** The Islamia University of Bahawalpur**

**Project Title**

**A project presented to**

**Department of Computer Science, IUB Bahawalpur**

**In partial fulfillment**

**of the requirement for the degree of**

***Bachelor of Science in Computer Science (20xx-20xx)***

**By**

**Student Name**

**Roll No**

**Session**

**DECLARATION**

We hereby declare that this software, neither whole nor as a part has been copied out from any source. It is further declared that we have developed this software and accompanied report entirely based on our personal efforts. If any part of this project is proved to be copied out from any source or found to be reproduction of some other. We will stand by the consequences. No Portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

Student Name

-----------------------------------

**CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (CS) “Project title” was developed by

STUDENT NAME , Roll No, SESSION under the supervision of “SUPERVISOR NAME” and that in (his/her) opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Computer Sciences.

Supervisor

-----------------------------------

External Examiner

------------------------------------

Chairman Department of Computer Science

-----------------------------------

**Acknowledgement**

We extend sincere thanks to our supervisor, [Supervisor's Name], for their invaluable guidance and support throughout the Hospital Management System (HMS) project. Special appreciation goes to the dedicated development team, peers, and families whose efforts and encouragement played a vital role in the project's success. We also acknowledge the insights provided by healthcare professionals and express gratitude to the academic institution for its support. The completion of the HMS project is a result of collaborative efforts, expertise, and encouragement from various stakeholders.

Student Name

-----------------------------

**Abbreviations**

Provide a list of all abbreviation used in the document.

|  |  |
| --- | --- |
| **SRS** | Software Require Specification |
| **HMS** | Hospital Management System |
| **PC** | Personal Computer |
| **SDD** | Software Design Document |
| **UC** | Use Case |
| **BR** | Business Rule |
| **PRE** | Precondition |
| **POST** | Postcondition |
| **FR** | Functional Requirement |
| **RAM** | Random-Access Memory |
| **DFD** | Data Flow Diagram |

Contents

[1 INTRODUCTION 7](#_Toc153203979)

[1.1 Brief 7](#_Toc153203980)

[1.2 Relevance to Course Modules 7](#_Toc153203981)

[1.3 Project Background 8](#_Toc153203982)

[1.4 Related Material and Literature 8](#_Toc153203983)

[1.6 Methodology and Software Lifecycle for This Project 9](#_Toc153203984)

[1.6.1 Rationale behind Selected Methodology 9](#_Toc153203985)

[2. Problem Definition 10](#_Toc153203986)

[2.1 Problem Statement 10](#_Toc153203987)

[2.2 Deliverables and Development Requirements 11](#_Toc153203988)

[2.3 Current System 11](#_Toc153203989)

[3 Requirement identifying technique 12](#_Toc153203990)

[3.1 SPECIFIC REQUIREMENTS 12](#_Toc153203991)

[3.1.1 External Interface Requirements 12](#_Toc153203992)

[3.1.2 Hardware Interfaces 13](#_Toc153203993)

[3.1.3 Software Interface 13](#_Toc153203994)

[3.1.4 Microsoft SQL Server 14](#_Toc153203995)

[3.2 Use Case Diagram 14](#_Toc153203996)

[3.2.1 Admin Use Case 15](#_Toc153203997)

[3.2.2 Use Case for Staff/Employee 16](#_Toc153203998)

[3.3 Use Case Description 17](#_Toc153203999)

[3.3.1 Use Case Description for Admin 17](#_Toc153204000)

[3.3.2 Use Case Description for Staff/Employee 18](#_Toc153204001)

[3.3.3 Use Case Description for Doctor 19](#_Toc153204002)

[4 Design and Architecture 20](#_Toc153204003)

[4.1 ARCHITECTURAL DESIGN 20](#_Toc153204004)

[4.1.1 Architectural Design 20](#_Toc153204005)

[21](#_Toc153204006)

[4.1.2 Decomposition Description 21](#_Toc153204007)

[4.2 Process flow 22](#_Toc153204008)

[4.3 Design models 23](#_Toc153204009)

[4.1 Class Diagram 23](#_Toc153204010)

[4.2 Sequence Diagram 24](#_Toc153204011)

[4.3 Data flow diagrams (DFD): 25](#_Toc153204012)

[4.3.1 Level Zero Data Flow Diagram 26](#_Toc153204013)

[4.3.2 Level 1 Data Flow Diagram 27](#_Toc153204014)

[4.3.3 Level 2 Data Flow Diagram 28](#_Toc153204015)

[5 Implementation 29](#_Toc153204016)

[5.1 Algorithm 29](#_Toc153204017)

[5.1.1 Patient Admission Algorithm 29](#_Toc153204018)

[5.1.2 Prescription Management Algorithm 29](#_Toc153204019)

[5.1.3 Billing System Algorithm 30](#_Toc153204020)

[5.2 External APIs 31](#_Toc153204021)

[5.3 User Interface 32](#_Toc153204022)

[5.3.1 Dashboard 32](#_Toc153204023)

[5.3.2 Patient Registration Form 32](#_Toc153204024)

[5.3.3 Appointment Scheduling Interface 32](#_Toc153204025)

[5.3.4 Billing and Payment Interface 32](#_Toc153204026)

[5.3.5 Inventory Management Interface 32](#_Toc153204027)

[5.4 Screen Images 33](#_Toc153204028)

[6 Testing and Evaluation 36](#_Toc153204029)

[6.1 Manual Testing 36](#_Toc153204030)

[6.1.1 System Testing 36](#_Toc153204031)

[6.1.2 Unit Testing 36](#_Toc153204032)

[6.1.3 Functional Testing 38](#_Toc153204033)

[6.1.4 Integration Testing 40](#_Toc153204034)

[7 Conclusion and Future Work 41](#_Toc153204035)

[7.1 Conclusion 41](#_Toc153204036)

[7.2 Future Work 41](#_Toc153204037)

[8. References 42](#_Toc153204038)

# 1 INTRODUCTION

The Hospital Management System (HMS) is a sophisticated software solution designed to automate and streamline both administrative and clinical processes in healthcare institutions. It focuses on efficiently managing patient records, billing, and inventory to enhance overall hospital efficiency, reduce paperwork, and improve patient care. The system aims to provide a user-friendly interface for healthcare professionals and administrative staff, fostering a more organized and efficient healthcare environment. The primary purpose is to centralize and integrate operations, including patient information management, medical record maintenance, billing, and inventory oversight. By minimizing manual paperwork and improving data accuracy, the HMS contributes to better patient care and increased productivity within the healthcare institution. The scope encompasses patient data entry, billing, and diagnosis information recording, addressing the challenges associated with manual handling and ensuring a more streamlined and integrated approach.

## Brief

The Hospital Management System (HMS) project serves as an advanced software solution, aiming to digitize and optimize administrative and clinical processes within healthcare institutions. The project outcome encompasses a streamlined and automated approach to patient record management, billing, and inventory, contributing to enhanced overall hospital efficiency and improved patient care. Employing a user-friendly interface, the HMS utilizes tools such as Python, TensorFlow, OpenCV, Numpy, and Keras to achieve its goals. The methodology involves a systematic approach to centralizing operations, reducing manual paperwork, and fostering a more organized healthcare environment. Throughout the report, discussions emphasize the significance of data accuracy, improved operational efficiency, and positive impacts on human-robot interactions within the healthcare setting.

## Relevance to Course Modules

The Hospital Management System (HMS) project aligns closely with the courses undertaken during the Bachelor of Science (BS) program. It encompasses a multidisciplinary approach, incorporating knowledge from courses such as Database Management Systems, Software Engineering, and System Analysis and Design. The application of programming languages like Python, utilization of machine learning frameworks such as TensorFlow and Keras, and integration of OpenCV and Numpy demonstrate the practical implementation of theoretical concepts from programming, data management, and artificial intelligence courses. The project's relevance extends to areas of system design, emphasizing the application of software engineering principles in developing a comprehensive and efficient healthcare management solution. Through this alignment, the HMS project bridges theoretical learning with practical implementation, reinforcing key concepts from various BS courses.

## Project Background

The project background provides insight into the genesis of the Hospital Management System (HMS). In the healthcare landscape, efficient management of administrative and clinical processes is crucial for streamlined operations. Recognizing the challenges posed by manual record-keeping, billing, and inventory management in healthcare institutions, the HMS project emerged as a solution to digitize and optimize these processes. The idea behind the project is to create a sophisticated software system that centralizes patient information, automates billing procedures, and facilitates inventory control. By leveraging technology, the HMS aims to enhance overall hospital efficiency, reduce paperwork, and contribute to improved patient care. The project addresses the need for a comprehensive and user-friendly system to manage the complex operations within healthcare facilities.

## 1.4 Related Material and Literature

The literature review for the Hospital Management System (HMS) project delves into current trends, research, and products within the healthcare and information technology domains. Existing studies highlight the increasing importance of digitizing healthcare processes for efficiency and accuracy. Relevant literature emphasizes the adoption of advanced technologies, such as machine learning and data analytics, in healthcare management systems. Research on electronic health records, inventory management, and patient care optimization provides insights into best practices. Notable products in the market showcase innovative solutions for patient information management, billing, and inventory control. By examining this literature, the HMS project integrates contemporary approaches and technologies, ensuring alignment with current trends and contributing to the ongoing discourse on optimizing healthcare through digital solutions.

## 1.6 Methodology and Software Lifecycle for This Project

For the Hospital Management System (HMS) project, a thoughtful selection of methodology and Software Development Life Cycle (SDLC) model was essential. The chosen methodology is the backbone of the project's organizational structure and guides the development process. In this case, a detailed discussion follows on the selected methodology, emphasizing its rationale and suitability for the project's objectives. Additionally, the chosen SDLC model is outlined, explaining how it aligns with the project's needs and facilitates a systematic approach to development. The discussion delves into the key reasons behind opting for a specific methodology and SDLC model, providing a clear understanding of the project's development framework. This information lays the foundation for the subsequent phases of the project, ensuring a structured and effective approach to HMS development.

