Integrated Health Systems

Madeline Abbott

Ben Hogan

INFO-C450

Professor Jafrina Jabin

Table of Contents

Title: Problem Statements & System Requirements

Madeline Abbott and Benjamin Hogan

Problem Statement

Managing patient information is complex and time-consuming, especially when using outdated, convoluted, or disjointed systems. Healthcare providers often find themselves juggling multiple outdated systems at once, which can lead to delays in accessing essential patient and system data, causing an overall decrease in efficiency and increase in risk of error. There creates a dire need for a standardized solution that simplifies patient data management, improves accessibility, and ensures data security.

Glossary of Terms

- **Medical History**: A record of a patient's past medical problems, treatments, and information.
- **Patient Record**: A digital file containing information about a given patient's medical history, treatments, personal information, active medications, etc.
- **Medical Report**: A formal document summarizing a patient's current condition, treatments, etc. This document is essential for billing, insurance, and more.
- **User Role**: A level of access a given user has access to, including "provider," and "administrator."

Functional Requirements:

No.	**Priority**	**Description**
REQ-1	High	Providers should be able to add, update, and delete records as necessary.
REQ-2	High	Providers should be able to facilitate and maintain appointment scheduling.

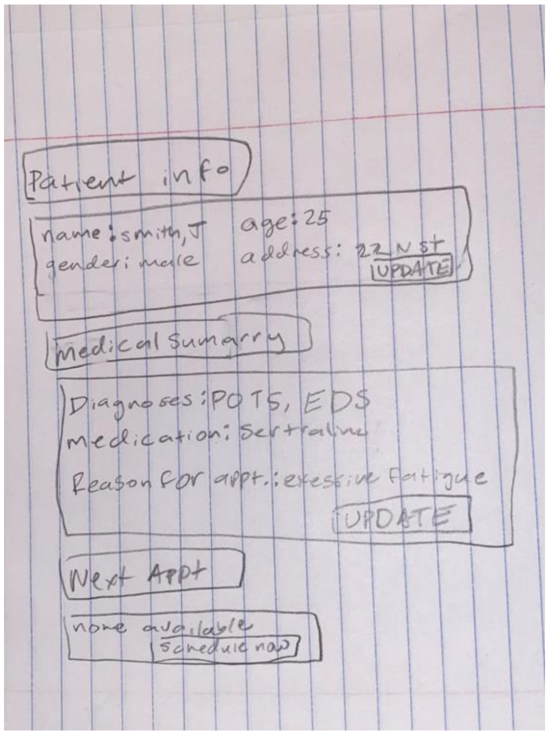
REQ-3	Medium	The system should be able to generate a medical report for each patient visit.
REQ-5	Medium	The system should employ secure user authentication and role management.
REQ-6	High	The system should be secure in compliance with healthcare data security regulations.
REQ-7	Medium	The system should provide a user friendly interface for both providers and administrators.
REQ-8	Low	The system should employ a secure password recovery system.

Nonfunctional Requirements

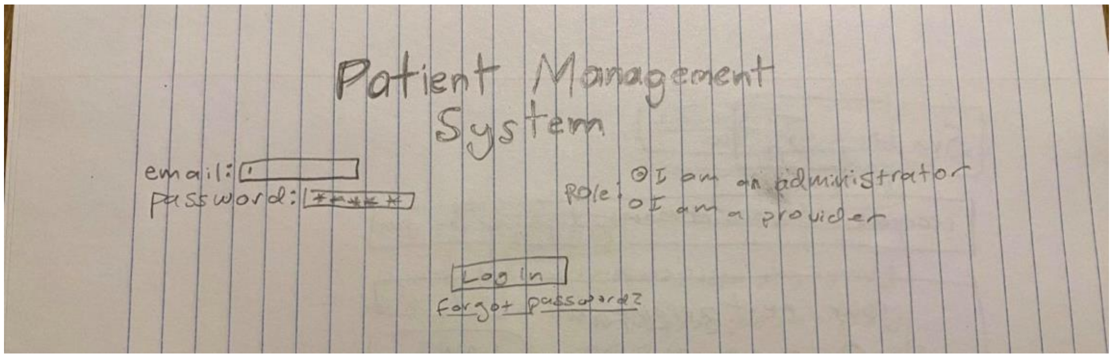
Nonfunctional Requirement	**Priority**	**Description**
Functionality	High	Providers should be able to add, update, and delete records as necessary.
Usability	High	The interface should be both intuitive and user-friendly for providers, patients, and administrators.
Reliability	Medium	The system should aim to have an uptime of 99.8%.
Performance	Medium	The system should load patient records in under 5 seconds.
Supportability	Low	The system should allow for updates.

