Sri Lanka Institute of Information Technology

Innobot Health Care Management System

Declaration

We declare that the project report was not copied from any content done by any other organization, university, institute, or a previous student project group in SLIIT and was not copied from the internet or any other resources.

Project Details

Project Title	Health Care Management System “Innobot Health”
Project ID	**ITP_2024_Y2_S2_WE10**

Group Members

	Name with Initials	**Registration Number**	**Contact Number**	**Email**
1.	S.A.N. Bamunusinghe	IT22515612	0772187484	it22515612@my.sliit.lk
2.	DME Wimalagunasekara	IT22917270	0768952222	it22917270@my.sliit.lk
3.	S A T Nayanapriya	IT22892058	0725203742	it22892058@my.sliit.lk
4.	Obeyesekere A D	IT22332080	0774801500	it22332080@my.sliit.lk
5.	Chamikara M G S	IT22905840	0712905241	it22905840@my.sliit.lk
6.	M F M Farsith	IT22354556	0770494812	it22354556@my.sliit.lk
7.	D S I Gamage	IT22907516	0779705099	it22907516@my.sliit.lk
8.	W A Malsha Haren	IT22307576	0788760703	it22307576@my.sliit.lk

Abstract

In this era, all the functionalities and most of the services are used to be automated and digitized. In the healthcare sector most hospitals are automated based on accuracy and efficiency, this report investigates Innobot Health, a healthcare management system leveraging intelligent automation. By analyzing its functionalities, we assess its effectiveness in streamlining data processing, optimizing revenue cycle management (RCM), and generating data-driven insights. This evaluation explores the potential benefits for both healthcare providers, who can focus on patient care and optimize resource allocation, and patients, who can experience improved service delivery and potentially reduced costs. We conclude by examining the impact of Innobot Health on the healthcare landscape and identifying potential areas for further development to enhance its functionalities and broaden its applications.

Acknowledgment

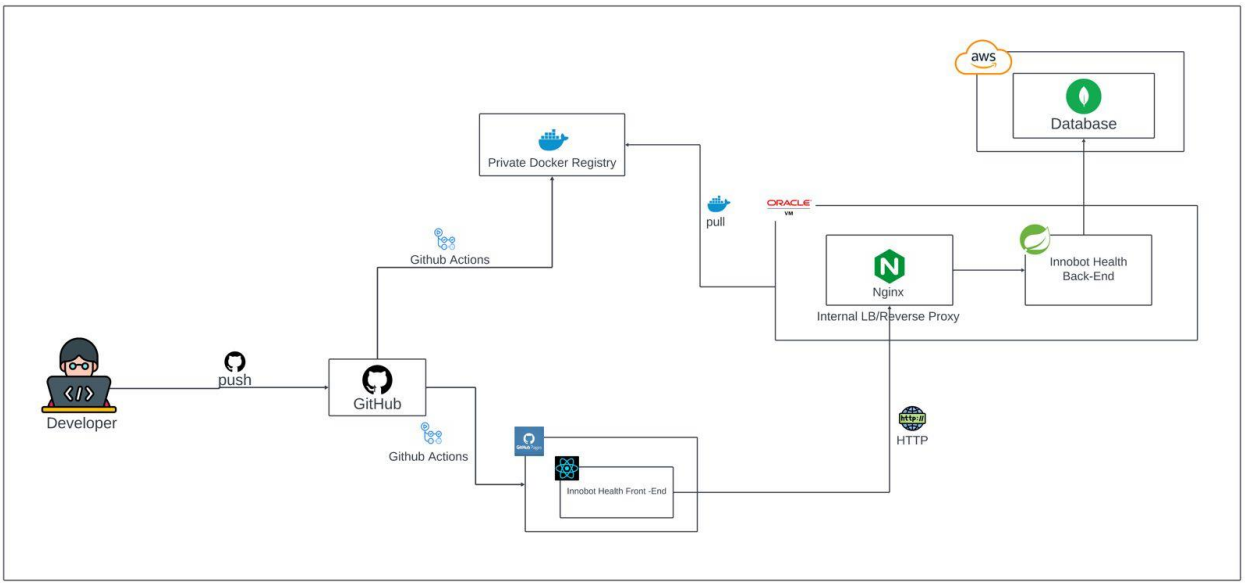
We would like to express our sincere gratitude to all those who helped us through this project. Special gratitude goes to the Sri Lanka Institute of Information Technology for their guidance in starting the project, constant supervision, and support in completing the project successfully. The success of the work described in this document was done as our second-year project for the subject Information Technology Project. This project is the result of all the dedicated work of the group members and the encouragement, support, and guidance given by many others. Therefore, we would like to express our appreciation to all who gave us the support to complete this significant task. Also, we wish to thank all colleagues and friends for all their help, support, and valuable advice.

Introduction

The ever-growing demands on healthcare systems necessitate innovative solutions to improve efficiency and patient care. Innobot Health emerges as a promising answer, leveraging automation to transform healthcare management. This report delves into the functionalities of Innobot Health, focusing on its automation capabilities. We will explore how Innobot automates various tasks within the healthcare management system, analyzing its impact on administrative processes, data handling, and overall workflow optimization. Ultimately, this report aims to assess the effectiveness of Innobot Health in streamlining healthcare operations and its potential to enhance the quality of care delivered.

This report describes, depicts the

High Level System Design Diagram



User stories or functionalities completed-

Inventory Management System

As a pharmacy manager, I need to add new medicines to the inventory system along with relevant details such as name, description, quantity, price, supplier data, and expiry date to ensure accurate and up-to-date inventory management.

As a pharmacy manager, I wish to make changes to the current medicine inventory, including updating quantities, prices, supplier information, and expiration dates.

As a pharmacist, I want to easily view the list of available medicines and search for specific medicines based on various criteria.

As a pharmacy manager, I want to keep track of medicine expiration dates to ensure expired medicines are removed from inventory in a timely manner, so that we maintain the quality and safety of our stock.

As a pharmacy manager, I want to oversee the receipt and inspection of incoming medicine shipments, including verifying quantities, inspecting quality, and updating inventory records, so that we can accurately track new stock arrivals and maintain inventory accuracy.

As a pharmacy manager, I want to maintain relationships with medicine suppliers, negotiate contracts, and manage procurement processes to ensure timely and cost-effective replenishment of medicine inventory, so that we can secure reliable sources of supply and minimize procurement costs.

As a pharmacy manager, I want to create various reports to monitor medicine inventory performance, examine patterns, and make informed decisions, so that we can keep track of stock levels, identify areas for improvement, and streamline inventory control procedures.

Staff Management Subsystem

As a practice administrator, I want to enroll my practice in the system, so that I can manage my staff efficiently.

Staff management Subsystem – user stories

- As a practice administrator, I want to enroll my practice in the system, so that I can manage my staff efficiently."
- "As a practice administrator, I want to fill out a registration form with practice details, so that I can initiate the enrollment process."
- "As a practice administrator, I want to receive confirmation upon successful enrollment, so that I can proceed with staff management setup."

- “As a practice administrator, I want to receive guidance on how to integrate the staff management system with my practice, so that I can ensure smooth operation.’

- “ As a practice manager, I want to allocate coordinators and staff during the initial setup phase, so that my practice can operate smoothly from the beginning.”
- “As a practice manager, I want to designate roles and responsibilities to coordinators, so that they can efficiently manage staff.”
- “As a practice manager, I want to assign staff to specific roles, so that they can begin their duties promptly.”
- “As a practice manager, I want to receive notifications upon successful allocation, so that I can track progress effectively.”

- "As a practice administrator, I want to grant specific access and privileges to staff members, so that they can perform their duties effectively.”
- “As a practice administrator, I want to define access levels based on staff roles, so that each staff member has appropriate system access.”
- “As a practice administrator, I want to set permissions for various system functionalities, so that staff members can only access relevant features.”
- “As a practice administrator, I want to receive alerts for any unauthorized access attempts, so that I can maintain system security.”

- “As a staff member, I want the ability to control the visibility of my profile, so that I can manage who can view my information.”
- “As a staff member, I want to set my profile visibility settings to public or private, so that I can control who can see my profile information.”
- “As a staff member, I want to receive confirmation when my visibility settings are updated, so that I can ensure my privacy preferences are applied.”
- “As a staff member, I want to be able to adjust my visibility settings at any time, so that I can adapt to changing preferences or circumstances.”

- " As a practice administrator, I want to send messages to all logged users by selecting users, so that I can communicate important information efficiently.”
- “As a practice administrator, I want to compose messages with customizable content, so that I can convey specific information to staff members.”
- “As a practice administrator, I want to select target recipients for messages, so that I can ensure relevant staff members receive the communication.”
- “As a practice administrator, I want to track message delivery status, so that I can confirm successful transmission and follow up if necessary.’

- “As a practice manager, I want to view staff activity logs, so that I can monitor staff performance and track changes.”
- “As a practice manager, I want to access a log of all staff activities within the system, so that I can review staff interactions and behaviors.”
- “As a practice manager, I want to filter activity logs based on criteria such as time period or user, so that I can focus on specific aspects of staff activity.”
- “As a practice manager, I want to export activity logs for further analysis or documentation purposes, so that I can maintain records and comply with regulations.”

Medicine Inventory Management System – user stories

- "As a pharmacy manager, I need to add new medicines to the inventory system along with relevant details such as name, description, quantity, price, supplier data, and expiry date to ensure accurate and up-to-date inventory management."

- As a pharmacy manager, I wish to make changes to the current medicine inventory, including updating quantities, prices, supplier information, and expiration dates.

- "As a pharmacy manager, I want to keep track of medicine expiration dates to ensure expired medicines are removed from inventory in a timely manner, so that we maintain the quality and safety of our stock."

- "As a pharmacy manager, I want to oversee the receipt and inspection of incoming medicine shipments, including verifying quantities, inspecting quality, and updating inventory records, so that we can accurately track new stock arrivals and maintain inventory accuracy."
- "As a pharmacy manager, I want to maintain relationships with medicine suppliers, negotiate contracts, and manage procurement processes to ensure timely and cost-effective replenishment of medicine inventory, so that we can secure reliable sources of supply and minimize procurement costs."

- "As a pharmacy manager, I want to create various reports to monitor medicine inventory performance, examine patterns, and make informed decisions, so that we can keep track of stock levels, identify areas for improvement, and streamline inventory control procedures."