### 1.6.1 Rationale behind Selected Methodology

The choice of methodology, specifically the Structural and Object-Oriented methodology, for the Hospital Management System (HMS) project is driven by its inherent advantages in managing complex healthcare systems. The structural methodology provides a systematic approach to design and development, ensuring clarity in system architecture and enhancing maintainability. The Object-Oriented methodology, on the other hand, allows for encapsulation of data and functionalities, promoting modular design and reusability of code components.

This selection aligns with the project's requirements for a scalable and adaptable system, accommodating the intricate nature of healthcare data and processes. The combination of these methodologies enables a comprehensive and organized approach to system development, crucial for a sophisticated application like HMS.

In terms of the Software Development Life Cycle (SDLC), the project opted for an iterative and incremental model. This choice is driven by the dynamic nature of healthcare requirements, where continuous feedback and flexibility in adapting to evolving needs are paramount. The iterative and incremental model ensures that the system can be refined through multiple cycles, allowing for regular assessments, adjustments, and enhancements. This approach is well-suited for the iterative nature of healthcare system development, providing a systematic and adaptable framework for the entire software life cycle.

# 2. Problem Definition

This section outlines the challenges encountered in the current manual Hospital Management System (HMS), emphasizing the imperative need for an automated system. The identified issues impact both healthcare professionals and patients, necessitating a transition to a computerized HMS for enhanced efficiency, time savings, and optimal resource utilization.

## 2.1 Problem Statement

Within the existing manual HMS, healthcare professionals and patients grapple with significant challenges, underscoring the demand for an automated solution. The identified problems are as follows:

1. **Time-Consuming Patient Information Retrieval:**
   * ***Issue****:* Healthcare professionals spend a substantial amount of time retrieving specific patient information.
   * ***Impact****:* Inefficiency and potential frustration among healthcare staff.
2. **Challenges in Medical Record Access:**
   * ***Issue****:* Healthcare professionals face difficulties in accessing real-time medical records.
   * ***Impact****:* Hindered hospital operations and potential delays in patient services.
3. **Verification of Patient Admission:**
   * ***Issue****:* Manual verification of patient admissions during new registrations poses challenges.
   * ***Impact****:* Difficulty in ensuring accurate admissions and potential errors in the process.
4. **Difficulties in Appointment Scheduling:**
   * ***Issue****:* Manual appointment scheduling processes present challenges.
   * ***Impact****:* Inconvenience for patients and limitations in service offerings.
5. **Manual Billing and Payment Processing:**
   * ***Issue****:* Manual calculation of treatment costs, room charges, and payment processing.
   * ***Impact****:* Time-consuming processes and potential errors in financial transactions.
6. **Inventory Management Challenges:**
   * ***Issue****:* Manual tracking of hospital inventory and updates for consumed medical supplies.
   * ***Impact****:* Inefficient inventory control and potential stockouts.
7. **Laboratory Results Retrieval Issues:**
   * ***Issue****:* Manual retrieval of laboratory results for diagnostics.
   * ***Impact****:* Delayed diagnostics and potential impacts on patient care.
8. **Limited Automation in Data Accuracy:**
   * ***Issue****:* Lack of automated processes affects data accuracy in hospital operations.
   * ***Impact****:* Risks to data integrity and potential errors in decision-making.

## 2.2 Deliverables and Development Requirements

The envisioned automated HMS aims to tackle the identified challenges, delivering the following key outcomes and necessitating specific development requirements:

**Deliverables**:

* Automated Patient Information Retrieval
* Real-time Access to Medical Records
* Streamlined Patient Admission Verification
* Online Appointment Scheduling System
* Automated Billing and Payment Processing
* Efficient Inventory Management System
* Automated Laboratory Results Retrieval
* Enhanced Automation for Data Accuracy

**Development Requirements:**

* Selection of Programming Language (e.g., Python, Java)
* Implementation of a Robust Database Management System
* Design of a User-Friendly Interface
* Integration of Secure Authentication Protocols

## 2.3 Current System

The existing manual hospital management system lacks the required efficiency and automation to address the identified challenges. A transition to an automated HMS is crucial for optimizing healthcare operations and enhancing the overall patient experience.

# 3 Requirement identifying technique

## 3.1 SPECIFIC REQUIREMENTS

### 3.1.1 External Interface Requirements

**System Interfaces**

The system interface is like forms which are to be filled to enter data into the system. The employee interacts with the system through system interfaces.

**User Interfaces**

The system is designed in such a way that the users can post reviews, ratings and complaints in the sections which are separately provided for it. The vendors are given provision to upload the image of their products. There is a section for the vendors and the customers to chat through a bot with smart technology. If it is a new question, then the user can post the question which will be answered within short span of time. The images which are uploaded should be of the .jpeg or .png format. There are also options for user to change the website language according to his/her wish.

**Home Page**

It is the first interface that appears on the screen when the application is being loaded. This interface displays the name of the application and some other information about the software. The page consists of logins that exist for several other levels in the application. They consist of administrator, supervisor and staff login.

**Admin Login Menu**

After a successful login supplying the correct username and password, it opens into another page where the activities of the admin module are fully stipulated. The admin controls all the major activity is of this application. Activities such as Add staff, Add Drugs, Change Username and password, manage drugs, View sales etc. can only be controlled by the admin. Furthermore, the Admin has control over the supervisor and all other user of the application.

**Staff Login**

The staff is responsible for updating the list of patients in the system. The staff can also engage in buying, selling of drugs and printing the reports. The staff is required to provide a valid username and password in other to be able to perform its activities

### 3.1.2 Hardware Interfaces

The system that going to developed is requires some hardware specifications so that it can be utilized efficiently. The computer on which this system is going to be used must have the specified hardware components. All the hardware needed here are generally the basic configuration of a typical office computer. A list of the hardware requirement used in the system given below:

**Minimum Configuration**

Minimum configuration for the system is described below:

* + Keyboard.
  + Mouse.
  + A printer.
  + 2.5 Hz Pentium processor.
  + 500 MB SD RAM.
  + Minimum 80 GB Hard Drive.
  + Monitor Display. Or LCD or LED.

**Recommended Configuration**

The configuration mention above is to use the system just effectively. The configuration described below can be used to have better and optimum result during the working of the system.

* + 2GHz core 2 duo.
  + Keyboard.
  + Mouse.
  + Crystal Report
  + 2GB or more SD RAM.
  + Minimum 80 or more GB Hard drive.
  + Color Monitor.
  + 500 V. UPS. (can be used in case of power failure)

### 3.1.3 Software Interface

The system will have form-based interface which allow an employee to enter a specific password assigned to it and username. After this he will open different screens for different purposes. This is customized software in the form of a package. The software which I have chosen for the system are explained below. These are the updated version compatible with the system.

* Operating system.
* Window 10 (32 or 64 bit).
* Visual studio 2019 Enterprise.
* .Net Framework 4.6 or higher.
* Notepad or MS word installed.

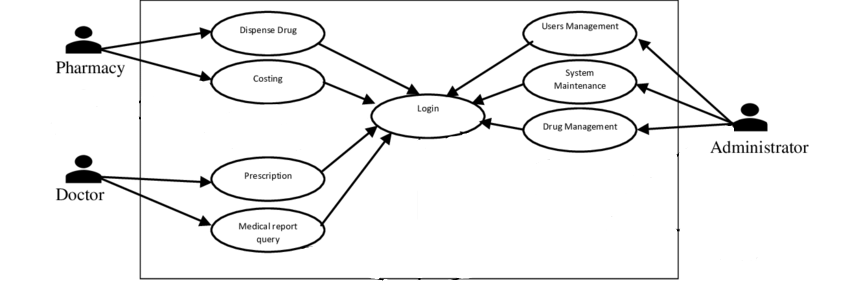
### 3.1.4 Microsoft SQL Server

SQL server is used to connect with the database using OLEDB and ODBC data connection. Many backend software is available for data storage such as MS access, Sybase, oracle and my SQL. Here we are choosing Microsoft SQL Server as database for the system. It has best features supporting the .net framework. Some features are:

* Multiple platforms supported by the system.
* Integration with Windows 8.1 or 10 or 7.
* Integration with MS .NET Enterprise Servers.
* Scalability.
* Replication.
* Centralized Management.
* Reliability.

## 3.2 Use Case Diagram

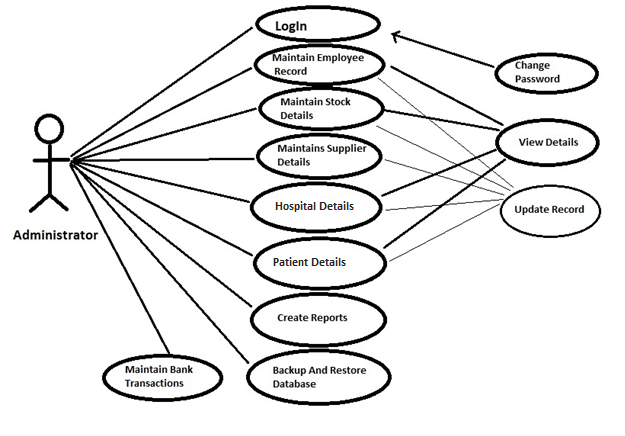
The Use Case Diagram for the Hospital Management System (HMS) provides a visual representation of the various interactions and functionalities within the system. Actors, including the Receptionist, Patient, and Calendar System, are illustrated, depicting their roles and relationships with distinct use cases. Key use cases such as Scheduling Patient Appointments, Updating Patient Information, Checking Appointment Availability, and Notifying Patients are outlined to offer a comprehensive overview of the system's capabilities. This diagram serves as a valuable tool for understanding the high-level functionalities and interactions within the HMS, aiding in system design and communication among stakeholders.



