automatically generated

Figure 5.Activity Diagram for adding a new prescription.

A screenshot of a computer

Description automatically generated

Figure 6.Activity diagram for viewing a prescription.

# Business Rules

Here is a list of the business rules that were used to construct the class diagram.

* Patients must be able to book an appointment.
* Each Patient can have multiple Medical Records created and edited by doctors.
* Patients must receive a Prescription from their Doctors after a consultation.
* Patients can view their Medical History, which is a collection of their Medical Records.
* Patients should be able to view their Booking Information for upcoming Appointments.
* Doctors must be able to create and edit Patient file.
* Doctors must be able to send a Prescription to Patients.
* Each Doctor can have multiple Appointments with different Patients.
* Nurses must collect Patient vitals and update patients file accordingly.
* Administrators must manage User Access and roles within the system.
* Administrators must manage, update, and edit Patient Appointments to ensure proper scheduling.
* Administrators have access to manage and update Patient Records.
* Appointments must link a Patient with a Doctor.
* Each Appointment must have a unique ID, date, and time.
* Pharmacies must manage a list of available Medications.
* Pharmacies must provide Medications to Patients based on prescriptions.
* Pharmacies can check the availability of Medications before dispensing them.

# Database

For the system to function properly, here are the scripta that you should use to gat the database up and not get any errors during form submission and retrieval.

### Tables

CREATE TABLE appointments (

appointmentID INT AUTO\_INCREMENT PRIMARY KEY,

appointmentDate DATE NOT NULL,

appointmentTime TIME NOT NULL,

appointmentDetails TEXT NOT NULL,

appointmentFirstName VARCHAR(100) NOT NULL,

appointmentLastName VARCHAR(100) NOT NULL,

IDNum INT NOT NULL UNIQUE

);

CREATE TABLE insurance (

insuranceID INT AUTO\_INCREMENT PRIMARY KEY,

insuranceProvider VARCHAR(255) NOT NULL,

policyNumber VARCHAR(100) NOT NULL,

coverageDetails TEXT NOT NULL

);

CREATE TABLE medications (

medicationID INT AUTO\_INCREMENT PRIMARY KEY,

medicationName VARCHAR(255) NOT NULL,

quantity INT NOT NULL

);

CREATE TABLE patients (

patientID INT AUTO\_INCREMENT PRIMARY KEY,

firstName VARCHAR(100) NOT NULL,

lastName VARCHAR(100) NOT NULL,

IDNum INT NOT NULL UNIQUE

gender VARCHAR(10) NOT NULL,

address TEXT NOT NULL,

pastConditions TEXT,

allergies TEXT,

familyMedicalHistory TEXT,

symptoms TEXT,

BP VARCHAR(10),

heartrate INT,

temperature DECIMAL(4,1),

respiratoryRate INT,

insuranceID INT,

prescriptionID INT,

appointmentID INT,

contactID INT,

FOREIGN KEY (insuranceID) REFERENCES insurance(insuranceID),

FOREIGN KEY (prescriptionID) REFERENCES prescriptions(prescriptionID),

FOREIGN KEY (appointmentID) REFERENCES appointments(appointmentID),

FOREIGN KEY (contactID) REFERENCES emergency\_contact(contactID)

);

CREATE TABLE prescriptions (

prescriptionID INT AUTO\_INCREMENT PRIMARY KEY,

IDNum INT NOT NULL UNIQUE,

medicationName VARCHAR(255) NOT NULL,

medicationPurpose TEXT,

medicationDosage VARCHAR(100),

medicationFrequency VARCHAR(100),

physicianSignature BLOB

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE users (

ID INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL,

role VARCHAR(50) NOT NULL,

address TEXT,

firstName VARCHAR(100) NOT NULL,

lastName VARCHAR(100) NOT NULL,

contactNO VARCHAR(15),

email VARCHAR(100) NOT NULL UNIQUE,

IDNum INT NOT NULL UNIQUE

);