Notifications Management system – user stories.

- "As a user, I should be able to send notifications to the desired stakeholders. So that I can notify them regarding their engagements."

- "As a user, I should be able to receive notifications from the respective parties. So that I can get notified about my upcoming responsibilities based on my notification preference."

- " As a user, I should be able to acknowledge notifications sent by the respected parties. So that I can notify the senders that I acknowledged the message."
- " As a user, I should be able to edit the notifications I sent to the respective stakeholders. So that I can rectify the message and resend the notification."

- " As a user, I should be able to reply to the notifications I received. So that I can notify the sender about my feedback regarding the message I received."

Appointment Scheduling System – user stories

- As a patient, I want to fill out a user-friendly appointment request form to schedule an appointment with a healthcare provider.
- As a Patient, I want to be able to add all my necessary information in scheduling my appointment.
- As a patient, I want to be able to cancel my scheduled appointments
- As a Patient, I want to be able to immediately delete my appointment onclick of cancel my appointment.
- As a patient, I want to view the availability of healthcare providers to select a suitable appointment slot.
- As a patient, I want to have access to reliable and up-to-date details of healthcare providers.
- As a Patient, I want to be able to confirm my Appointment.
- As a patient, I want to select a convenient appointment slot based on provider availability
- As a Patient, I want to be able to select a Health Sector (i.e.: Dentistry , Cardiologists).
- .As a Health Desk Supporter ,I want to be able to select a Health Sector(i.e.: Dentistry , Cardiologists).

- As a patient, I want the overview to display detailed information about each scheduled appointment, including the provider, appointment type, date, and time
- As a patient, I want the ability to manage my scheduled appointments directly from the overview page, including rescheduling or canceling appointments.
- As a patient, I want to be able to update my scheduled appointments, allowing me to make changes if necessary.
- As a healthcare desk support staff, I want to be able to modify existing appointments on behalf of patients, ensuring accurate scheduling.
- As a patient, I want to book appointments seamlessly based on provider availability and my preferences.
- As a patient, I want to be able to search for a doctor in the Overview of Doctor UI.
- As a patient, I want to get a report of the scheduled appointment.

Procedure and Diagnosis System | System Administration

User Stories-

- As a system administrator, I want to be able to grant privileges to privileged users to add, update, and delete procedure codes.
- As a privileged user, I want to add a new procedure code with its description.
- As a privileged user, I want to add a new diagnosis code with its description.
- As a privileged user, I want to update an existing procedure code.
- As a privileged user, I want to update an existing diagnosis code.

Claim Management System

User stories -

- As a user, I want to submit a claim for reimbursement.
- As a staff member, I want to check and verify the details for submission.
- As an approver staff, I want to review and Approve submitted claims.
- As a staff, I want to track the status of my submitted claims
- As an Administrator, I want to generate reports on claim statistics.
- As an administrator, I want to view analytics on overall claim trends.

Insurance Management

User stories -

- As an authorized staff member, I want to create new insurance records so I can manage patient insurance details.
- As a user, I want to delete or archive insurance records to maintain data accuracy.
- As an administrator, I want to define and manage fee schedules for different CPT codes.
- As a user with appropriate permissions, I want to update fee schedules to reflect changes in allowed amounts.
- As a user, I want the system to generate insurance records reports.
- As a user, I want to access reports and analytics on insurance records to identify trends and patterns.

Patient management system
User stories -

- As a staff user, I want to register patients, So I can maintain their profiles easily.
- As a staff user, I want to update patient records when needed
- As a staff, I want to the patient's billing status So, I could be able to record the claim balance of the patient
- As a staff member, I want to delete the duplicated patient records, So I could be able to maintain patient records accurately.
- As a staff user, I want to generate the reports So, I could be able to analyze how many patients got treatment during a period.

Repositories

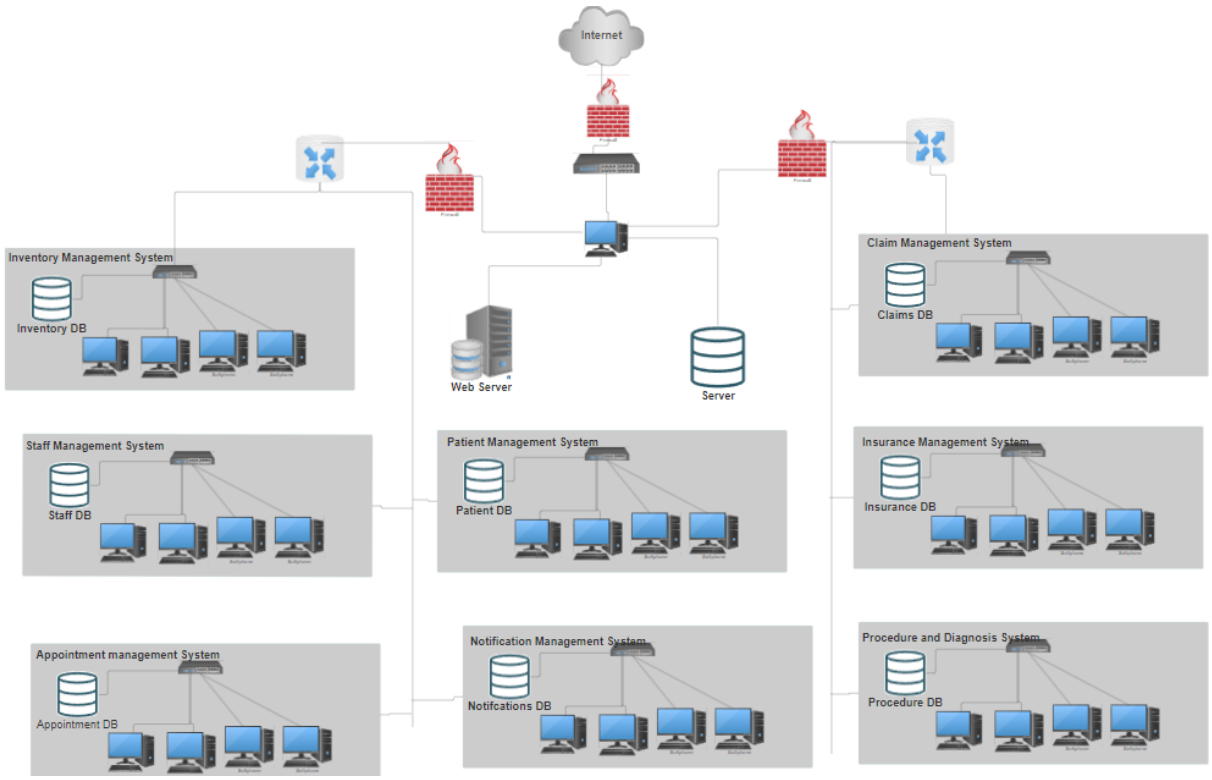
Frontend

https://github.com/InnobotHealth/InnobotHealthFE

Backend

https://github.com/DulangaMW/Innobothealth-Access-Management-System.git

Network Diagram



Innovative Parts of the Project:

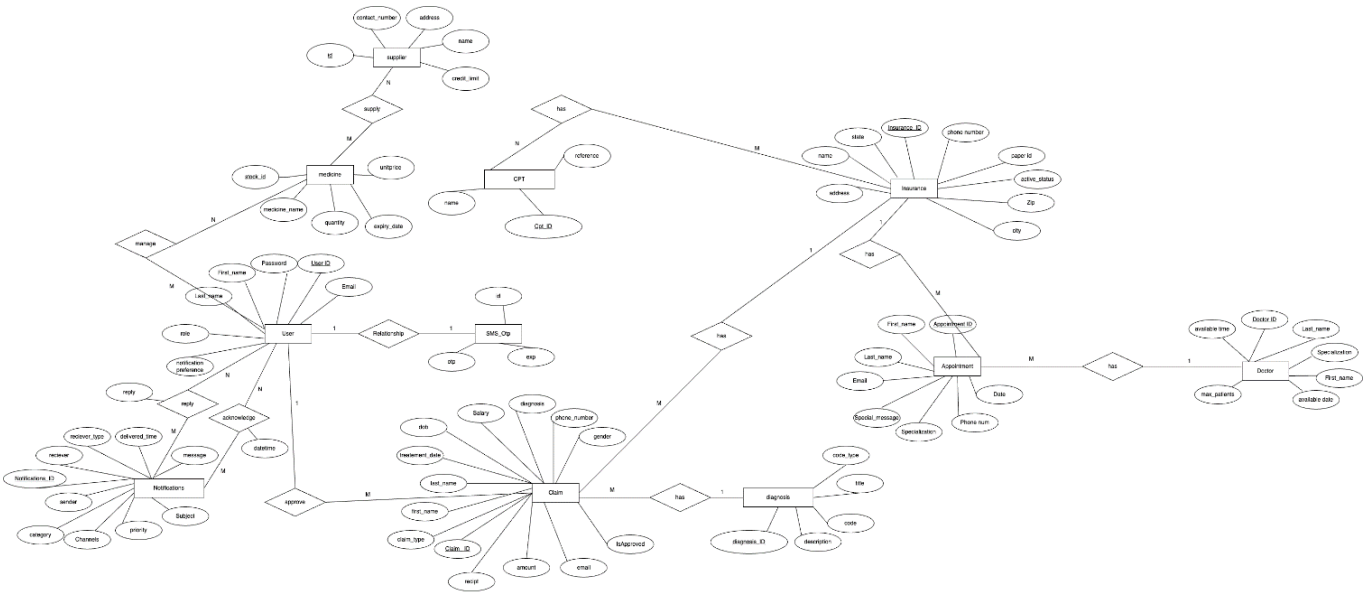
1. **Blockchain-Powered Data Security**: Implement a blockchain-based data security system to ensure the integrity and confidentiality of patient records and sensitive information. Blockchain technology provides a decentralized and immutable ledger, enhancing security and privacy in healthcare data management.
2. **AI-Driven Predictive Analytics**: Integrate artificial intelligence algorithms for predictive analytics to forecast patient demand, optimize resource allocation, and improve operational efficiency within healthcare facilities. By analyzing historical data and trends, the system can assist healthcare providers in making informed decisions and mitigating risks.
3. **IoMT Integration for Remote Monitoring**: Incorporate Internet of Medical Things (IoMT) devices into the system to enable remote monitoring of patients' health conditions and vital signs. IoMT technology allows for real-time data collection and analysis, facilitating proactive healthcare management and early intervention.
4. **Virtual Reality (VR) for Patient Education**: Utilize virtual reality technology to create immersive educational experiences for patients, offering simulations of medical procedures, treatment options, and health information. VR-based patient education can enhance understanding, engagement, and adherence to treatment plans.
5. **Gamification for Staff Training**: Implement gamification elements within the system to enhance staff training and engagement. By incorporating game-like mechanics such as rewards, challenges, and progress tracking, healthcare providers can promote continuous learning and improve staff performance.