**Use Case Diagram 1**

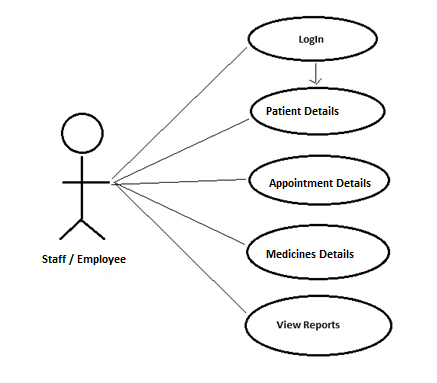
### 3.2.1 Admin Use Case

In the Hospital Management System (HMS), the Administrator Management use case is essential for overseeing the system's overall functionality. The Administrator, as the primary actor, holds responsibilities such as user management, system configuration, data security, and system maintenance. By managing user accounts, configuring system settings, ensuring data security and privacy, and handling routine maintenance tasks, the Administrator plays a crucial role in maintaining the integrity and security of patient data. Additionally, the generation of reports and analytics falls under the Administrator's purview, contributing to informed decision-making. This use case ensures that the HMS operates seamlessly, aligns with organizational policies, and remains secure.



**Use Case Diagram 2**

### 3.2.2 Use Case for Staff/Employee

The Staff Interaction use case caters to the daily operational needs of healthcare professionals within the HMS. Staff members, as primary actors, engage with the system for patient management, appointment scheduling, billing and payments, prescription management, and resource allocation. This use case empowers staff with the tools needed for efficient patient care, streamlined appointment processes, and effective financial management. Staff members can also utilize the system for internal communication, collaboration on patient care, and accessing training modules for professional development. Together, the Administrator and Staff Interaction use cases contribute to the holistic functioning of the HMS, ensuring it meets the diverse needs of both administrators and healthcare professionals.

**Use Case Diagram 3**

## 3.3 Use Case Description

### 3.3.1 Use Case Description for Admin

|  |  |
| --- | --- |
| **Use Case ID:** | UC-Admin-1 |
| **Use Case Name:** | Administer System Configuration |
| **Actors:** | Primary Actor: Administrator  Secondary Actors: Database Management System (DBMS |
| **Description:** | The Administrator accesses the Hospital Management System (HMS) to configure and administer system settings. This includes managing user roles and permissions, configuring data backup schedules, and ensuring data security protocols are in place. The Administrator may also oversee integrations with external systems and manage the overall system configuration to align with organizational requirements. |
| **Trigger:** | The Administrator initiates the use case by indicating the need to configure or administer system settings. |
| **Preconditions:** | PRE-1: The Administrator is logged into the HMS.  PRE-2: The HMS is in an operational state. |
| **Postconditions:** | POST-1: System configuration changes are saved and implemented.  POST-2: User roles and permissions are updated if modified.  POST-3: Data security protocols are in effect.  POST-4: Integration settings, if modified, are applied. |
| **Normal Flow:** | 1. The Administrator accesses the system configuration panel. 2. The Administrator selects the category of configuration to be modified (e.g., user roles, data backup). 3. The Administrator makes necessary changes. 4. The Administrator confirms the changes, triggering the system to apply the modifications. 5. The system displays a confirmation message indicating successful configuration changes. |
| **Alternative Flows: [Alternative Flow 1 – Not in Network]** | 1. The Administrator selects the user roles category. 2. The Administrator modifies user roles and permissions. 3. The Administrator confirms changes, and the system updates user roles accordingly. |
| **Exceptions:**  **Configuration Error:** | If there is an error in the configuration changes, the system displays an error message.  The Administrator can choose to correct the error and reconfirm the changes or cancel the process. |
| **Business Rules** | BR-1: Configuration changes must comply with organizational policies.  BR-2: User roles and permissions must adhere to the principle of least privilege. |
| **Assumptions:** | Assumption 1: The Administrator has the necessary knowledge to make informed system configuration decisions.  Assumption 2: The HMS is in a stable operational state during the configuration process. |

### 3.3.2 Use Case Description for Staff/Employee

|  |  |
| --- | --- |
| **Use Case ID:** | UC-Staff-1 |
| **Use Case Name:** | Perform Daily Tasks |
| **Actors:** | Primary Actor: Staff Member  Secondary Actors: None |
| **Description:** | Staff members utilize the Hospital Management System (HMS) to perform daily tasks related to patient care, appointment scheduling, and updating patient records. This includes recording vital signs, scheduling appointments, and updating patient information. Staff members interact with the system to ensure accurate and up-to-date patient records, contributing to efficient healthcare service delivery. |
| **Trigger:** | The Staff Member initiates the use case when there are daily tasks to be performed, such as recording patient information or scheduling appointments. |
| **Preconditions:** | PRE-1: The Staff Member is logged into the HMS.  PRE-2: The HMS is in an operational state. |
| **Postconditions:** | PRE-1: The Staff Member is logged into the HMS.  PRE-2: The HMS is in an operational state. |
| **Normal Flow:** | 1. The Staff Member accesses the patient records module. 2. The Staff Member selects a patient file to update. 3. The Staff Member records or updates vital signs, symptoms, or any relevant patient information. 4. The Staff Member accesses the appointment scheduling module. 5. The Staff Member schedules or modifies appointments based on patient needs. 6. The Staff Member confirms the updates, and the system reflects the changes. |
| **Alternative Flows: Urgent Appointment:** | 1. If an urgent appointment is required, the Staff Member prioritizes scheduling. 2. The Staff Member updates the appointment details, ensuring timely patient care. |
| **Exceptions:**  **Data Entry Error:** | If there is an error in data entry, the system prompts the Staff Member to review and correct the information.  The Staff Member corrects the error and confirms the updates. |
| **Business Rules** | BR-1: Patient records must be updated in real-time to ensure accuracy.  BR-2: Appointment scheduling follows a prioritization algorithm based on urgency and availability. |
| **Assumptions:** | Assumption 1: Staff members have received training on using the HMS for daily tasks.  Assumption 2: The HMS is in a stable operational state during daily task execution. |

|  |  |
| --- | --- |
| **Use Case ID:** | UC-Doctor-1 |
| **Use Case Name:** | Review Patient Health Information |
| **Actors:** | Primary Actor: Doctor  Secondary Actors: None |
| **Description:** | Doctors utilize the Hospital Management System (HMS) to review and analyze the health information of their assigned patients. This use case enables doctors to access patient records, view medical history, and make informed decisions regarding diagnoses and treatment plans. |
| **Trigger:** | The Doctor initiates the use case when they need to review the health information of a specific patient. |
| **Preconditions:** | PRE-1: The Doctor is registered in the HMS.  PRE-2: The Doctor is logged into the system.  PRE-3: The Doctor is assigned to the patient whose information they want to review |
| **Postconditions:** | POST-1: The Doctor successfully reviews the patient's health information.  POST-2: Any annotations or updates made by the Doctor are saved in the patient's records. |
| **Normal Flow:** | 1. The Doctor logs into the HMS using valid credentials. 2. The Doctor navigates to the "Patient List" or similar section. 3. The system displays a list of patients assigned to the Doctor. 4. The Doctor selects a specific patient from the list to review their health information. 5. The system presents a summary of the patient's medical records, including current medications, test results, and relevant medical history. 6. The Doctor can access detailed information by selecting specific categories such as lab results, imaging reports, or treatment plans. |
| **Alternative Flows: Update Patient Records:** | 1. If the Doctor identifies the need to update the patient's records (e.g., adding new test results or updating medication), they can initiate the update process. 2. The Doctor follows the prompts to add or modify information. 3. The updated information is saved in the patient's records. |
| **Exceptions:**  **Patient Not Assigned:** | If the Doctor attempts to access the information of a patient not assigned to them, the system notifies the Doctor.  The Doctor is prompted to select a patient from their assigned list. |
| **Business Rules** | BR-1: Doctors have both read and write access to patient information within their assigned patient list.  BR-2: Any modifications or annotations made by the Doctor are logged for audit purposes. |
| **Assumptions:** | Assumption 1: Doctors are assigned patients based on their specialty or workload.  Assumption 2: The HMS provides a user-friendly interface for Doctors to quickly access and review patient information. |

### 3.3.3 Use Case Description for Doctor

# 4 Design and Architecture

The Hospital Management System (HMS) is a digitized healthcare solution designed for efficient administration and clinical operations. It employs Object-Oriented Programming (OOP) principles, using C# in Visual Studio to create a user-friendly interface. The system's key functionalities include patient management, appointment scheduling, medical records maintenance, billing, and inventory management. With a modular design for easy scalability, the HMS aims to streamline healthcare processes, reduce paperwork, and enhance patient care. It serves as a centralized hub for stakeholders, promoting data accuracy and tailored access for different user roles. The project responds to the evolving complexities of healthcare data management, providing a robust, secure, and adaptable solution for healthcare institutions.

## 4.1 ARCHITECTURAL DESIGN

The Hospital Management System (HMS) adopts a modular program structure to ensure a scalable and maintainable architecture. The system comprises key subsystems that collaborate to achieve comprehensive functionality.