CREATE TABLE emergency\_contact (

contactID INT AUTO\_INCREMENT PRIMARY KEY,

contactName VARCHAR(100) NOT NULL,

relationship VARCHAR(50) NOT NULL,

contactNO VARCHAR(15) NOT NULL

);

### Triggers

CREATE TRIGGER update\_patient\_appointment

AFTER INSERT ON APPOINTMENTS

FOR EACH ROW

BEGIN

UPDATE PATIENTS SET appointmentID = NEW.appointmentID WHERE patientID = NEW.patientID;

END;

CREATE TRIGGER check\_doctor\_availability

AFTER INSERT ON APPOINTMENTS

FOR EACH ROW

BEGIN

DECLARE doctorAvailable INT;

-- Check availability of the doctor based on their schedule

SELECT COUNT(\*) INTO doctorAvailable

FROM DOCTOR\_SCHEDULE

WHERE doctorID = NEW.doctorID

AND NEW.appointmentDate = scheduleDate

AND NEW.appointmentTime = scheduleTime;

-- If doctor is unavailable, throw an error

IF doctorAvailable > 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Doctor is unavailable at the selected time';

END IF;

END;

CREATE TRIGGER update\_patient\_prescription

AFTER INSERT ON PRESCRIPTIONS

FOR EACH ROW

BEGIN

UPDATE PATIENTS

SET prescriptionID = NEW.prescriptionID

WHERE patientID = NEW.patientID;

UPDATE MEDICATION

SET quantity = quantity - 1

WHERE medicationName = NEW.medicationName AND quantity > 0;

END;

CREATE TRIGGER update\_medication\_stock

AFTER INSERT ON PRESCRIPTIONS

FOR EACH ROW

BEGIN

UPDATE MEDICATION

SET quantity = quantity - 1

WHERE medicationName = NEW.medicationName;

IF (SELECT quantity FROM MEDICATION WHERE medicationName = NEW.medicationName) <= 10 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Medication stock is running low';

END IF;

END;

CREATE TRIGGER alert\_nurse\_doctor

AFTER UPDATE ON PATIENTS

FOR EACH ROW

BEGIN

IF NEW.BP > 180 OR NEW.heartRate > 120 THEN

INSERT INTO ALERTS(patientID, alertMessage)

VALUES(NEW.patientID, 'Critical vital signs');

END IF;

END;

CREATE TRIGGER discharge\_patient\_cleanup

AFTER UPDATE ON PATIENTS

FOR EACH ROW

BEGIN

IF NEW.status = 'Discharged' THEN

DELETE FROM APPOINTMENTS

WHERE patientID = NEW.patientID AND appointmentDate > CURRENT\_DATE;

END IF;

END;

CREATE TRIGGER assign\_user\_roles

AFTER INSERT ON USERS

FOR EACH ROW

BEGIN

IF NEW.role = 'Doctor' THEN

INSERT INTO PERMISSIONS(ID, permissionType)

VALUES(NEW.ID, 'edit\_records');

ELSEIF NEW.role = 'Nurse' THEN

INSERT INTO PERMISSIONS(ID, permissionType)

VALUES(NEW.ID, 'view\_records');

END IF;

END;

CREATE TRIGGER schedule\_followup\_appointment

AFTER INSERT ON APPOINTMENTS

FOR EACH ROW

BEGIN

-- Create a follow-up appointment 30 days after the original appointment

INSERT INTO APPOINTMENTS(patientID, appointmentDate, appointmentTime, appointmentDetails)

VALUES(NEW.patientID, DATE\_ADD(NEW.appointmentDate, INTERVAL 30 DAY), NEW.appointmentTime, 'Follow-up appointment');

END;

### Procedures

CREATE PROCEDURE GetAppointmentsByPatient(

IN p\_patientID INT

)

BEGIN

SELECT \* FROM APPOINTMENTS

WHERE patientID = p\_patientID;

END;

CREATE PROCEDURE UpdateDoctorSchedule(

IN p\_scheduleID INT,

IN p\_scheduleDate DATE,

IN p\_startTime TIME,

IN p\_endTime TIME

)