Commercialization Topics:

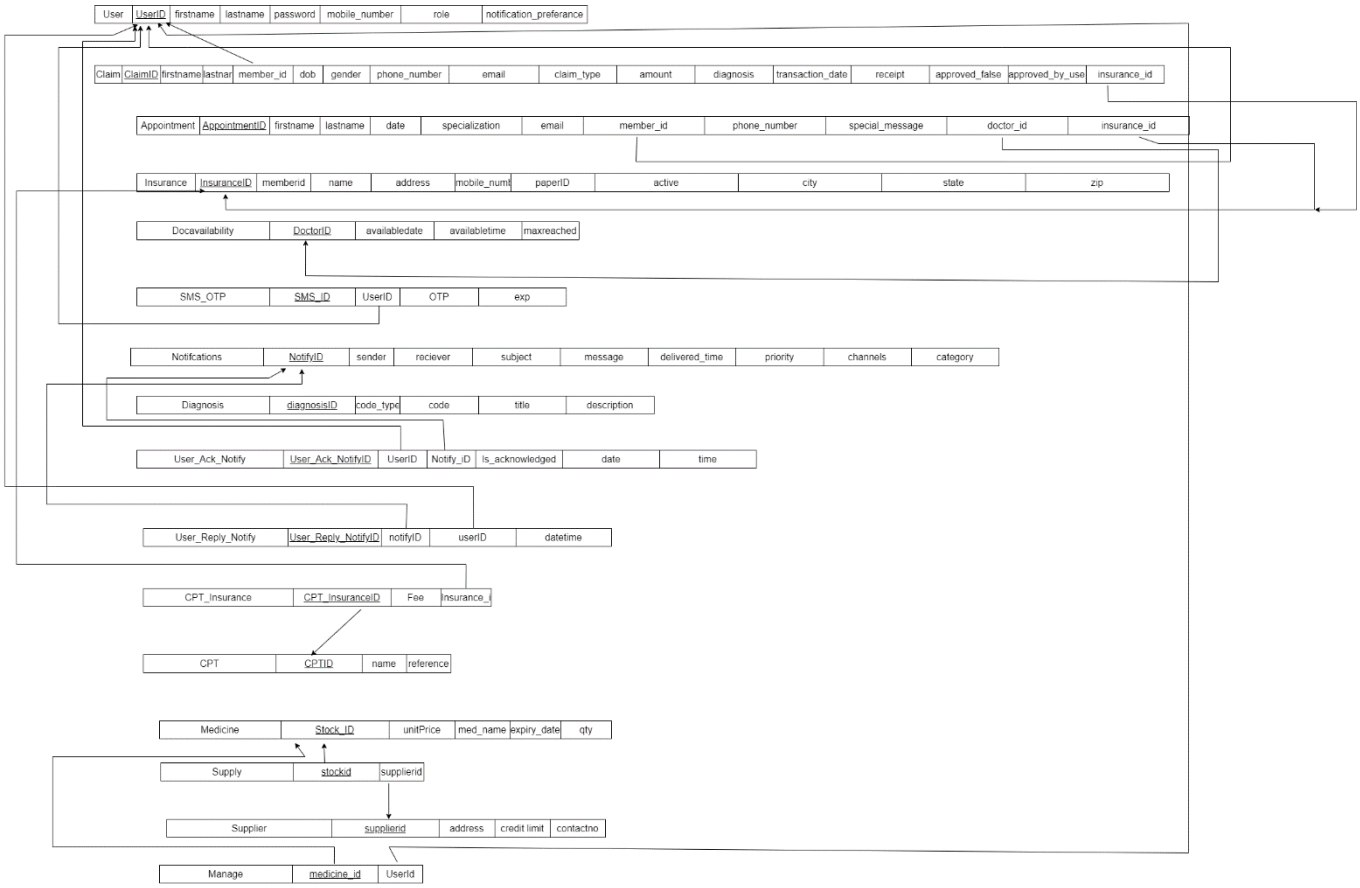
1. **Subscription-Based Pricing Model**: Develop a subscription-based pricing model for the practice management system, offering healthcare providers a recurring payment structure for access to the software and ongoing support services.
2. **Strategic Partnerships with Telehealth Providers**: Form strategic partnerships with telehealth companies to integrate telemedicine capabilities into the practice management system, expanding the range of services offered and addressing the growing demand for virtual care solutions.
3. **Customization and White-Labeling Options**: Offer customization and white-labeling options for the practice management system, allowing healthcare providers to tailor the software to their specific needs and branding requirements.
4. **Data Monetization Strategies**: Explore opportunities for data monetization by anonymizing and aggregating patient data within the system and offering insights and analytics to pharmaceutical companies, research institutions, and other healthcare stakeholders.

5. **Market Analysis and Go-to-Market Strategy**: Conduct market research to identify target customer segments, competitors, and market trends, and develop a comprehensive go-to-market strategy for launching and promoting the practice management system.

ER diagram



Normalized schema



Test case design

Inventory Management System – S.A.N. Bamunusinghe IT22515612

Test case 1: Adding New Medicine

Project ID: TP_2024_Y2_S2_WE10					
Project Name: Medicine Inventory System					
Testing Function: Add New Medicine					
Test Case ID: TC_ADD_001	**Project Name: Test Adding Valid Medicine**				
Test Priority: High					
Test Description: This test case verifies that a new medicine can be added to the inventory with all required fields correctly filled out.					
Test Steps: Step 1: Navigate to the 'Add New Medicine' section. Step 2: Enter all required patient details into the form. Step 3: Click the "Save" button to submit the information. Step 4: Check for a confirmation message with the assigned patient ID.					
Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
ADD_01	Name: Paracetamol Description: Pain reliever and a fever reducer Quantity: 100 Price: $5.00 Supplier Information: ABC Pharmaceuticals	A confirmation message stating "Medicine added successfully with ID: [UniqueID]"	A confirmation message stating "Medicine added successfully with ID: MED987654321"	PASS	Medicine was added successfully, and a unique ID was generated

	Expiry Date: 2025-12-31				

Inventory Management System - S.A.N. Bamunusinghe IT22515612

Test case 2: Updating Medicine Inventory

Project ID: TP_2024_Y2_S2_WE10					
Project Name: Medicine Inventory System					
Testing Function: Update Medicine Inventory					
Test Case ID: TC_UPDATE_001			**Project Name: Test Updating Medicine Quantity**		
Test Priority: Medium					
Test Description: This test case checks if the medicine quantity can be updated in the inventory.					
Test Steps: Step 1: Navigate to the 'Update Medicine Inventory' section. Step 2: Search for the medicine by name 'Paracetamol'. Step 3: Select the 'Quantity' field and enter the new quantity. Step 4: Click the 'Update' button and verify the confirmation message.					
Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
UPD_01	Medicine Name: Paracetamol New Quantity: 150	A confirmation message stating, "Medicine updated successfully."	A confirmation message stating, "Medicine updated successfully."	PASS	Quantity updated correctly in the database.

Completed	Work in progress	To be completed
View medicines records	Generate system inventory reports	Generate system inventory reports
Create and update records		
Track expired medicine and remove them		

SQL queries used

- None SQL base.

Algorithms and pseudo-codes

1. Adding New Medicine:

Algorithm:

- Get Medicine Information: Collect details like name, description, quantity, price, supplier information, expiry date, etc.
- Validate Information: Check for missing fields or invalid data formats (e.g., negative quantity, invalid date format for expiry).
- Generate Unique ID (Optional): If the system doesn't assign IDs, generate a unique identifier for the medicine.
- Store Medicine Data: Save the information in the medicine inventory database.
- Confirmation: Provide a confirmation message with the assigned ID (if applicable).

Pseudocode:

```
FUNCTION AddMedicine()
  DECLARE name, description, quantity, price, supplier, expiryDate, etc.
  INPUT name, description, quantity, price, supplier, expiryDate, etc.
  IF any field is empty THEN
    DISPLAY "Error: Missing information!"
  ELSEIF quantity < 0 OR invalid date format in expiryDate THEN
    DISPLAY "Error: Invalid data!"
  ELSE
    IF unique ID generation required THEN
      Generate unique ID
    ENDIF
    STORE medicine data in database
    DISPLAY "Medicine added successfully! ID: (uniqueID)" (if applicable)
  ENDIF
END FUNCTION
```

2. Updating Medicine Inventory:

Algorithm:

- Search for Medicine: Allow searching by ID or name to locate the medicine to be updated.
- Select Fields to Update: Allow selection of specific fields to modify (e.g., quantity, price, supplier information, expiry date).
- Get Updated Information: Collect the new values for the chosen fields.
- Update Database: Modify the corresponding entries in the medicine inventory database.
- Confirmation: Display a confirmation message indicating successful update.

Pseudocode:

```
FUNCTION UpdateMedicine(medicineID)
  DECLARE updatedFields, newData
  SearchMedicine(medicineID)  // Call search function
  IF medicine not found THEN
    DISPLAY "Medicine not found."
  ELSE
    SELECT updatedFields from user
    INPUT newData for updatedFields
    UPDATE database with newData for chosen fields in medicine record
    DISPLAY "Medicine information updated successfully."
  ENDIF
END FUNCTION
```

3. Viewing and Searching Medicine List:

Algorithm:

- Display Options: Provide options to view the entire medicine list or search based on criteria (e.g., name, supplier).
- Search Implementation (Optional): If searching is chosen, accept search criteria and query the database.
- Display Results: List all medicines (or search results) with details like name, description, quantity, etc.