### 4.1.1 Architectural Design

|  |  |
| --- | --- |
| Notation | Description |
|  | This shape is called actor. Its play role in a system. Actor play role as a human users or other subject and external hardware. One actor may be played different activities. Its show with stick man icon. |
|  | Ellipse shape is used as use case. Its show the functionality of any system.  Each use case must have a name. |
|  | This shape shows association. It shows the relationship between actor and use case. Through this use case show interaction. |

## 

**User Case Diagram 1**

|  |  |
| --- | --- |
| Use Case | UC1 |
| Use case name | How system will work. |
| actor | Admin, Doctor, Pharmacist. |
| Description | Admin can save edit update and delete records |

### 

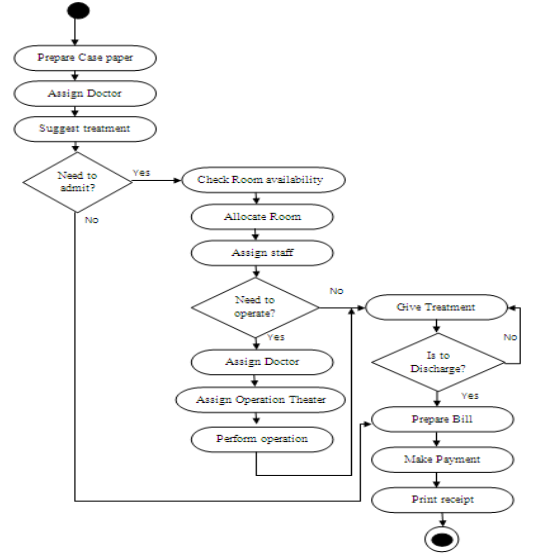
### 4.1.2 Decomposition Description

The decomposition of the Hospital Management System (HMS) was executed to enhance the system's manageability, security, and scalability. By breaking down the system into modular components like Patient Management, Appointment Scheduling, and Billing, each module addresses a specific concern, ensuring clear separation of functionalities. Encapsulation is employed to secure sensitive data within modules, promoting a controlled access approach. The modular design allows for scalability, accommodating future enhancements without disrupting the entire system. This approach improves system maintainability, as updates to one module do not necessitate changes in others, facilitating efficient collaboration and reducing overall complexity. The decomposition strategy is a deliberate effort to streamline development, enhance security, and support the evolving needs of healthcare management.

## 4.2 Process flow

In the Hospital Management System (HMS), the major processes are briefly represented through an activity diagram. The patient registration process is initiated by the receptionist, gathering essential information and creating a unique patient ID. Appointment scheduling, facilitated by healthcare professionals, involves selecting available time slots and linking them to patient records. Following patient appointments, medical records are updated with diagnoses, treatments, and test results. The billing process is triggered by completed medical services, generating invoices based on the services provided, which subsequently impacts inventory management. Report generation, both periodic and on-demand, compiles information on appointments, billing, and inventory. Emergency handling involves quick patient registration, appointment scheduling, and medical records updates during critical situations. Routine tasks like system maintenance, end-of-day procedures, and system shutdown are seamlessly integrated into the HMS workflow, ensuring efficient and organized hospital operations.

**Activity Diagram**



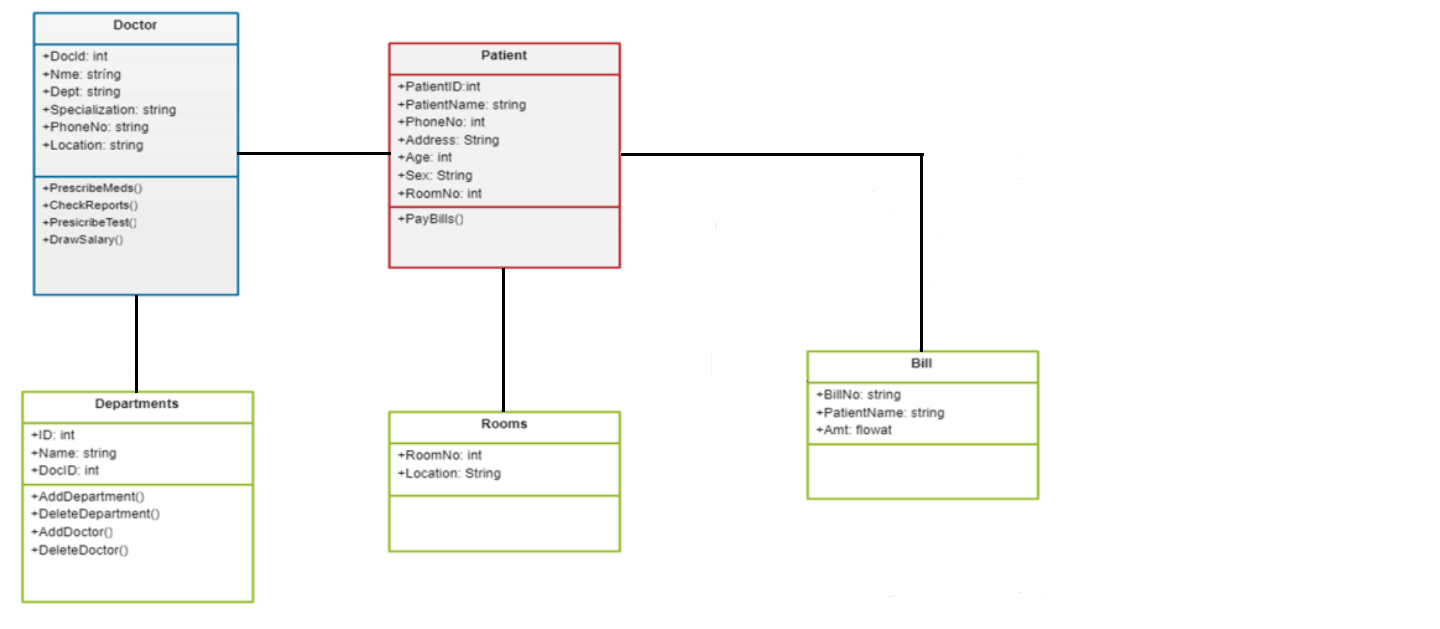
**Activity Diagram**

## 4.3 Design models

The design models for the Hospital Management System (HMS) encompass a range of visual representations tailored to capture distinct facets of the system's architecture and functionality. The Class Diagram delineates the object-oriented structure, illustrating classes, relationships, and inheritance. Sequence Diagrams offer insights into the dynamic interactions between system components during processes such as patient registration and billing. State Transition Diagrams illuminate the various states and transitions within the HMS, particularly focusing on appointment status changes. Data Flow Diagrams provide a comprehensive view of information flow, showcasing processes, data stores, and external entities.

## Class Diagram

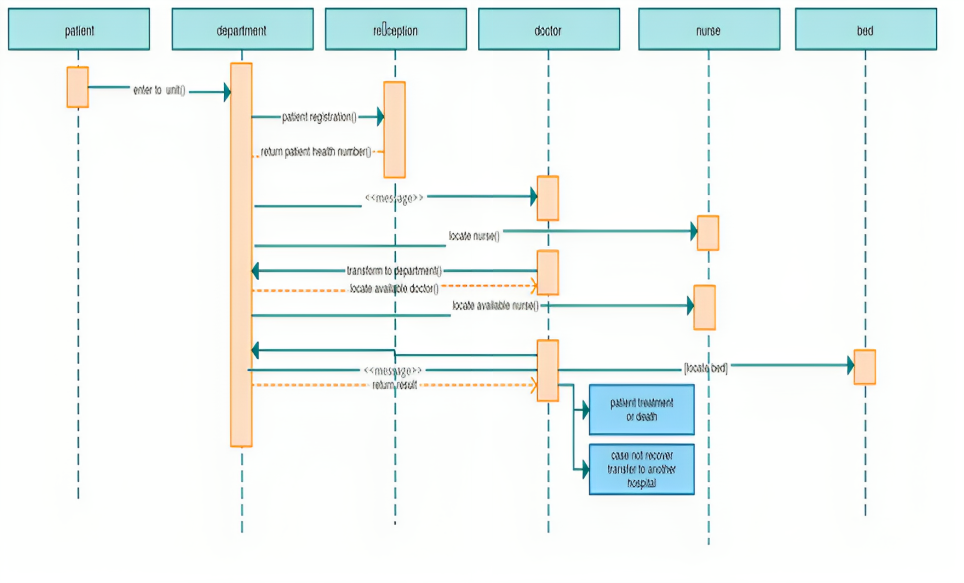
The class diagram illustrates the object-oriented design of the HMS, depicting classes like Patient, Appointment, and Billing. Relationships such as aggregation, composition, and inheritance are used to represent associations and hierarchies between classes. A Class is a category or group of things that has similar attributes and common behavior. A Rectangle is the icon that represents the class it is divided into three areas. The upper most area contains the name, the middle; area contains the attributes and the lowest areas show the operations. Class diagrams provides the representation that developers work from. Class diagrams help on the analysis side, too.



**Class Diagram**

## Sequence Diagram

A Sequence Diagram is an interaction diagram that emphasis the time ordering of messages; a collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages. Sequence diagrams and collaboration diagrams are isomorphic, meaning that you can take one and transform it into the other.



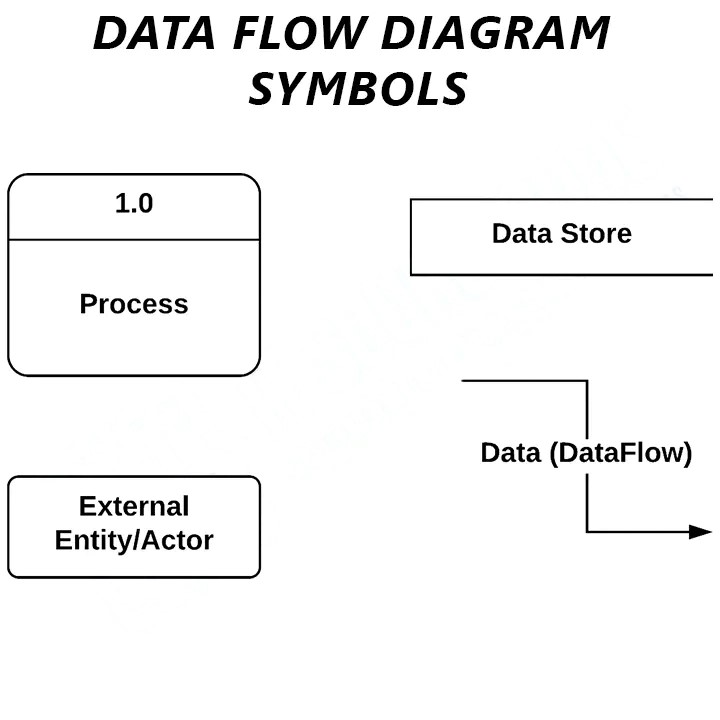
**Sequential Diagram**

## Data flow diagrams (DFD):

DFDs provide a visual representation of how data flows through the system. In the HMS, a multi-level DFD can showcase processes like Patient Registration, Appointment Scheduling, and Billing, along with data stores such as Patient Database and Inventory.