BEGIN

UPDATE DOCTOR\_SCHEDULE

SET scheduleDate = p\_scheduleDate,

startTime = p\_startTime,

endTime = p\_endTime

WHERE ID = p\_scheduleID;

END;

CREATE PROCEDURE AddUser(

IN p\_username VARCHAR(50),

IN p\_password VARCHAR(50),

IN p\_role VARCHAR(20),

IN p\_address VARCHAR(255),

IN p\_firstName VARCHAR(50),

IN p\_lastName VARCHAR(50),

IN p\_contactNO VARCHAR(15),

IN p\_email VARCHAR(100)

)

BEGIN

INSERT INTO USERS (username, password, role, address, firstName, lastName, contactNO, email)

VALUES (p\_username, p\_password, p\_role, p\_address, p\_firstName, p\_lastName, p\_contactNO, p\_email);

END;

CREATE PROCEDURE GetDoctorSchedule(

IN p\_doctorID INT

)

BEGIN

SELECT \* FROM DOCTOR\_SCHEDULE

WHERE doctorID = p\_doctorID;

END;

CREATE PROCEDURE GetPatientDetails(

IN p\_patientID INT

)

BEGIN

SELECT \* FROM PATIENTS

WHERE ID = p\_patientID;

END;

CREATE PROCEDURE CancelAppointment(

IN p\_appointmentID INT

)

BEGIN

DELETE FROM APPOINTMENTS

WHERE ID = p\_appointmentID;

END;

CREATE PROCEDURE UpdatePatient(

IN p\_patientID INT,

IN p\_firstName VARCHAR(50),

IN p\_lastName VARCHAR(50),

IN p\_dob DATE,

IN p\_address VARCHAR(255),

IN p\_contactNO VARCHAR(15),

IN p\_email VARCHAR(100)

)

BEGIN

UPDATE PATIENTS

SET firstName = p\_firstName,

lastName = p\_lastName,

dob = p\_dob,

address = p\_address,

contactNO = p\_contactNO,

email = p\_email

WHERE ID = p\_patientID;

END;

CREATE PROCEDURE ScheduleAppointment(

IN p\_patientID INT,

IN p\_doctorID INT,

IN p\_appointmentDate DATE,

IN p\_startTime TIME,

IN p\_endTime TIME

)

BEGIN

INSERT INTO APPOINTMENTS (patientID, doctorID, appointmentDate, startTime, endTime)

VALUES (p\_patientID, p\_doctorID, p\_appointmentDate, p\_startTime, p\_endTime);

END;

CREATE PROCEDURE AddPatient(

IN p\_firstName VARCHAR(50),

IN p\_lastName VARCHAR(50),

IN p\_dob DATE,

IN p\_address VARCHAR(255),

IN p\_contactNO VARCHAR(15),

IN p\_email VARCHAR(100)

)

BEGIN

INSERT INTO PATIENTS (firstName, lastName, dob, address, contactNO, email)

VALUES (p\_firstName, p\_lastName, p\_dob, p\_address, p\_contactNO, p\_email);

END;