Pseudocode:

```
FUNCTION ViewMedicineList()
  DECLARE displayOption, searchCriteria (optional)
  DISPLAY options: View all medicines or Search by criteria
  INPUT displayOption
  IF displayOption is "View All" THEN
    DISPLAY List of all medicines with details from database
  ELSEIF displayOption is "Search" THEN
    INPUT searchCriteria
    QUERY database for medicine matching criteria
    DISPLAY List of search results with details
  ELSE
    DISPLAY "Invalid option."
  ENDIF
END FUNCTION
```

4. Expired Medicine Tracking:

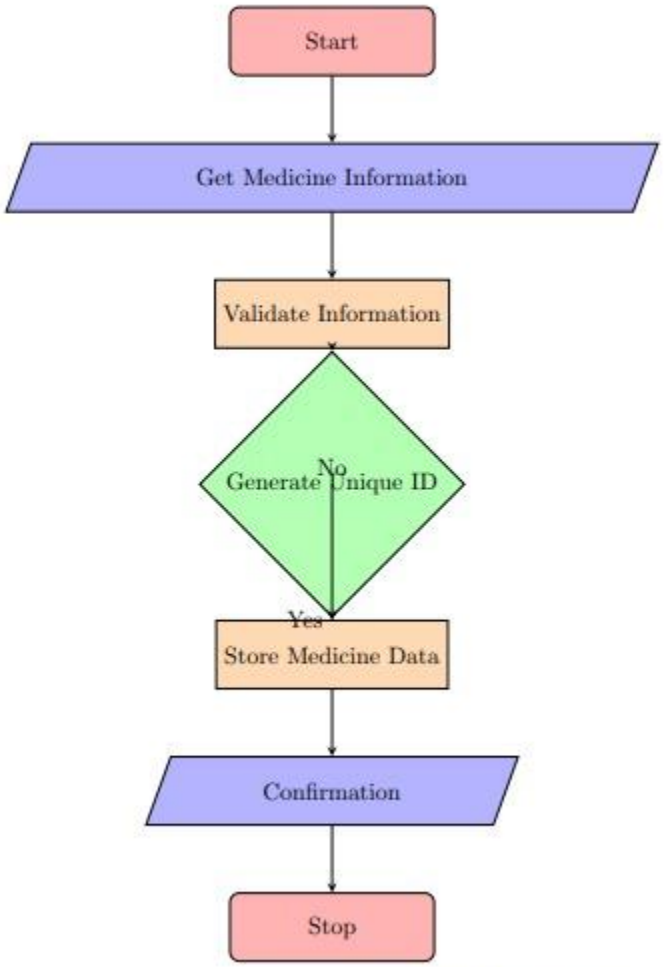
Algorithm:

- Query Database: Regularly search the medicine inventory database for entries with expiry dates approaching or exceeding the current date.
- Identify Expired Medicines: Flag or list the medicines identified as expired.
- Alert Pharmacy Manager: Provide a notification or report to the pharmacy manager regarding expired medicines.

Pseudocode

```
FUNCTION CheckForExpiredMedicines()
  DECLARE currentDate, expiredList
  Get current date
  QUERY database for medicines with expiryDate <= currentDate
  STORE expired medicines in expiredList
  IF expiredList is not empty THEN
    NOTIFY pharmacy manager about expired medicines in expiredList
  ENDIF
END FUNCTION
```

Flow Chart


Start
Get Medicine Information
Validate Information
Generate Unique ID
No
Yes
Store Medicine Data
Confirmation
Stop

Staff Management System -S A T Nayanapriya IT22892058

Test case 1: Adding Staff

Project ID: TP_2024_Y2_S2_WE10					
Project Name: Staff Management System					
Testing Function: Adding Staff					
Test Case ID: 1					
Test Priority: High					
Test Description: Verify that staff can be added successfully with all required details, and the data is saved in the MongoDB database.					
Test Steps: a. Step 1 : Fill in all required fields (username, first name, last name, email, role). b. Step 2 : Submit the form to add the staff.					
Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
	Valid values for username, first name, last name, email, and role	Staff should be added to the user list in the front-end. Data should be saved in the MongoDB database	Verify if the staff is displayed in the user list after submission. Check the MongoDB database to ensure the staff data is stored correctly	Pass if the staff is successfully added and data is saved in the database; fail otherwise	Check for error messages if any fields are missing or if the username/email already exists.

Staff Management System - S A T Nayanapriya IT22892058

Test case 2: Editing Staff

Project ID: TP_2024_Y2_S2_WE10					
Project Name: Staff Management System					
Testing Function: Editing Staff					
Test Case ID: 2			**Project Name:**		
Test Priority:					
Test Description: Verify that staff details can be edited and saved successfully, and the updated data is reflected in the MongoDB database.					
Test Steps: Step 1: Click on the "Edit" button for a staff member in the user list. Step 2: Modify the staff details. Step 3: Save the changes					
Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
	Modified values for staff details (e.g., username, first name, last name, email, role).	Staff details should be updated in the user list with the modified values. Updated data should be saved in the MongoDB database	Verify if the staff details are updated after saving changes. Check the MongoDB database to ensure the updated data is stored correctly.	Pass if the staff details are successfully updated and data is saved in the database; fail otherwise.	Ensure that the staff details are correctly reflected in the user list and database after editing

Staff Management System - S A T Nayanapriya IT22892058

Test case 3: Deleting Staff

Project ID: TP_2024_Y2_S2_WE10					
Project Name: Staff Management System					
Testing Function: Deleting Staff					
Test Case ID: 3			**Project Name:**		
Test Priority: High					
Test Description: Verify that staff can be deleted from the system, and the deleted data is removed from the database.					
Test Steps: Step 1: Click on the "Delete" button for a staff member in the user list.					
Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
	Selection of a staff member to delete.	Staff details The selected staff member should be removed from the user list. Deleted data should be removed from the MongoDB database.	Verify if the staff member is no longer present in the user list after deletion. Check the MongoDB database to ensure the deleted data is removed. .	Pass if the staff member is successfully deleted and data is removed from the database; fail otherwise	Ensure that the deletion process does not cause any unexpected behavior or errors.

Algorithm for Adding Staff:

Input: Staff details (username, first name, last name, email, role).
Process:

- Check for duplicate username or email.
- If duplicates found, display error message.
- If no duplicates:
  - Create a new user object with the provided details.
  - Add the user object to the user list.
  - Save the user list to the database.

Output: Success message if staff is added; error message if duplicates found.

Pseudo codes

Adding Staff

```
function addStaff(username, firstName, lastName, email, role):
    if duplicateUsername(username) or duplicateEmail(email):
        displayErrorMessage("A user with this username/email already exists.")
    else:
        newUser = createUserObject(username, firstName, lastName, email, role)
        addUserToList(newUser)
        saveUserListToDatabase()
        displaySuccessMessage("Staff added successfully.")

function duplicateUsername(username):
    // Check if username already exists in user list
    for user in userList:
        if user.username == username:
            return true
    return false

function duplicateEmail(email):
    // Check if email already exists in user list
```

```
    for user in userList:
        if user.email == email:
            return true
    return false

function createUserObject(username, firstName, lastName, email, role):
    newUser = {
        username: username,
        firstName: firstName,
        lastName: lastName,
        email: email,
        role: role
    }
    return newUser

function addUserToList(newUser):
    userList.append(newUser)

function saveUserListToDatabase():
    // Code to save user list to MongoDB database
```

Editing existing staff

```
function editStaff(index, newDetails):
    updateUserDetails(index, newDetails)
    saveUserListToDatabase()
    displaySuccessMessage("Staff details updated successfully.")

function updateUserDetails(index, newDetails):
    userList[index].username = newDetails.username
    userList[index].firstName = newDetails.firstName
    userList[index].lastName = newDetails.lastName
    userList[index].email = newDetails.email
    userList[index].role = newDetails.role
```

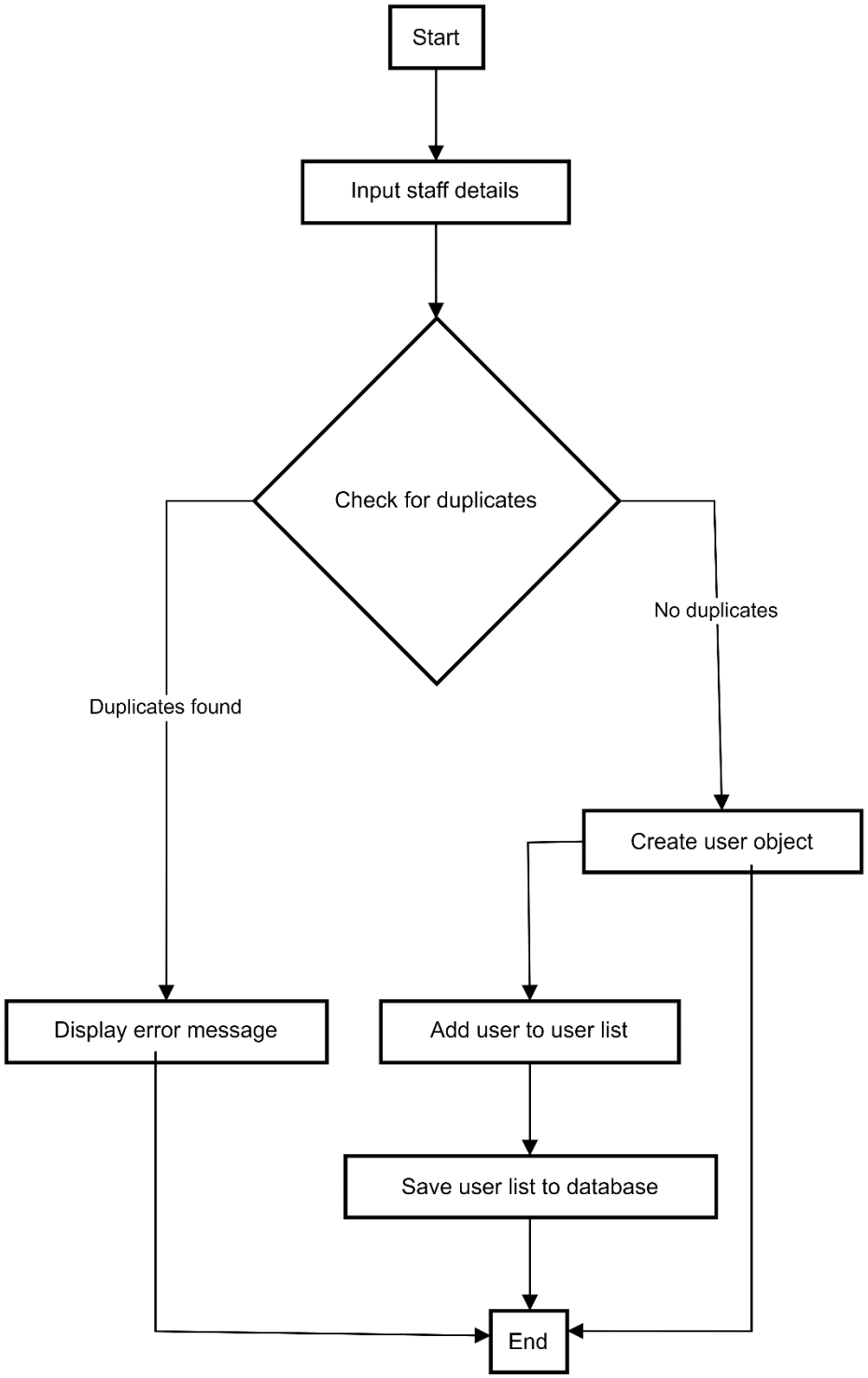
Deleting Staff

```
function deleteStaff(index):
    removeUserFromList(index)
    saveUserListToDatabase()
    displaySuccessMessage("Staff deleted successfully.")

function removeUserFromList(index):
    userList.(index, 1)
```