This DFD Diagram for the hospital management system uses defined symbols that present the system’s processes.

These symbols are the following:

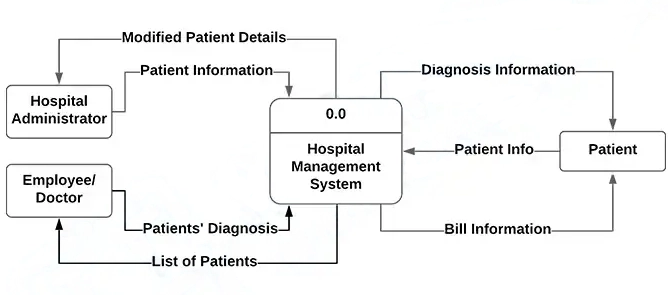


**Data Flow Diagram**

* **External Entities** are the entry and exit points for data entering and leaving the system. Entities are referred to as terminators, sources, sinks, and actors.
* The **process** is the portion of DFD that modifies and generates output from data.
* **A Data Store (database)** is a table that stores the files or repositories for future use.
* **Data Flow** is the flow of data between external entities, processes, and data stores.

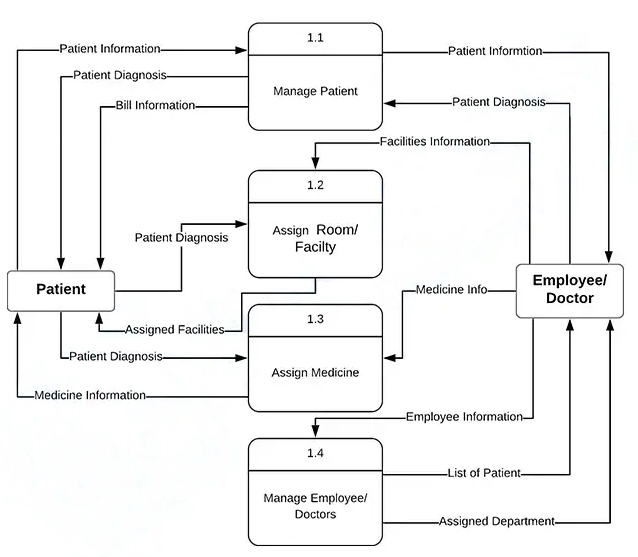
### 4.3.1 Level Zero Data Flow Diagram

The Zero Level DFD for hospital management system depicts the overview of whole hospital management system. It is supposed to be an abstract view of overall system. This is also called as context diagram for hospital management system in which entire system is represented as single process with its relationship with external entities such as admin, staff, doctor etc.



**Zero Level Data Flow Diagram**

### 4.3.2 Level 1 Data Flow Diagram

The first level DFD (1St Level) of Hospital management system shows more details of processing. Level 1 DFD list all the major sub processes that makes the entire system. It also identifies data store of hospital master data that contains all records of patient, doctor, staff, room, treatment etc. that will be used to during different other process like patient registration and patient discharge.

**Level 1 Data Flow Diagram**

This Diagram shows the sub-processes of hospital management. These sub-processes comprise the important functions of the system which complete the whole project.

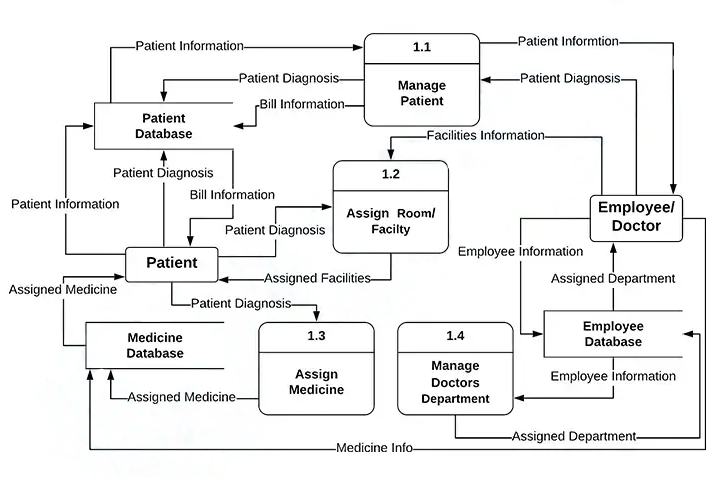
Hospital Management System Sub-processes:

* Patient Management
* Assigning Room or Facility
* Assigning Medicine
* Employee (Doctor and Nurses) Management

These processes specify the paths (flows) of data that may enter and exit the system.

### 4.3.3 Level 2 Data Flow Diagram

DFD level 2 is the highest abstraction among the data flow diagram levels. It focuses more on discussing the processes of the specific sub-process in the DFD level 1.

In short, this level provides a specific explanation of the specific sub-process.

**Level 2 Data Flow Diagram**

In this example, the databases of the hospital management system are as follows:

* Patient Database
* Medicines Database
* Employee Database

Each of these databases (data store) becomes the storage of data that comes in and the source of data that comes out.

# Implementation

## Algorithm

In this section, we discuss the algorithms implemented in the Hospital Management System (HMS). Each major module is covered, providing either pseudocode or a natural language explanation of the functionality.

### Patient Admission Algorithm

**Overview**

The Patient Admission Algorithm manages the process of admitting a patient to the hospital. It involves checking bed availability, assigning a unique identifier, and updating the hospital records.

**Pseudocode**

* Step 1: Check bed availability

if beds\_available():

* Step 2: Assign a unique identifier to the patient

patient\_id = generate\_patient\_id()

* Step 3: Update patient records and assign a bed

admission\_status, assigned\_bed = admit\_patient(patient\_id, patient\_info)

* Step 4: Output admission status and assigned bed

output (admission\_status, assigned\_bed)

else:

* Step 5: Output no available beds

output ("No available beds", None)

### Prescription Management Algorithm

**Overview**

The Prescription Management Algorithm handles the generation and management of prescriptions for patients. It ensures accurate medication details, dosage, and links prescriptions to patient records.

**Pseudocode**

* Step 1: Verify patient existence and validate medication details

If(validate\_patient(patientid)andvalidate\_medication\_details(medication\_info)

* Step 2: Generate a unique prescription identifier

prescription\_id = generate\_prescription\_id()

* Step 3: Record prescription details in the system

prescription\_status = record\_prescription(patient\_id, prescription\_id, medication\_info)

* Step 4: Output prescription status

output(prescription\_status)

else:

* Step 5: Output invalid patient or medication details

output ("Invalid patient or medication details", None)

### Billing System Algorithm

**Overview**

The Billing System Algorithm calculates and generates bills for patients, considering various factors such as treatment costs, room charges, and additional services. It also manages payment processing and updates financial records.

**Pseudocode**

* Step 1: Verify patient existence and validate treatment details

If(validate\_patient(patient\_id)and validate\_treatment\_details(treatment\_info):

* Step 2: Calculate treatment cost

treatment\_cost = calculate\_treatment\_cost(treatment\_info)

* Step 3: Calculate room charges

room\_charges = calculate\_room\_charges(room\_type)

* Step 4: Calculate additional service charges

additional\_service\_charges= calculate\_additional\_service\_charges(additional\_services)

* Step 5: Calculate total bill amount

total\_bill\_amount=treatment\_cost+room\_charges+ additional\_service\_charges

* Step 6: Process payment (if required)

payment\_status = process\_payment(patient\_id, total\_bill\_amount)

* Step 7: Update financial records

update\_financial\_records(patient\_id, total\_bill\_amount, payment\_status)

* Step 8: Output bill amount and payment status

output (total\_bill\_amount, payment\_status)

else:

* Step 9: Output invalid patient or treatment details

output ("Invalid patient or treatment details", None)

## 5.2 External APIs

The table provided in the pseudocode will include details about the external APIs used in the Hospital Management System (HMS) project. Each API will have a name, description, purpose of usage, and the corresponding function or class name where it is implemented.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name of API** | **Description of API** | **Purpose of usage** | **List down the function/class name in which it is used** |
| PatientInfoAPI | Retrieves patient information | Obtain patient details for admission | AdmissionModule.getPatientInfo |
| MedicalRecordAPI | Accesses medical records | Retrieve patient's medical history | MedicalRecordModule.fetchRecord |
| LabResultsAPI | Retrieves laboratory results | Access laboratory results for diagnostics | DiagnosticsModule.getLabResults |
| BillingSystemAPI | Manages billing and payment processing | Calculate and process patient bills | BillingModule.processBilling |
| InventoryManagementAPI | Controls hospital inventory | Update inventory for consumed medical supplies | InventoryModule.updateInventory |

## 5.3 User Interface

The Hospital Management System (HMS) boasts an intuitive and purpose-driven User Interface (UI) meticulously crafted for seamless navigation and enhanced user experience. Tailored for healthcare professionals, administrative staff, and patients, the UI prioritizes user-friendly design principles. From a comprehensive Dashboard to specialized forms, every component is designed for efficiency. The UI's graphical representations, intuitive forms, and interactive elements contribute to a seamless experience, empowering users to effortlessly manage hospital operations, electronic medical records, billing, and inventory. This section explores each UI component, revealing how design choices align with functional requirements, fostering a user-centric approach to healthcare administration.