# Class Diagram

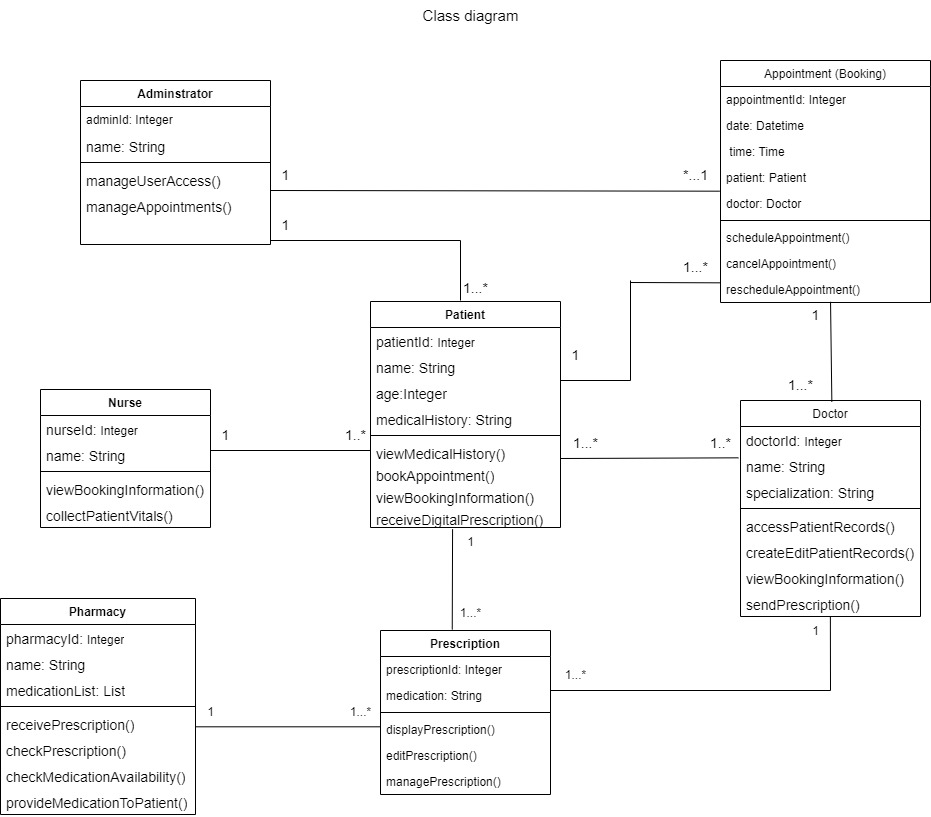


Figure 7. Class Diagram for MediVault

# System Sequence Diagrams

A diagram of a system

Description automatically generated

Figure 8. SSD for the Nurse

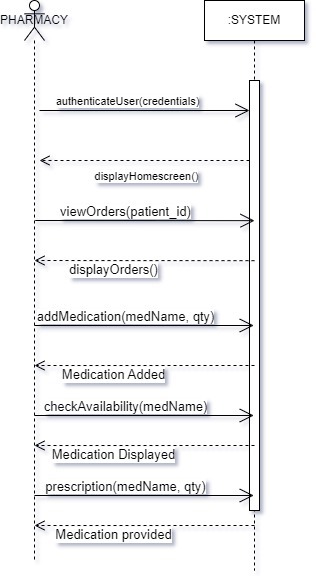


Figure 9. SSD for the Pharmacy

A diagram of a patient

Description automatically generated

Figure 10. SSD for the Doctor

A close-up of a website

Description automatically generated

Figure 11. SSD for the Patient

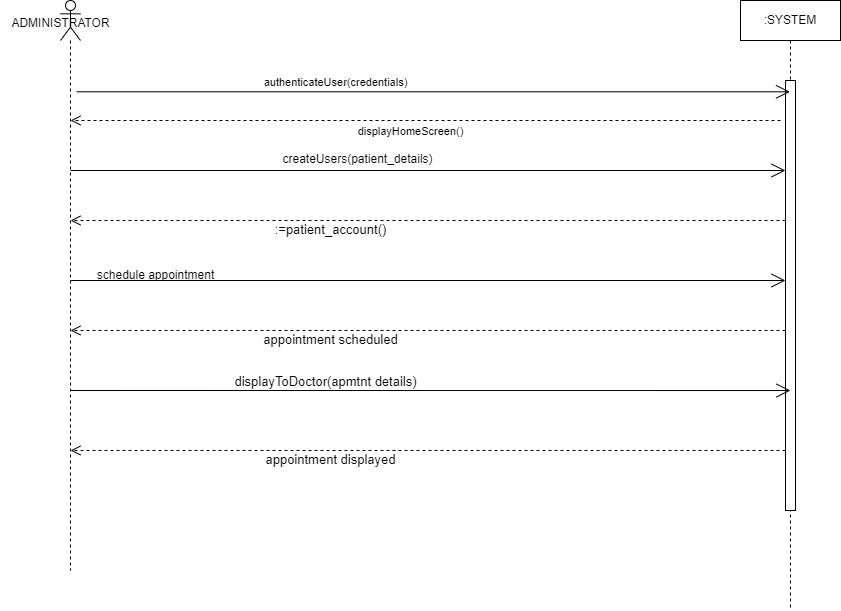


Figure 12.SSD for Administrator

# Testing Plan

This testing plan outlines the strategy and approach for testing MediVault. The aim is to ensure the system meets all requirements, functions as intended, and provides a smooth, secure, and efficient experience for users.