Searching Staff

```
function searchStaff(query):
    filteredUsers = filterUsers(query)
    displayFilteredUsers(filteredUsers)

function filterUsers(query):
    filteredUsers = []
    for user in userList:
        if matchesQuery(user, query):
            filteredUsers.append(user)
    return filteredUsers

function matchesQuery(user, query):
    // Check if user details match the search query
    if user.username contains query:
        return true
    else if user.firstName contains query:
        return true
    else if user.lastName contains query:
        return true
    else if user.email contains query:
        return true
    else if user.role contains query:
        return true
    return false

function displayFilteredUsers(filteredUsers):
    // Display filtered users in the user interface
```

Flow Chart


Start
Input staff details
Check for duplicates
Duplicates found
No duplicates
Create user object
Display error message
Add user to user list
Save user list to database
End

Appointment Scheduling Management System -Obeyesekere A D IT22332080

Test case 1:

Project ID: TP_2024_Y2_S2_WE10					
Project Name: Appointment Scheduling Management System					
Testing Function: Add an Appointment to the System					
Test Case ID: 1		**Project Name:**			
Test Priority:		High			
Test Description: Add an Appointment to a Doctor in a relevant health sector registered in the system.					
Test Steps: Step 1: Start from the homepage, the central hub of the system Step 2: navigate to the dashboard for an overview of Healthsectors. Step 3:Enter the patient details and the appointment details. Step 4: Display Overview of the available Doctors, Time and Dates Step 5: Find healthcare providers based on specialty or location. Step 6: Confirm the Appointment					
Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
1	First name Last name Date Specialization email member_id phone_number special_message doctor_id	The Appointment added must be saved to the database, and later displayed with the doctor selected and patient details entered as Appointment Scheduled.	Expected Output	Pass if Appointment details added successfully to DB, else Fail	Adding new appointment function worked

Test case 2:

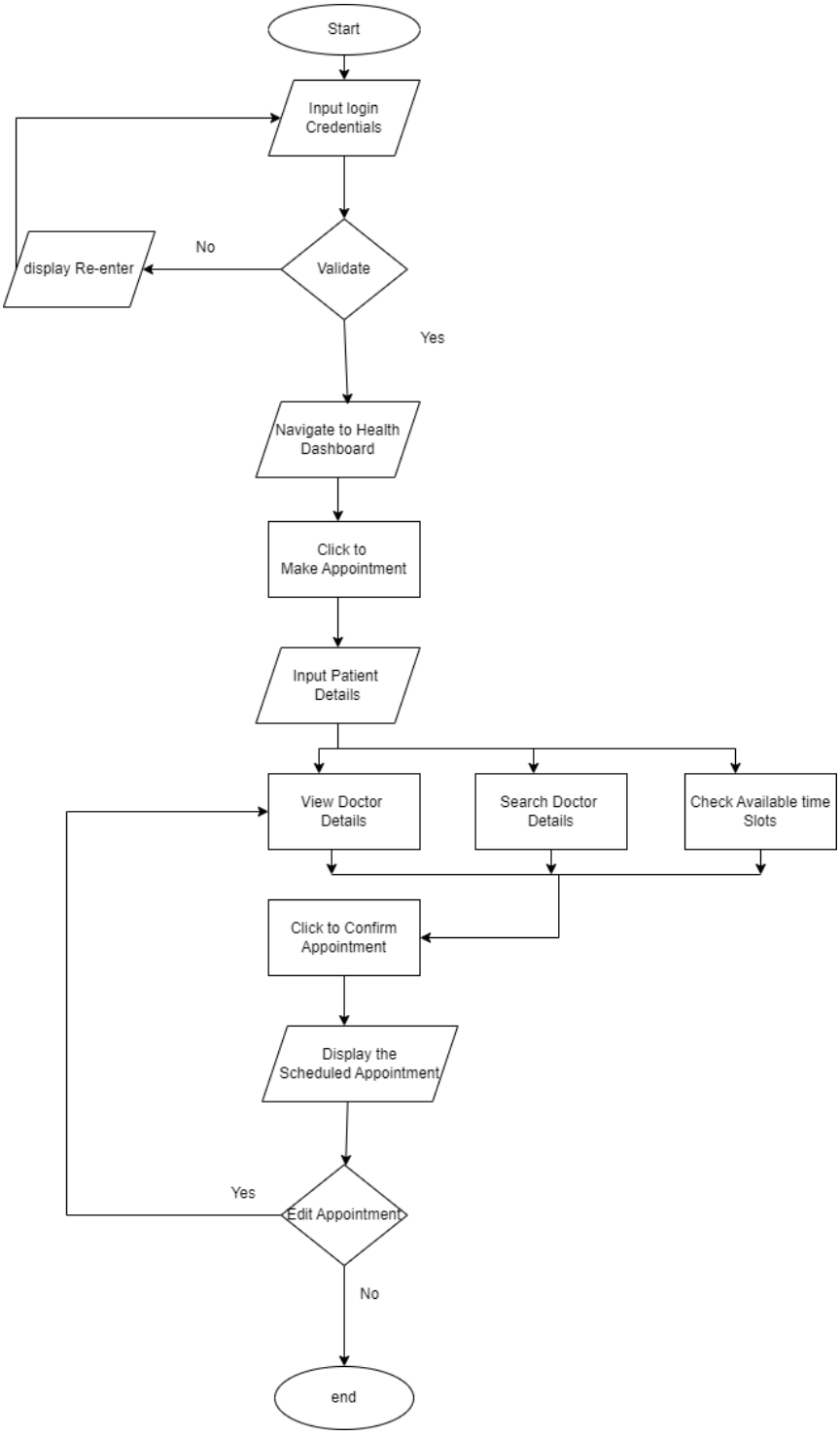
Project ID: TP_2024_Y2_S2_WE10	
Project Name: Appointment Scheduling Management System	
Testing Function:Update Appointment Scheduled	
Test Case ID: 2	**Project Name:**
Test Priority:	High
Test Description: Update a Scheduled appointment.	
Test Steps: Step 1: Start from the homepage, the central hub of the system Step 2: navigate to the dashboard for an overview of Healthsectors. Step 3: Enter the patient details and the appointment details. Step 4: Display Overview of the available Doctors, Time and Dates Step 5: Find healthcare providers based on specialty or location. Step 6: Confirm the Appointment. Step 7. Update the Appointment.	

Test ID	Test Inputs	Expected Outputs	Actual Output	Result (Pass/Fail)	Comments
2	First name Last name Date Specialization email member_id phone_number special_message doctor_id	The Appointment added must be saved to the database,and should be able to update the appointment details.	Expected Output	Pass	Adding new appointment function worked

Features and Completion level

Completed	Work in progress	To be completed
View Available Doctors Overview of Appointment Scheduled.	Generate Appointment report.	Generate Appointment report.
Create and update appointments.		
Search for Doctors		

FlowChart for Appointment Scheduling



Algorithm Associated

1.Input login Credentials:
1.1User enters username and password.
2.Validate:
2.1System checks the entered credentials for authentication.
3.Navigate to Health Dashboard:
3.1Upon successful validation, the system redirects the user to the health dashboard.
4.Click to Make Appointment:
4.1User clicks on the "Make Appointment" button .
5.Input Patient Details:
5.1User provides necessary patient information such as name, contact details, etc.
6.View Doctor Details, Search Doctor Details, Check Available Time Slots:
6.1User can view doctor details, search for specific doctors, and check their available time slots all within the same interface.
7.Click to Confirm Appointment:
7.1After selecting a doctor and preferred time slot, user clicks on the "Confirm Appointment" button.
8.Display the Scheduled Appointment:
8.1System confirms the appointment and displays the scheduled appointment details to the user.
9.Edit Appointment (if needed, navigated to edit Appointment Page):
9.1If modifications are required, user navigates to the edit appointment page where they can update details such as date, time, or doctor.

Peudocode for Create Appointment

```
FUNCTION InsertAppointment()

 DECLARE firstName, lastName, dateOfBirth, doctorSpecialization, email, memberID, phoneNumber, specialMessage, doctorID, appointmentID

 INPUT firstName, lastName, dateOfBirth, doctorSpecialization, email, memberID, phoneNumber, specialMessage, doctorID

 IF any field is empty THEN
   DISPLAY "Error: Missing information!"
 ELSE
   IF dateOfBirth is not in valid date format THEN
     DISPLAY "Error: Invalid date format for date of birth!"
   ELSE
     IF doctorSpecialization is not valid THEN
       DISPLAY "Error: Invalid doctor specialization!"
     ELSE
       IF appointment already exists for the same patient and doctor THEN
         DISPLAY "Error: Appointment already exists!"
       ELSE
         IF unique appointment ID generation required THEN
           Generate unique appointment ID
         ENDIF
         STORE appointment data in database
         DISPLAY "Appointment scheduled successfully! Appointment ID: (appointmentID)"
       ENDIF
     ENDIF
   ENDIF
 ENDIF

END FUNCTION
```

Peudocode for Update Appointment

```
FUNCTION UpdateAppointment()

  DECLARE appointmentID, newAppointmentDate, newAppointmentTime

  INPUT appointmentID, newAppointmentDate, newAppointmentTime

  IF appointmentID is empty OR newAppointmentDate is empty OR newAppointmentTime is empty THEN
    DISPLAY "Error: Missing information!"
  ELSE
    IF newAppointmentDate is not in valid date format OR newAppointmentTime is not in valid time format THEN
      DISPLAY "Error: Invalid date or time format!"
    ELSE
      IF appointment does not exist with the given appointmentID THEN
        DISPLAY "Error: Appointment not found!"
      ELSE
        UPDATE appointment data in database
        DISPLAY "Appointment updated successfully!"
      ENDIF
    ENDIF
  ENDIF

END FUNCTION
```

SQL Queries

No SQL queries used.