### 5.3.1 Dashboard

* **Description:** The dashboard serves as the main entry point for users and provides an overview of key metrics and activities within the hospital.
* **Components:**
  + Quick access to patient information
  + Notifications for pending tasks
  + Graphical representation of occupancy rates

### 5.3.2 Patient Registration Form

* **Description:** This form allows hospital staff to input and save patient information into the system.
* **Components:**
  + Patient details (name, date of birth, contact information)
  + Emergency contact information
  + Insurance details

### 5.3.3 Appointment Scheduling Interface

* **Description:** Enables staff to schedule appointments for patients with doctors or for various medical procedures.
* **Components:**
  + Calendar for selecting appointment dates
  + Dropdowns for selecting doctors or medical services
  + Patient details and reason for appointment

### 5.3.4 Billing and Payment Interface

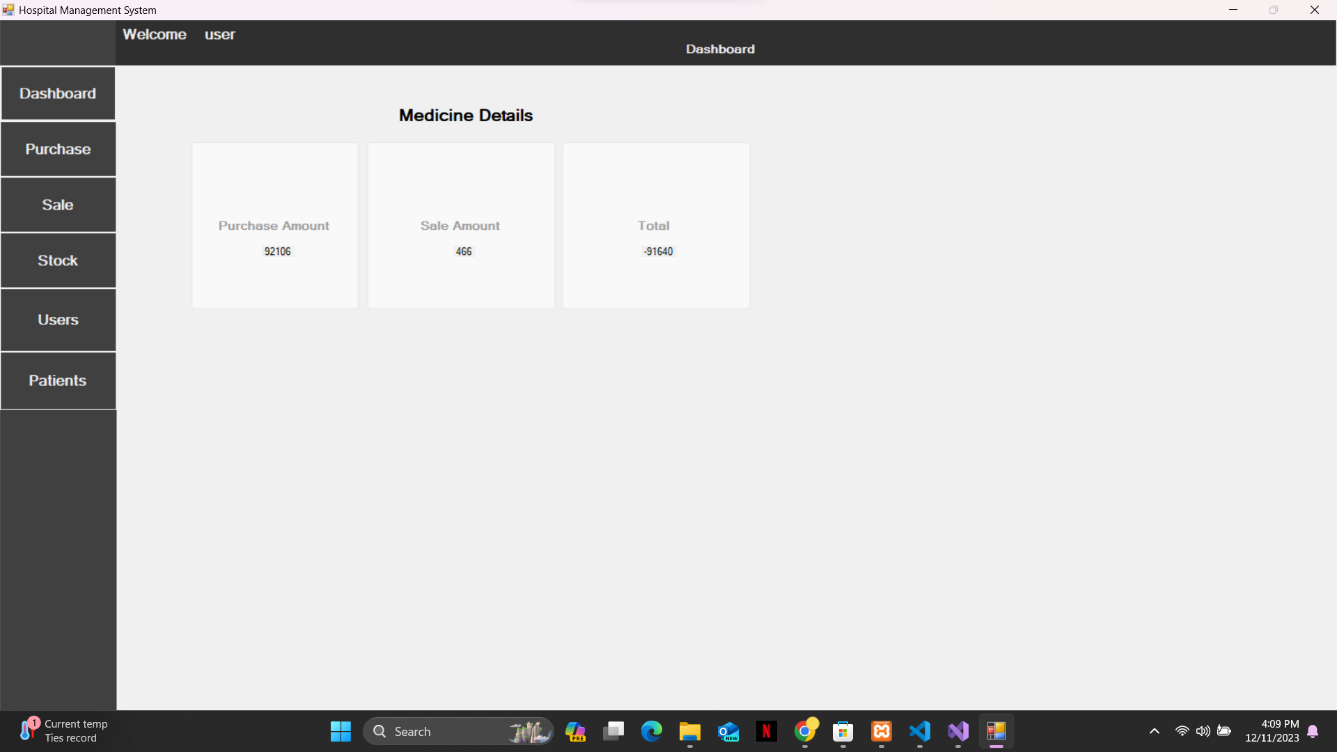
* **Description:** Allows staff to generate bills for patients and manage payment transactions.
* **Components:**
  + Itemized billing details
  + Payment status tracking
  + Options for payment methods

### 5.3.5 Inventory Management Interface

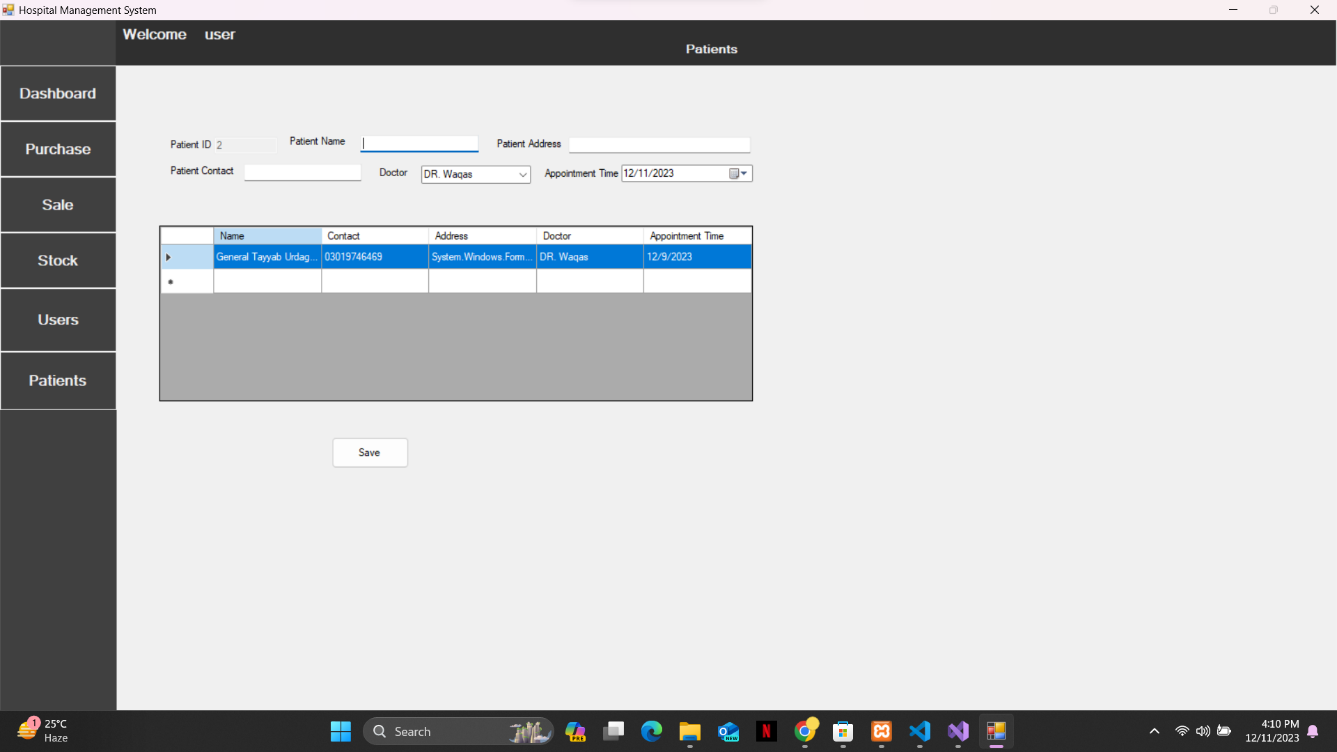
* **Description:** Provides a platform for managing hospital inventory, tracking supplies, and placing orders.
* **Components:**
  + Current inventory levels
  + Order history
  + Alerts for low stock levels