### This testing plan should:

* Verify that the system functions correctly according to the specifications.
* Ensure the system is secure and protects patient confidentiality.
* Validate that the user interfaces are user-friendly and accessible.
* Confirm that the system integrates seamlessly with existing laboratory and diagnostic equipment.

### The testing will cover:

* User authentication and authorization.
* Patient file management.
* Appointment booking system.
* Digital prescription system.
* Pharmacy inventory management.

### Test types:

* Functional Testing: Verify each function of the system against the requirements.
* Usability Testing: Ensure the system is easy to use.
* Performance Testing: Check the system's performance under various conditions.
* Security Testing: Validate the system's security measures.
* Integration Testing: Ensure the system works well with external systems like lab equipment.
* Recovery Testing: Test the backup and data recovery processes.

### Test Environment:

* Hardware: Computers, servers, diagnostic equipment.
* Software: Operating systems, web browsers, database systems.
* Network: LAN/WAN connections for communication between system components.

### Test Cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module Name** | User Authentication | | | |
| **Test Case ID** | TC001 | | | |
| **Tester Name** | L Senamela | | | |
| **Test Case Description** | Verify login functionality for users (patients, nurses, doctors, and pharmacists). | | | |
| **Pre-requisite** | User must be registered | | | |
| **Test Steps** | **Test Input** | **Expected Results** | **Actual Result** | **Status** |
| Enter valid credentials and log in | Username  Password  role | User should be logged in and redirected to the dashboard of their respective pages. | User is redirected to their respective pages based on their roles | Passed |

Table 7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module Name** | User Authorization | | | |
| **Test Case ID** | TC002 | | | |
| **Tester Name** | L Senamela | | | |
| **Test Case Description** | Verify login functionality for users (patients, nurses, doctors and pharmacists). | | | |
| **Pre-requisite** | User must be registered. | | | |
| **Test Steps** | **Test Input** | **Expected Results** | **Actual Result** | **Status** |
| Enter valid credentials and log in (wrong password or role) | Username  Password  role | Access should be denied. | Access is denied | Passed |

Table 8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module Name** | Patient Registration | | | |
| **Test Case ID** | TC003 | | | |
| **Tester Name** | L Senamela | | | |
| **Test Case Description** | Create a new patient file. | | | |
| **Pre-requisite** | Patient must have a booking. | | | |
| **Test Steps** | **Test Input** | **Expected Results** | **Actual Result** | **Status** |
| 1. Go to bookings. 2. Check if booking is available. 3. Go to patients. 4. Fill in the form. 5. Submit the form. | First Name  Last Name  Gender  Address  Past Conditions  Allergies  Family Medical History  Symptoms  BP  Heartrate  Temperature Respiratory Rate | Patient file should be created and retrievable. | Patient file is created and retrievable. | Passed |

Table 9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module Name** | Search Patient File | | | |
| **Test Case ID** | TC004 | | | |
| **Tester Name** | L Senamela | | | |
| **Test Case Description** | Search for an existing patient file. | | | |
| **Pre-requisite** | Patient must have a record. | | | |
| **Test Steps** | **Test Input** | **Expected Results** | **Actual Result** | **Status** |
| 1. Go to search bar. 2. Choose patient. 3. Enter the first name | First Name | Patient file should be retrievable and displayed. | Patient file is retrievable and displayed. | Passed |