Test case 1: Adding a Patient

Project ID: TP_2024_Y2_S2_WE10					
Project Name: Patient Management System					
Testing Function: Add Patient					
Test Case ID: TC_ADD_01		**Project Name: Test Add New Patient**			
Test Priority: High					
Test Description: This test case checks if a new patient can be added to the system with all required information.					
Test Steps: Step 1: Navigate to the "Add Patient" section. Step 2: Enter all required patient details into the form. Step 3: Click the "Save" button to submit the information. Step 4: Check for a confirmation message with the assigned patient ID					
Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
ADD_01	Name: John Doe ID: (Should be auto-generated) Address: 123 Health St. Phone Number: 555-1234 Insurance Information: HealthInsure Co. Policy #98765	A confirmation message saying "Patient successfully added with ID: [UniquePatientID]"	A confirmation message saying "Patient successfully added with ID: P123456"	PASS	The patient was successfully added, and a unique ID was generated and displayed.

Test case 2: Deleting a Patient

Project ID: TP_2024_Y2_S2_WE10	
Project Name: Patient Management System	
Testing Function: Delete Patient	
Test Case ID: TC_DEL_01	**Project Name: Test Delete Existing Patient**
Test Priority: Medium	
Test Description: This test case verifies that an existing patient can be deleted from the system after confirmation.	
Test Steps: Step 1: Use the search function to find the patient with ID P123456. Step 2: Select the patient record found. Step 3: Click the "Delete" button. Step 4: Confirm the deletion when prompted.	

Test ID	Test Inputs	Expected Outputs	Actual Output	Result (Pass/Fail)	Comments
DEL_01	Patient ID: P123456	A confirmation message saying "Patient record deleted successfully."	A confirmation message saying "Patient record deleted successfully."	PASS	The patient record was successfully deleted after confirmation

Features and Completion level

Completed	Work in progress	To be completed
View patient records	Generate system reports	Generate system reports
Create and update records		
Delete existing records		

SQL queries used

- None SQL base

Algorithms and pseudo-codes

1. Adding a Patient:

Algorithm:

- Get Patient Information: Collect details like name, ID, address, phone number, insurance information, etc.
- Validate Information: Check for missing fields or invalid data formats.
- Generate Unique ID (Optional): If the system doesn't assign IDs, generate a unique identifier.
- Store Patient Data: Save the information in the patient database.
- Confirmation: Provide confirmation message with the assigned ID (if applicable).

Pseudocode:

```
FUNCTION AddPatient()
  DECLARE name, address, phone, insurance, etc.
  INPUT name, address, phone, insurance, etc.
  IF any field is empty THEN
    DISPLAY "Error: Missing information!"
  ELSE
    IF unique ID generation required THEN
      Generate unique ID
    ENDIF
    STORE name, address, phone, insurance, etc. in database
    DISPLAY "Patient added successfully! ID: (uniqueID)" (if applicable)
  ENDIF
END FUNCTION
```

2. Searching for a Patient:

Algorithm:

- Get Search Criteria: Allow searching by ID, name, or other relevant fields.
- Query Database: Search the patient database based on the provided criteria.
- Display Results: If a match is found, display the patient's information. Otherwise, provide a "Not Found" message.

Pseudocode:

```
FUNCTION SearchPatient(criteria)
  DECLARE results
  IF criteria is empty THEN
    DISPLAY "Error: Enter search criteria!"
  ELSE
    QUERY database for patients matching criteria
    STORE results in list
    IF results is empty THEN
      DISPLAY "Patient not found."
    ELSE
      DISPLAY Patient information from results list
    ENDIF
  ENDIF
END FUNCTION
```

3. Updating Patient Information:

Algorithm:

- Search for Patient: Use the search function to locate the patient to be updated.
- Select Fields to Update: Allow selection of specific fields to modify (e.g., address, phone number).
- Get Updated Information: Collect the new values for the chosen fields.
- Update Database: Modify the corresponding entries in the patient database.
- Confirmation: Display a confirmation message indicating successful update.

Pseudocode:

```
FUNCTION UpdatePatient(patientID)
  DECLARE updatedFields, newData
  SearchPatient(patientID)  // Call search function
  IF patient not found THEN
    DISPLAY "Patient not found."
  ELSE
    SELECT updatedFields from user
    INPUT newData for updatedFields
    UPDATE database with newData for chosen fields in patient record
    DISPLAY "Patient information updated successfully."
  ENDIF
END FUNCTION
```

4. Deleting a Patient:

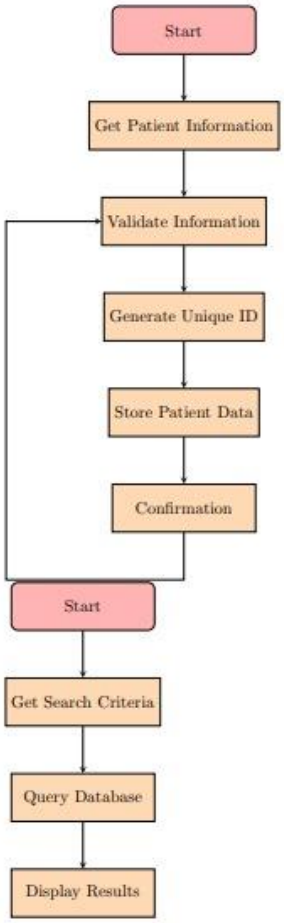
Algorithm:

- Search for Patient: Use the search function to locate the patient to be deleted.
- Confirmation: Prompt the user for confirmation to avoid accidental deletion.
- Delete Patient Record: If confirmed, remove the patient's information from the database.
- Confirmation: Display a message indicating successful deletion (or cancellation if not confirmed).

Pseudocode:

```
FUNCTION DeletePatient(patientID)
  SearchPatient(patientID)  // Call search function
  IF patient not found THEN
    DISPLAY "Patient not found."
  ELSE
    DISPLAY "Are you sure you want to delete this patient? (y/n)"
    INPUT confirmation
    IF confirmation is "y" THEN
      DELETE patient record from database
      DISPLAY "Patient deleted successfully."
    ELSE
      DISPLAY "Patient deletion cancelled."
    ENDIF
  ENDIF
END FUNCTION
```

Flow chart


Start
Get Patient Information
Validate Information
Generate Unique ID
Store Patient Data
Confirmation
Start
Get Search Criteria
Query Database
Display Results

Claim Management System - M F M Farsith IT22354556

Test case 1:

Project ID: TP_2024_Y2_S2_WE10					
Project Name: Claim Management System					
Testing Function: Create Claim					
Test Case ID: 1			**User : Staff**		
Test Priority: High					
Test Description: Create Claim for a Patient					
Test Steps: Step 1: Login to the system Step 2: Enter the Claim Overview Interface Step 3: Click on the "Create Claim" buttom Step 4: Click the agreement checkbox and create claim. Step 5: Confirm by getting the prompt Modal.					
Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
ClaimTest01	MemberId, FirstName, LastName, DOB, Amount, Receipt	Confirmation Modal appearing and details stored in DB	Confirmation Modal appearing and details stored in DB	Pass	Checking on the possibility of constructing a Modal to be printed.

Claim Management System - M F M Farsith IT22354556

Test case 2:

Project ID: TP_2024_Y2_S2_WE10					
Project Name: Claim Management System					
Testing Function: Approve Claim					
Test Case ID: 2			**User : Staff**		
Test Priority: High					
Test Description: Approve Pending Claims for Patients					
Test Steps: Step 1: Login to the system Step 2: Enter Approve Claim Interface Step 3: Check on the unapproved claims Step 4: Validate the claim by double verification Step 5: Approve the claim and notify the patient					
Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
Approve Claim Test01	MemberId, ClaimId, Name, Amount, Receipt	Change of the claim status in the overview interface after the approval.	Change of the claim status in the overview interface after the approval.	Fail	Checking on the progress of the current approval and verification methods.

Description

The Claim Management module of the Innobot Insurance System simplifies claim processing for insurers and policyholders. It allows for easy initiation, tracking, and settlement of claims. Administrators can customize workflows and access insightful analytics for better decision-making. Comprehensive reporting features offer detailed insights into claim status and performance metrics, enhancing operational efficiency and customer satisfaction.

Work in Progress	To be Completed
Read Claim Profiles	Approve Claim Function Activity
Verify Claim Record	Access Claim Modification
Delete Claim record	
Assign Task	

SQL queries used

- None SQL base.

Algorithms and pseudo-codes

1. Adding a Claim:

Algorithm:

Get Claim Information: Collect details like claim ID, member ID, diagnosis codes, date of claim, receipt.
Validate Information: Check for missing fields or invalid data formats.
Generate Unique Claim ID.
Store Claim Data: Save the information in the claim database.
Confirmation: Provide a confirmation notification with the assigned claim ID.

Pseudocode:

```
FUNCTION AddClaim()
  DECLARE claimID, patientID, diagnosisCodes, dateOfService, providerInfo, etc.
  INPUT claimID, patientID, diagnosisCodes, dateOfService, providerInfo, etc.
  IF any field is empty THEN
    DISPLAY "Error: Missing information!"
  ELSE
    IF unique Claim ID generation required THEN
      Generate unique Claim ID
    ENDIF
    STORE claimID, patientID, diagnosisCodes, dateOfService, providerInfo, etc. in database
    DISPLAY "Claim added successfully! Claim ID: (claimID)" (if applicable)
  ENDIF
END FUNCTION
```

2. Updating a Claim

Algorithm:

Search for Claim: Use the search function to locate the claim to be updated.
Select Fields to Update: Allow selection of specific fields to modify.
Get Updated Information: Collect the new values for the chosen fields.
Update Database: Modify the corresponding entries in the claim database.
Confirmation: Display a confirmation message indicating successful update.