## 5.4 Screen Images

**Dashboard:**

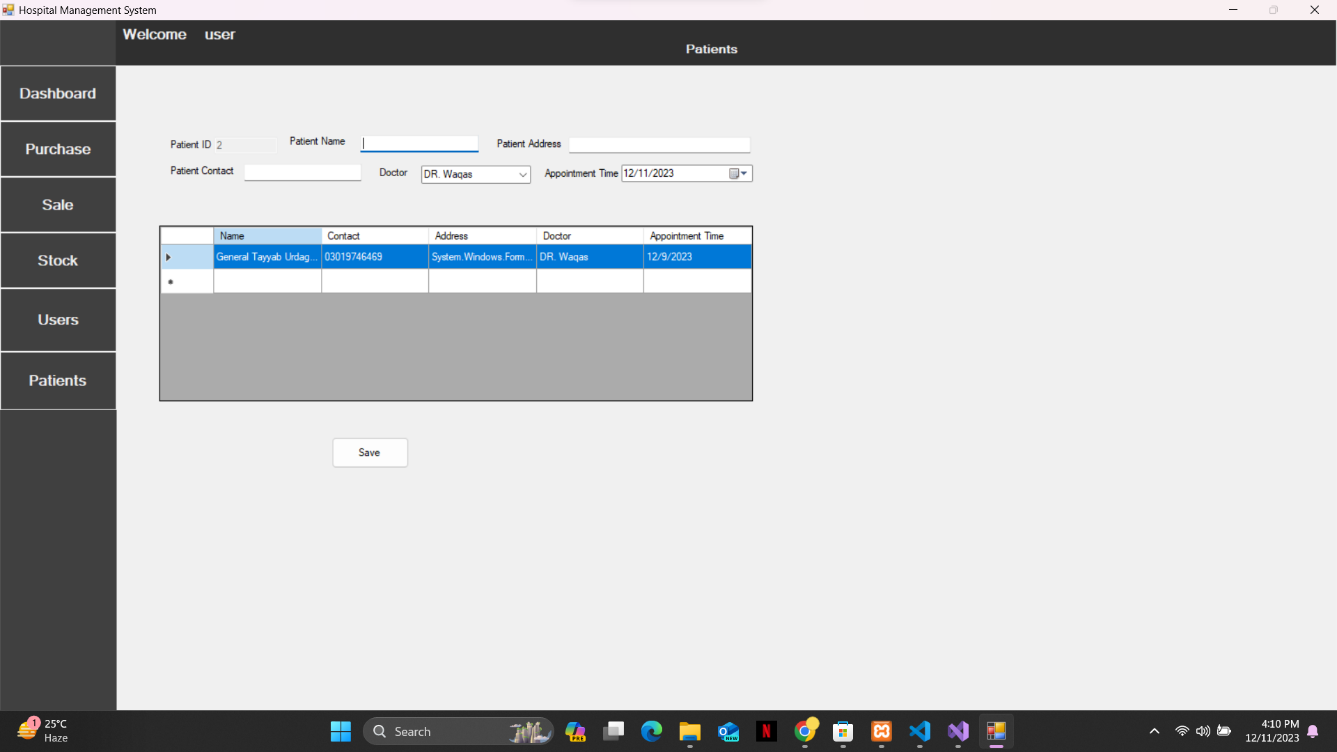


**Figure 1**

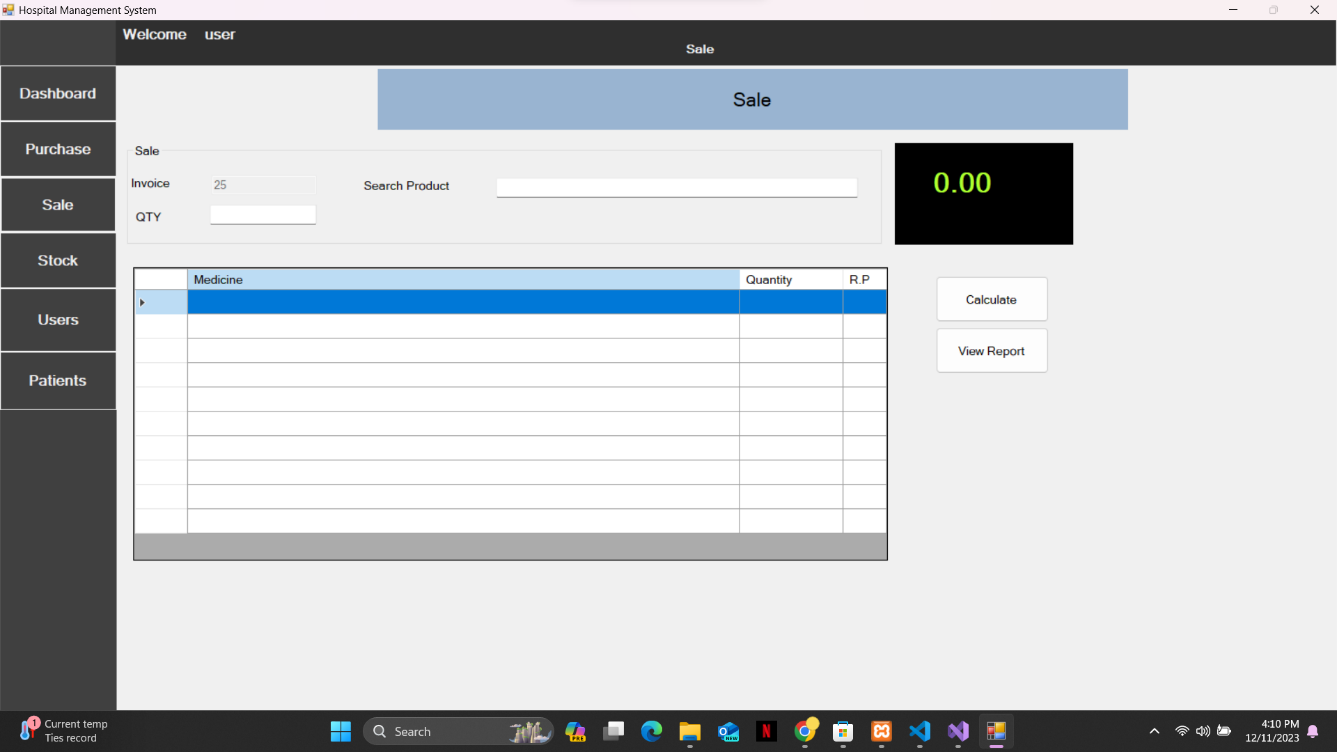
**Patient Registration Form:**

**Figure 2**

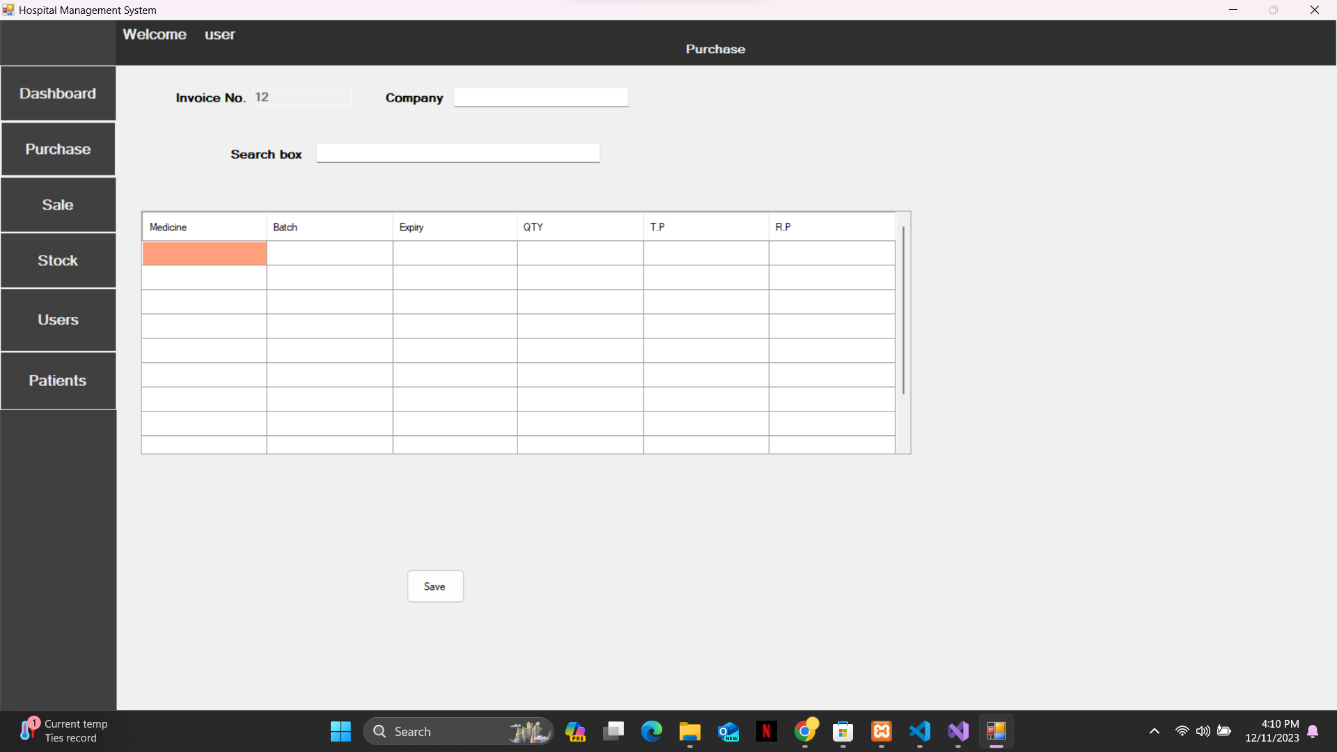
**Appointment Scheduling Interface:**



**Figure 3**

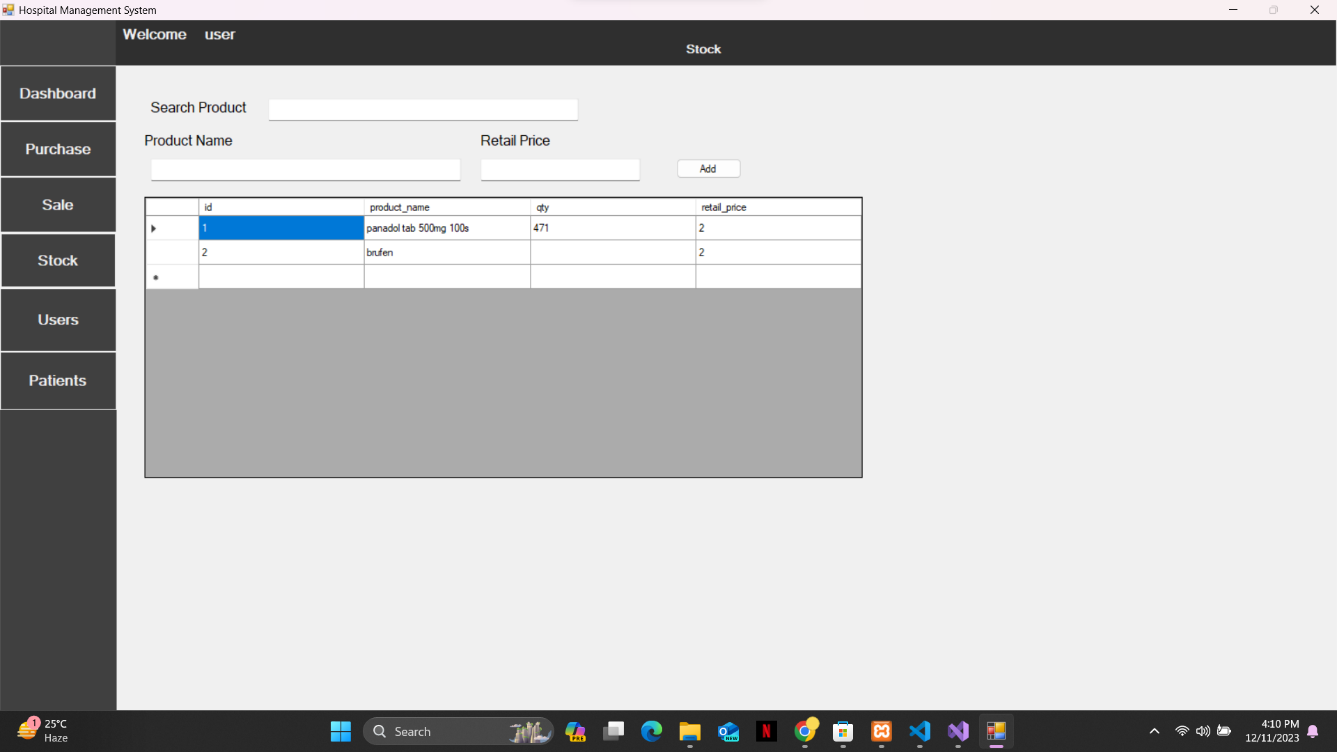
**Billing and Payment Interface:**

**Figure 4**



**Figure 5**

**Inventory Management Interface:**



**Figure 6**

# 6 Testing and Evaluation

## 6.1 Manual Testing

### 6.1.1 System Testing

After the successful development of the system, a comprehensive system testing phase is crucial to ensure its proper functioning. This step validates whether the system aligns with the predefined requirements. System testing encompasses various levels, including unit testing, functional testing, and integration testing. These tests aim to unveil any potential errors that might be concealed from the end user. It is imperative that all testing procedures are concluded satisfactorily before deploying the system for user interaction.