Table 10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module Name** | Edit Patient File | | | |
| **Test Case ID** | TC005 | | | |
| **Tester Name** | L Senamela | | | |
| **Test Case Description** | Modify an existing patient file. | | | |
| **Pre-requisite** | Patient must have a record. | | | |
| **Test Steps** | **Test Input** | **Expected Results** | **Actual Result** | **Status** |
| 1. Go to patients list. 2. Click edit button. 3. Modify data in the form. 4. Submit the form. | First Name  Last Name  Gender  Address  Past Conditions  Allergies  Family Medical History  Symptoms  BP  Heartrate  Temperature Respiratory Rate | Patient file should be retrieved and displayed with data to modify.  Allow to edit patient file. | * Patient file is be retrieved and displayed with data to modify. * Allow to edit patient file. | Passed |

Table 11

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module Name** | Delete Patient File | | | |
| **Test Case ID** | TC006 | | | |
| **Tester Name** | L Senamela | | | |
| **Test Case Description** | Delete an existing patient file. | | | |
| **Pre-requisite** | Patient must have a record. | | | |
| **Test Steps** | **Test Input** | **Expected Results** | **Actual Result** | **Status** |
| 1. Go to patients list. 2. Click delete button. | N\A | User should be asked for confirmation to delete. | User is asked for confirmation to delete. | Passed |

Table 12

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module Name** | Appointment Booking | | | |
| **Test Case ID** | TC007 | | | |
| **Tester Name** | L Senamela | | | |
| **Test Case Description** | Book an appointment. | | | |
| **Pre-requisite** | User must have an account. | | | |
| **Test Steps** | **Test Input** | **Expected Results** | **Actual Result** | **Status** |
| 1. Go to booking. 2. Fill in the form. 3. Submit the form. | Appointment Date  Appointment Time  Appointment Details  First Name  Last Name | * User should be able to see the form and fill it in. * User should be able to submit the booking. | * User can see the form and fill it in. * User can submit the booking. | Passed |

Table 13

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module Name** | Edit Appointment | | | |
| **Test Case ID** | TC008 | | | |
| **Tester Name** | L Senamela | | | |
| **Test Case Description** | Edit an existing appointment. | | | |
| **Pre-requisite** | User must have a booking. | | | |
| **Test Steps** | **Test Input** | **Expected Results** | **Actual Result** | **Status** |
| 1. Go to appointments list 2. Click edit button 3. Modify data in the form. 4. Submit the form | Appointment Date  Appointment Time  Appointment Details  First Name  Last Name | * User should be asked for confirmation to edit. * User should be able to modify form data. * User should be able to submit edited form. | * User is asked for confirmation to edit. * User can modify form data. * User can submit edited form. | Passed |

Table 14

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module Name** | Delete Appointment | | | |
| **Test Case ID** | TC009 | | | |
| **Tester Name** | L Senamela | | | |
| **Test Case Description** | Delete an existing appointment. | | | |
| **Pre-requisite** | Patient must have an appointment. | | | |
| **Test Steps** | **Test Input** | **Expected Results** | **Actual Result** | **Status** |
| 1. Go to appointments list. 2. Click delete button. | N\A | User should be asked for confirmation to delete. | User is asked for confirmation to delete. | Passed |

Table 15

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module Name** | Create Prescriptions. | | | |
| **Test Case ID** | TC010 | | | |
| **Tester Name** | L Senamela | | | |
| **Test Case Description** | Create a digital prescription. | | | |
| **Pre-requisite** | Patient must be registered. | | | |
| **Test Steps** | **Test Input** | **Expected Results** | **Actual Result** | **Status** |
| 1. Go to prescriptions. 2. Fill in the form. 3. Submit the form. | Medication Name  Medication Purpose Medication Dosage Medication Frequency  Physician Signature | * Doctor should be able to see the form and fill it in. * Doctor should be able to submit the prescription. | * Doctor can see the form and fill it in. * Doctor can submit the prescription. | Passed |