Pseudocode:

```
FUNCTION UpdateClaim(claimID)
  DECLARE updatedFields, newData
  SearchClaim(claimID)  // Call search function
  IF claim not found THEN
    DISPLAY "Claim not found."
  ELSE
    SELECT updatedFields from user
    INPUT newData for updatedFields
    UPDATE database with newData for chosen fields in claim record
    DISPLAY "Claim information updated successfully."
  ENDIF
END FUNCTION
```

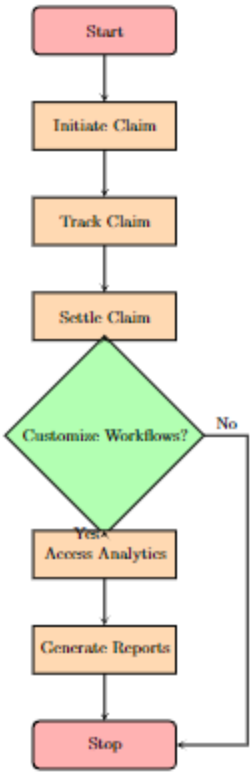
3. Delete Claim

Algorithm:

Search for Claim: Use the search function to locate the claim to be deleted.
Confirmation: Prompt the user for confirmation to avoid accidental deletion.
Delete Claim Record: If confirmed, remove the claim's information from the database.
Confirmation: Display the rest of the claims in the Overview

Pseudocode:

```
FUNCTION DeleteClaim(claimID)
  SearchClaim(claimID)  // Call search function
  IF claim not found THEN
    DISPLAY "Claim not found."
  ELSE
    DISPLAY "Are you sure you want to delete this claim? (y/n)"
    INPUT confirmation
    IF confirmation is "y" THEN
      DELETE claim record from database
    ELSE
      DISPLAY "Claim deletion cancelled."
    ENDIF
  ENDIF
END FUNCTION
```


Start
Initiate Claim
Track Claim
Settle Claim
Customize Workflows?
No
Yes
Access Analytics
Generate Reports
Stop

Insurance Management System - Chamikara M G S IT22905840

Test case 1:

Project ID: **ITP_2024_Y2_S2_WE10**					
Project Name: Health Care Management System "Innobot Health"					
Testing Function: Create New Insurance Record					
Test Case ID: 1		**Test Case Designed By,** **ID: IT22905840** **Name:** Chamikara M G S			
Test Priority:		High			
Test Description: verifies the functionality to create a new insurance record.					
Test Steps: Step 1: Login to the system Step 2: Go to the Create New Record function Step 3: Input valid insurance record details Step 4: Call the Create New Insurance Record() function					
Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
1	Insurance Provider Coverage Details Effective Date Expiry Date Premium Amount	Display "Insurance record create successfully"	Expected output	pass	

Insurance Management System - Chamikara M G S IT22905840

Test case 2:

Project ID: ITP_2024_Y2_S2_WE10	
Project Name: Health Care Management System "Innobot Health"	
Testing Function: Delete/Archive Insurance Record ()	
Test Case ID:2	**Test Case Designed By,** **ID: IT22905840** **Name:** Chamikara M G S
Test Priority:	High
Test Description: Verify that an existing insurance record can be deleted or archived successfully	
Test Steps: Step 1: Call the delete/Archive Insurance Record() function Step 2: Provide the ID of the insurance record need to delete Step 3: Confirm the deletion/archival action when prompted Step 4: Verify that the insurance record is successfully deleted or archived	

Test ID	Test Inputs	Expected Outputs	Actual Output	Result (Pass/Fail)	Comments
2	Insurance Record ID	Display "Insurance record deleted/archived successfully"	Expected output	pass	delete/archive function worked.

Description

The Insurance Management module of the Innobot Hospital Management System streamlines insurance-related processes for healthcare providers and patients. This module facilitates efficient management of insurance records, enabling users to create, update, and maintain comprehensive insurance information within the system. Administrators are empowered with tools to define and manage fee schedules for different CPT codes, enabling precise reimbursement calculations. Furthermore, the system generates comprehensive insurance records reports and provides robust analytics features.

Completed	Work in progress	To be completed

View insurance records	Define fee schedules	Generate insurance reports
Create and update records		
Delete existing records		

SQL queries used

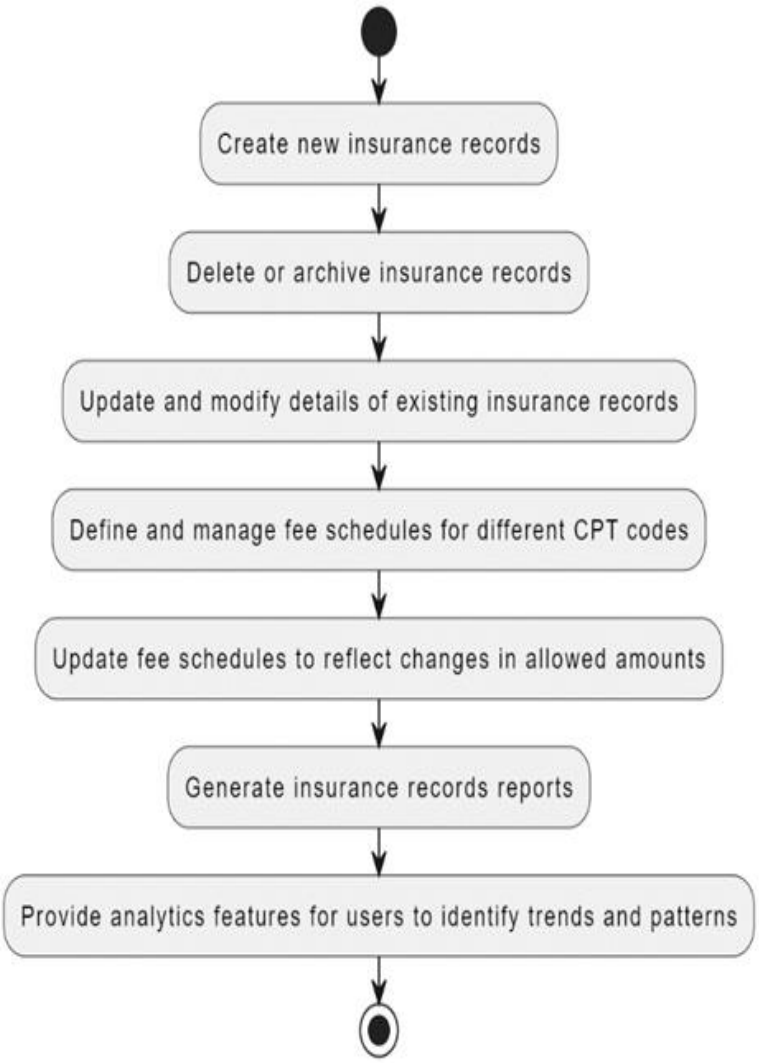
- None SQL base

Algorithm

1. **User Authentication:** Users input login credentials, and the system validates them. If valid, they proceed; otherwise, they try again.
2. **Insurance Record Management:** Users can create new records, update existing ones, delete/archive records, or view existing records.
3. **Create New Insurance Record:** Users input details for a new record. If valid, the record is created; otherwise, they try again.
4. **Update Existing Insurance Record:** Users select and update a record. If valid, the record is updated; otherwise, they try again.
5. **Delete/Archive Insurance Record:** Users select and confirm deletion/archival of a record. If confirmed, the record is deleted/archived; otherwise, they go back**.**
6. **View Insurance Records:** Users can view existing records and perform actions like filtering/sorting or generating reports.
7. **Filter/Sort Insurance Records:** Users input criteria to filter/sort records. If valid, records are filtered/sorted; otherwise, they try again.
8. **Generate Reports:** Users select a report type and input parameters. If valid, the report is generated; otherwise, they try again**.**
9. **End:** The algorithm concludes after users have completed their desired actions.

Pseudocode

```
Begin
 UserAuthentication():
    Input username and password
    If credentials are valid:
        Display "Login successful"
        Return true
    Else:
        Display "Invalid username and password"
 Return false

CreateNewInsuranceRecord():
    Input insurance details
    If details are valid:
        Create new record
        Display success message
    Else:
        Display error message

UpdateExistingInsuranceRecord():
    Input record to update and new details
    If record exists and details are valid:
        Update record
        Display success message
    Else:
        Display error message

DeleteOrArchiveInsuranceRecord():
    Input record to delete/archive and confirmation
    If confirmed:
        Delete/archive record
        Display success message
 Else:
        Display cancellation message

ViewInsuranceRecords():
    Display list of records
    Repeat until user exits:
        Input option
        If option is to filter/sort:
            FilterOrSortInsuranceRecords()
        Else if option is to generate reports:
            GenerateReports()
        Else if option is to go back:
            Return to main menu

FilterOrSortInsuranceRecords():
    Input criteria
    If criteria is valid:
        Filter/sort records
        Display filtered/sorted records
    Else:
        Display error message
End
```


Create new insurance records
Delete or archive insurance records
Update and modify details of existing insurance records
Define and manage fee schedules for different CPT codes
Update fee schedules to reflect changes in allowed amounts
Generate insurance records reports
Provide analytics features for users to identify trends and patterns

Test case 1:

Project ID: TP_2024_Y2_S2_WE10					
Project Name: Notification Management System					
Testing Function: Send Notifications					
Test Case ID: 1			**Project Name: Send new notification**		
Test Priority: 1					
Test Description: create a new notification and send					
Test Steps: Step 1: log into the system by using email and password Step 2: Navigate to the notifications section and tap on '+create new' Step 3: Fill the forms as per to the validations. Step 4: Click and 'send notification'					
Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
1	Category:custom Receiver_type: Doctor Receiver: all Subject: Test bulk notification Message: Test Notification Annonuynous: not selected Priority: High Scheduled/Instant: Instant	202 response status, and the notificatin should be sent to all the doctors available in the system instantly.	The notification has been sent to the doctors based on their notification preference.	PASS	The record has also been created in the database successfuly.