### Unit Testing

**Unit Testing 1**: Admin Login

**Testing Objective:** To ensure the admin login functionality is working correctly.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Verify admin login after clicking the 'Login' button with correct input data | Username: admin, Password: [admin\_password] | Successfully log into the admin module. | [Pass/Fail] |
| 2. | Attempt admin login with incorrect credentials | Username: admin, Password: [incorrect\_password] | Login should fail, and an error message should be displayed. | [Pass/Fail] |

**Unit Testing 2:** Staff Login

**Testing Objective:** To ensure the staff login functionality is working properly.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Verify staff login after clicking the 'Login' button with correct input data | Username: staff001, Password: [staff\_password] | Successfully log into the staff account. | [Pass/Fail] |
| 2. | Attempt staff login with incorrect credentials | Username: staff001, Password: [incorrect\_password] | Login should fail, and an error message should be displayed. | [Pass/Fail] |

**Unit Testing 3:** Patient Registration

**Testing Objective:** To ensure the patient registration process works correctly.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Register a new patient with valid information | Patient details: [valid\_patient\_info] | Patient should be registered successfully. | [Pass/Fail] |
| 2. | Attempt to register a patient with incomplete information | Patient details: [incomplete\_patient\_info] | Registration should fail, and an error message should be displayed. | [Pass/Fail] |

**Unit Testing 4:** Doctor Login

**Testing Objective:** To ensure the doctor login functionality works properly.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Doctor logs in with correct credentials | Username: [valid\_username] Password: [valid\_password] | Doctor should log in successfully. | [Pass/Fail] |
| 2. | Doctor attempts to log in with incorrect credentials | Username: [invalid\_username] Password: [invalid\_password] | Login should fail, and an error message should be displayed. | [Pass/Fail] |

### 6.1.3 Functional Testing

**Functional Testing 1:** Login with Different Roles

**Objective:** To ensure that the correct page with the correct navigation bar is loaded.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Login as an "Administrator" | Username: admin123 Password: admin@pass | Dashboard for the Administrator is loaded with the admin navigation bar. | [Pass/Fail] |
| 2. | Login as a "Doctor" | Username: doc456 Password: doc@pass | Dashboard for the Doctor is loaded with the doctor navigation bar. | [Pass/Fail] |
| 3. | Login as a "Nurse" | Username: nurse789 Password: nurse@pass | Dashboard for the Nurse is loaded with the nurse navigation bar. | [Pass/Fail] |
| 4. | Login as a "Pharmacist" | Username: pharm101 Password: pharm@pass | Dashboard for the Pharmacist is loaded with the pharmacist navigation bar. | [Pass/Fail] |
| 5. | Attempt to login with incorrect credentials | Username: Invalid Password: Invalid | System should not allow login, and an error message should be displayed. | [Pass/Fail] |

**Functional Testing 2:** Patient Registration

**Objective:** To ensure that patient registration functionality works correctly.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Register a new patient | Provide valid patient details in the registration form | Patient should be successfully registered, and the system should display a success message. | [Pass/Fail] |
| 2. | Attempt to register a patient with incomplete details | Provide incomplete information in the registration form | System should not allow registration and should display appropriate error messages. | [Pass/Fail] |
| 3. | Attempt to register a patient with an existing ID | Provide details with an ID that already exists in the system | System should not allow registration and should display an error indicating that the ID is already in use. | [Pass/Fail] |

**Functional Testing 3:** Appointment Scheduling

**Objective:** To ensure that appointment scheduling functions correctly.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Schedule an appointment for a patient | Select a patient, choose a date and time for the appointment | Appointment should be successfully scheduled, and the system should confirm the appointment details. | [Pass/Fail] |
| 2. | Attempt to schedule an appointment without selecting a patient | Choose a date and time without selecting a patient | System should not allow scheduling and should prompt to select a patient. | [Pass/Fail] |

### 6.1.4 Integration Testing

**Integration Testing 1:** Admin Module Integration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Login as an administrator | Username: Admin, Password: [Admin's password] | Successful login, and the Admin Dashboard with navigation bar is loaded. | [Pass/Fail] |
| 2. | Add a new staff member | Provide valid staff details and click "Add" | Staff member is successfully added, and the system returns to the Admin Dashboard. | [Pass/Fail] |
| 3. | View sales report | Navigate to the sales report section | Sales report is displayed with relevant data. | [Pass/Fail] |

**Integration Testing 2:** Pharmacy Module Integration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Login as a staff member | Provide valid staff credentials | Successful login, and the Staff Dashboard is loaded. | [Pass/Fail] |
| 2. | Add a new drug to the inventory | Provide valid drug details and click "Add" | Drug is successfully added, and the inventory is updated. | [Pass/Fail] |
| 3. | Perform a drug sale transaction | Select a patient, choose a drug, and complete the sale | Sale transaction is successful, and the drug stock is updated. | [Pass/Fail] |

**Integration Testing 3:** Patient Module Integration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Register a new patient | Provide valid patient details | Patient is successfully registered, and the system returns to the Patient Module. | [Pass/Fail] |
| 2. | Schedule an appointment for the patient | Choose a date and time for the appointment | Appointment is successfully scheduled, and the system confirms the details. | [Pass/Fail] |

# 7 Conclusion and Future Work

The completion of the Hospital Management System (HMS) project marks a significant achievement in our commitment to revolutionize healthcare administration. The system, designed to streamline hospital operations, has successfully addressed critical challenges faced by healthcare institutions.

In the course of this project, we have implemented robust solutions for user authentication, inventory management, and communication within the hospital ecosystem. The user-friendly interface ensures a seamless experience for administrators, staff, doctors, and patients, fostering efficiency and accuracy in daily operations.

## 7.1 Conclusion

The completion of the Hospital Management System (HMS) project marks a significant milestone in addressing the challenges faced by healthcare institutions. Through this FYP, we have successfully implemented a comprehensive system that streamlines various processes within a hospital environment. Key achievements include:

* **User Authentication and Authorization:** Implemented secure login mechanisms for administrators, staff, doctors, and patients, ensuring data privacy and access control.
* **Inventory Management:** Developed functionalities for staff to manage drug inventory efficiently, facilitating seamless procurement and distribution processes.
* **User-Friendly Interface:** Designed an intuitive user interface for enhanced user experience, allowing users to navigate through the system with ease.
* **Communication Features:** Integrated a chatbot for real-time communication between vendors, customers, and staff, fostering efficient information exchange.
* **Multi-language Support:** Provided language customization options for users, accommodating diverse linguistic preferences.

## 7.2 Future Work

While the current HMS system addresses critical aspects of hospital management, there is ongoing potential for improvement and expansion. Future work may include:

* **Enhanced Data Analytics:** Implement advanced analytics to derive insights from patient records, inventory trends, and operational data, aiding in decision-making processes.
* **Telemedicine Integration:** Explore the integration of telemedicine features, enabling remote consultations and enhancing accessibility to healthcare services.
* **Mobile Application Development:** Develop a mobile application to extend the reach of the HMS, allowing users to access features on the go.
* **Integration with Wearable Devices:** Explore possibilities of integrating the HMS with wearable health devices for real-time health monitoring and data synchronization.
* **Machine Learning for Predictive Analysis:** Implement machine learning algorithms to predict patient admission rates, optimize resource allocation, and improve overall operational efficiency.

By pursuing these avenues of future work, the HMS can continue to evolve, ensuring it remains at the forefront of technology and continues to meet the dynamic needs of healthcare institutions.

# 8. References

1. Algabry, O., 2016. Requirements Engineering — Requirements Validation (Part 4).

[Online]. Available at: <https://medium.com/omarelgabrys-blog/requirements-engineering-requirements-validation-part-6-29778d7bde24> . [Accessed: 4 December

2019].

1. ASQ, 2019. What is the iso 9000 standards series? [Online]. Available at:

<https://asq.org/quality-resources/iso-9000> . [Accessed: 11 December 2019].

1. Baghdasaryan, A., 2018. Why is software quality assurance important? [Online].

Available at: <https://medium.com/fintegro-company-inc/why-is-software-quality-assurance-important-bd5f0f2bcd9> . [Accessed: 11 December 2019].

1. Barkova, D., 2017a. HIPAA compliance testing for web applications. [Online].

Available at: <https://www.scnsoft.com/healthcare/hipaa-compliance/software-testing> . [Accessed: 4 December 2019].

1. Barkova, D., 2017b. 4 Testing Types To Focus On in Healthcare Projects. [Online].

Available at: <https://www.scnsoft.com/software-testing/healthcare> . [Accessed: 4 December 2019].

1. CareerRide, 2016. Explain different types of Verification - Testing. [Online]. Available

at: <https://www.careerride.com/testing-types-of-verification.aspx> . [Accessed: 6

December 2019].

1. Catherine, W., 2018. Project Verification Methods & Techniques. [Online]. Available

at: <https://study.com/academy/lesson/project-verification-methods-techniques.html> . [Accessed: 6 December 2019].

1. Corporate, A., 2018. Top 5 Requirements Elicitation Techniques. [Online]. Available

at: <https://anarsolutions.com/top-5-requirements-elicitation-techniques/>

[Accessed: 4 Decemeber 2019].