Table 16

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module Name** | Edit Prescription | | | |
| **Test Case ID** | TC011 | | | |
| **Tester Name** | L Senamela | | | |
| **Test Case Description** | Edit an existing prescription. | | | |
| **Pre-requisite** | User must have a prescription. | | | |
| **Test Steps** | **Test Input** | **Expected Results** | **Actual Result** | **Status** |
| 1. Go to appointments list. 2. Click edit button. 3. Modify data in the form. 4. Submit the form. | Appointment Date  Appointment Time  Appointment Details  First Name  Last Name | * Doctor should be asked for confirmation to edit. * Doctor should be able to modify form data. * Doctor should be able to submit edited form. | * Doctor is asked for confirmation to edit. * Doctor can modify form data. * Doctor can submit edited form. | Passed |

Table 17

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module Name** | Pharmacy Inventory Management | | | |
| **Test Case ID** | TC012 | | | |
| **Tester Name** | L Senamela | | | |
| **Test Case Description** | Check medication availability. | | | |
| **Pre-requisite** | N\A | | | |
| **Test Steps** | **Test Input** | **Expected Results** | **Actual Result** | **Status** |
| 1. Go to medication availability. 2. Enter medication name. 3. Submit the form. | Medication Name | * Pharmacist should be asked for confirmation to submit. * Pharmacist should be able to submit the form. * System should display availability status. | * Pharmacist is asked for confirmation to submit. * Pharmacist can submit the form. * Availability status is displayed. | Passed |

Table 18

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module Name** | Pharmacy Inventory Management | | | |
| **Test Case ID** | TC013 | | | |
| **Tester Name** | L Senamela | | | |
| **Test Case Description** | Update inventory. | | | |
| **Pre-requisite** | User must have a prescription. | | | |
| **Test Steps** | **Test Input** | **Expected Results** | **Actual Result** | **Status** |
| 1. Go to medication list. 2. Fill data in the form. 3. Submit the form. | Medication Name  Quantity | * Pharmacist should be asked for confirmation to edit. * Pharmacist should be able to enter form data. * Doctor should be able to submit the form. | * Pharmacist is asked for confirmation to edit. * Pharmacist can enter form data. * Pharmacist can submit the form. | Passed |

Table 19

### Test Execution

* Define the sequence of test execution.
* Assign responsibilities to team members for executing tests.
* Record the results of each test case.

### Defect Management

* Log any defects found during testing.
* Prioritize and address defects based on severity.
* Re-test to ensure defects are resolved.

### Acceptance Criteria

* All test cases must pass.
* No critical defects remain open.
* System meets all specified requirements.

This testing plan is designed to ensure that MediVault is reliable, secure, and user-friendly, meeting all outlined requirements and providing a seamless experience for users.

# Recommendations

### Patient Self-Service Portal

Adding a portal where patients can log in and do things like view their health records, update personal info, and check appointment dates could make the system more useful. Patients could also access information about their health conditions, helping them stay more involved in their own care.

### Mobile App

Developing a MediVault mobile app would make it easier for doctors and patients to use the system on the go. It could include features like booking appointments, getting digital prescriptions, and messaging between patients and healthcare providers. Having it on a mobile device would increase accessibility and ease of use.

### Better Data Analysis

Adding more data analysis tools could help hospitals and clinics get a clearer picture of what's going on with patients, like what health issues are most common or how resources are being used. This could lead to better decision-making and more efficient use of hospital resources.

### Blockchain for Security

Using blockchain technology could make patient data more secure. Blockchain creates a system where data can’t be changed by unauthorized users, ensuring that records stay safe and reliable. This would help build more trust in how patient data is handled.

### Support for Multiple Languages

To make MediVault easier for people from different backgrounds, it would be helpful to include support for multiple languages. This way, both patients and healthcare workers can use the system more easily, no matter what language they speak.

### Integration with Other Systems

MediVault should work well with other healthcare systems so that hospitals and clinics can securely share patient info. This would be especially useful when patients switch between different healthcare providers, ensuring that their records are always up to date.

### Automated Notifications

Adding automatic notifications could improve communication within the system. For instance, doctors could be notified when lab results are ready, or patients could get reminders about their upcoming appointments or prescription renewals. This would help reduce missed appointments and keep things running smoothly.

### Regular Security Audits

The system should go through regular security checks to catch any potential issues and make sure it meets data protection standards. Doing these audits would help keep patient data safe and protect against new security threats.