Test case 2:

Project ID: TP_2024_Y2_S2_WE10					
Project Name: Notification Management System					
Testing Function: Receive Notifications					
Test Case ID: 2		**Project Name: Receive Notifications**			
Test Priority: 2					
Test Description: Users receive system notifications once logged in.					
Test Steps: Step 1: Log into the system with email and password Step 2: Click on the notification icon in the top bar.					
Test ID	**Test Inputs**	**Expected Outputs**	**Actual Output**	**Result (Pass/Fail)**	**Comments**
1	User's access token	The system notifications received by the user should be listed down.	The notifications received by the logged in user listed down with acknowledge and reply to buttons.	PASS	User's id has been embeded in the JWT access token and BE system authorized the user by using that.

SQL queries used
None SQL base.

Notifications Management System - DME Wimalagunasekara_IT22917270

The notification system within the healthcare management system serves to inform stakeholders about relevant updates and actions within the system. Stakeholders, including admins, staff members, coordinators, doctors, and insurance companies, can receive notifications through three channels: SMS, email, and system notifications.

Notification Channels

- **SMS**: Stakeholders who opt to receive notifications via SMS will be sent text messages containing relevant information.
- **Email**: Notifications can also be sent via email, containing detailed information about the updates or actions.
- **System Notifications**: Within the system itself, stakeholders will receive notifications directly on their dashboard. These notifications can be interacted with, allowing stakeholders to acknowledge and respond to them directly within the system.

Notification Preferences

At the time of account creation, stakeholders can specify their notification preferences, indicating which channels they prefer to receive notifications through.

Notification Triggers

Notifications are triggered by various actions and events within the system, such as appointment bookings, cancellations, claim approvals, and other significant activities.

For example, when a patient books an appointment, the doctor assigned to the appointment, as well as any relevant staff members and coordinators, will receive notifications based on their specified preferences.

Bulk Notifications

Users have the capability to send bulk notifications to multiple stakeholders simultaneously. This feature is particularly useful for broadcasting important updates or announcements to a large group of recipients.

Acknowledgment and Reply

Stakeholders can acknowledge receipt of notifications and respond to them directly within the system. This two-way communication enhances collaboration and ensures that all parties are informed and can take appropriate actions as needed.

Integration with System Actions

The notification system is seamlessly integrated with various system actions and processes. Whenever a significant action is performed, such as appointment management or claim processing, relevant stakeholders are automatically notified to keep them informed in real-time.

Customization and Scalability

The notification system is designed to be customizable and scalable, allowing for the addition of new notification types and channels as the system evolves. This ensures that stakeholders can stay updated on relevant information efficiently and effectively.

Scheduled Notifications

Users have the ability to schedule notifications to be sent at a particular time. This feature allows for the timely delivery of notifications, ensuring that stakeholders receive important information exactly when it's needed.

Anonymous Notifications

Users can also send notifications anonymously if required. This feature allows for sensitive or confidential notifications to be delivered without revealing the identity of the sender, providing flexibility and privacy where needed.

By implementing this robust notification system, the healthcare management system ensures that stakeholders are kept informed about important updates and actions, facilitating seamless communication and collaboration within the healthcare ecosystem.

The Completion Level of Features

Work Completed	Work Completed by Final
User authentication and authorization	Reply to received notifications
Send bulk notifications via SMS, email and system notifications	Search sent notifications
Send single notifications	Search received notifications
Send anonymous notifications	Finetune and UI enhancements
Send scheduled notifications	
View sent notifications	
Edit and update sent notifications	
Delete sent notifications	
View received system notifications	
Acknowledge received system notifications	

Algorithm

1. **Start**
   - The notification management system initializes.

2. **User Authentication**
   - The user logs in by providing their credentials (email and password).

3. **Authentication Validation**
   - The system validates the provided credentials server-side.
   - If the credentials are valid
     - Proceed to step 4.
   - If the credentials are invalid
     - Display "Invalid username or password" message.
     - Return to step 2 for the user to input login credentials again.

4. **Authentication Successful**
   - Redirect the user to the appropriate page based on their role after successful sign-in.

5. **Notification Sending Interface**
   - Upon successful authentication and role verification
   - Provide the user with an interface to send notifications.
   - Options include selecting recipients (admins, staff members, coordinators, doctors, insurance companies), composing the message, and scheduling the notifications for a particular time.
   - Additionally, provide an option for the user to send notifications anonymously if required.

6. **Notification Sending**
   - After the user inputs the notification details and sends the notification
   - The system processes the notification requests based on the notification-preference of the selected receiver/s.
   - Notifications are sent to the selected recipients through the specified channels.
   - If scheduled, notifications are queued (in-memory) for delivery at the specified time.

7. **Acknowledgment and Reply Handling**
   - Recipients receive notifications through their preferred channels.
   - Recipients can acknowledge receipt of notifications and reply to them directly within the system (if system notification has been received).

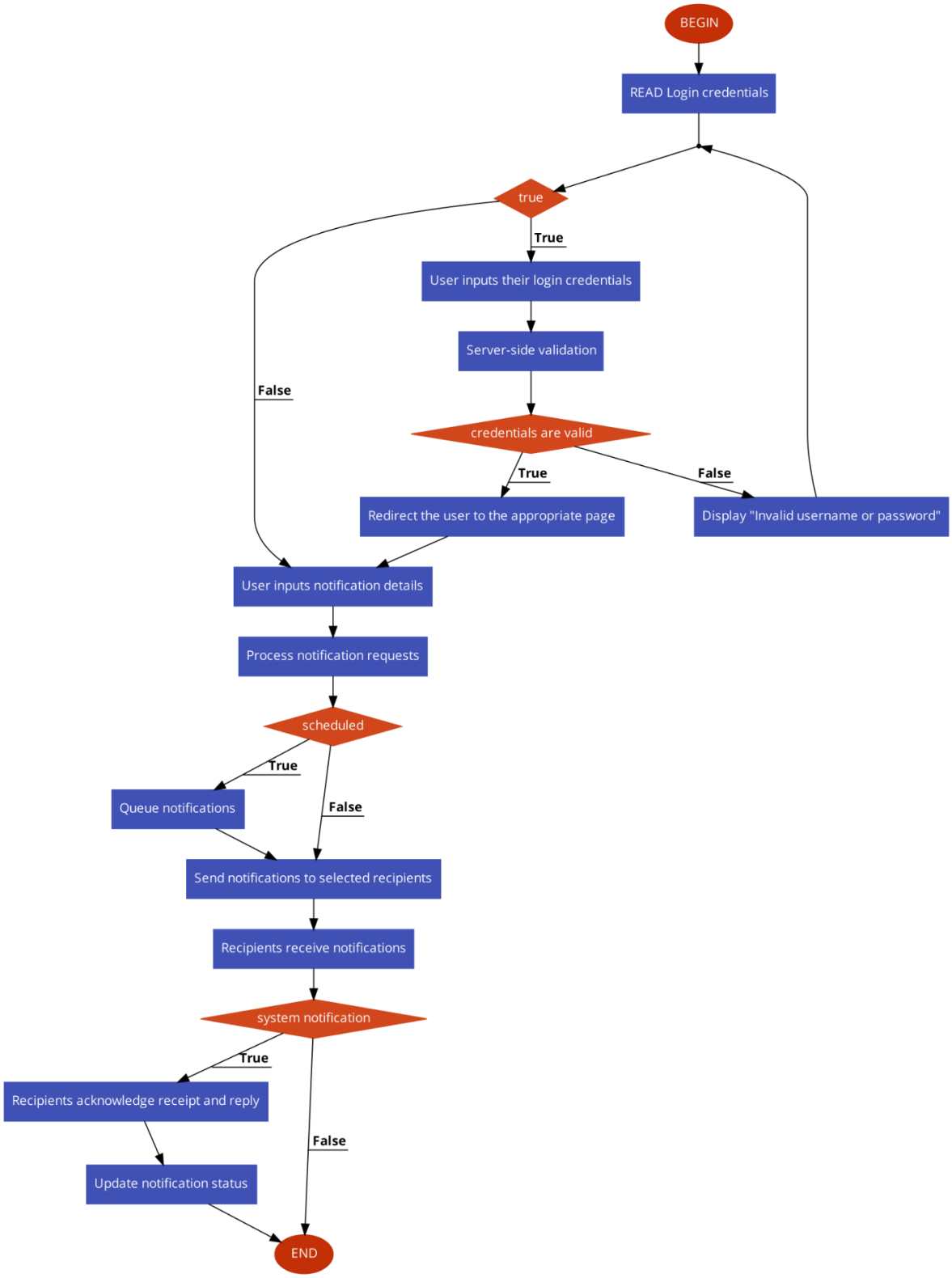
- The system handles acknowledgment and replies, updating the notification status accordingly.

8. **Navigate to Other System Actions**
   - After managing notifications, the user can navigate to other system actions such as appointment booking, cancellations, claim approvals, etc., if needed.

Pseudocode

```
BEGIN
   READ Login credentials.
   WHILE true
      User inputs their login credentials (email and password).
      Server-side validation verifies the authenticity of the entered login credentials.
      IF credentials are valid THEN
         Redirect the user to the appropriate page based on their role after successful sign-in.
         BREAK loop
      ELSE
         Display "Invalid username or password" message.
         Continue loop.
      ENDIF
ENDWHILE

Provide the user with an interface to send notifications.
Options include selecting recipients, composing the message, selecting channels, and scheduling
notifications.
Provide option for anonymous notifications if required.

   WHILE true
      User inputs notification details (recipients, message, channels, schedule_time)
      Process notification requests.
      Send notifications to selected recipients through specified channels.
      If scheduled, queue notifications for delivery at scheduled time.
      Recipients receive notifications through preferred channels.
      Recipients can acknowledge receipt and reply within the system.
      Update notification status based on acknowledgments and replies.

      After managing notifications, user can navigate to other system actions as needed.
      BREAK loop if user chooses to exit notification management system.
   ENDWHILE

END
```

Flowchart


BEGIN
READ Login credentials
true
True
User inputs their login credentials
Server-side validation
False
credentials are valid
True
False
Redirect the user to the appropriate page
Display "Invalid username or password"
User inputs notification details
Process notification requests
scheduled
True
Queue notifications
False
Send notifications to selected recipients
Recipients receive notifications
system notification
True
Recipients acknowledge receipt and reply
False
Update notification status
END