User Interface Requirements

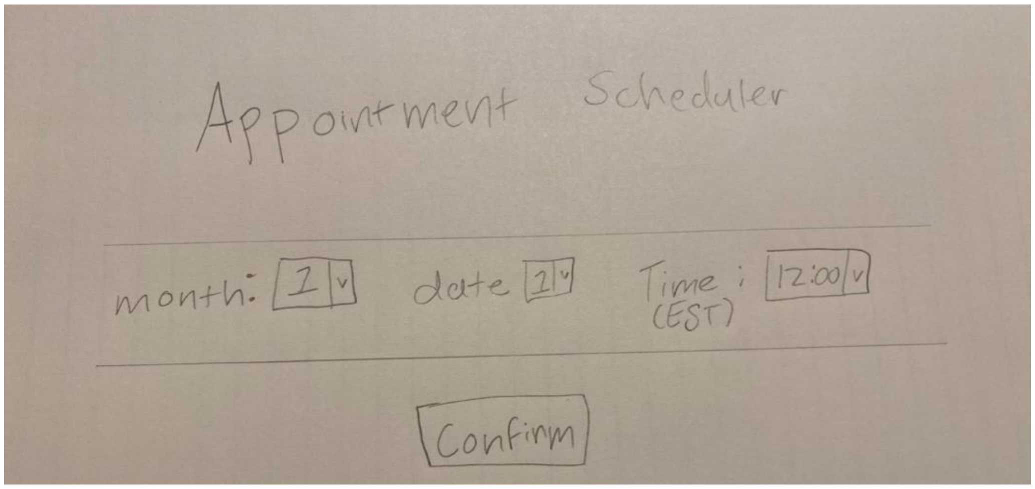
- **Patient Record Management Interface**: A user-friendly interface for providers to view and manage patient records with.

- Priority: High
- Sketch:
- 

- **Account Log-In Interface**: An interface where providers and administrators can log in to their account.
  - Priority: High
  - Sketch:



- **Appointment Scheduling Interface**: An interface to manage appointments, availability, and scheduling.
  - Priority: Medium
  - Sketch:



Development Plan Progress

- **Weeks 1-2**:
  - Worked on setting up and configuring the development environments and databases.
- **Weeks 3-4**:
  - Currently brainstorming the frontend interface with HTML, CSS, and JavaScript.

Patient Management System

Stake Holders

1. Health Networks
2. Hospitals
3. Doctors
4. Nurse Practitioners
5. Urgent Care

Actors and Goals

Primary Actors

1. Health Care Providers- This would include doctors, nurses, nurse practitioners and anyone else who is a part of the treatment team. They would need access to make changes, updates, and remove information as a patient moves through the process. They would have access to the system for the specific patients that they have. They would not have access to a patient outside of their responsibilities.

Secondary Actors

1. Admin- They would oversee training and making sure the patient data is up to date and properly filled out. They would have full access to the system as they needed.
2. System- Able to be interacted with. Store data entered in and able to recall data when needed.

Cases

Admin –

1.Log in and off(3)

2. Add/ Remove health records(3)

3. Access all patient record(3)

4. Check and adjust scheduling.(3)

5. Assign permissions for users.(3)

Health Care Providers-

1.Log in and off(3)

2. Add/ Remove health records (3)

3. Access all patient record (3)

4. Check and adjust scheduling. (3)

5. Generate reports and labs. (3)

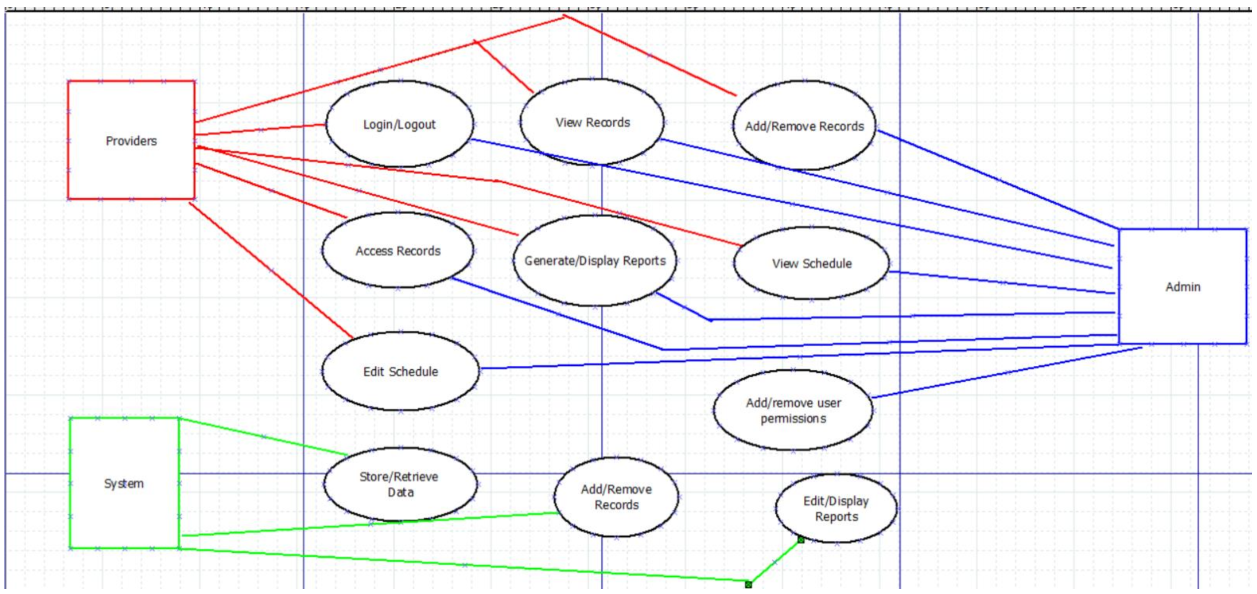
System-

1. Store data and allow it to be retrieved by approved users. (4)

2.Display reports and medical history. (6)

3. Ability to add/remove medical records when appropriate.

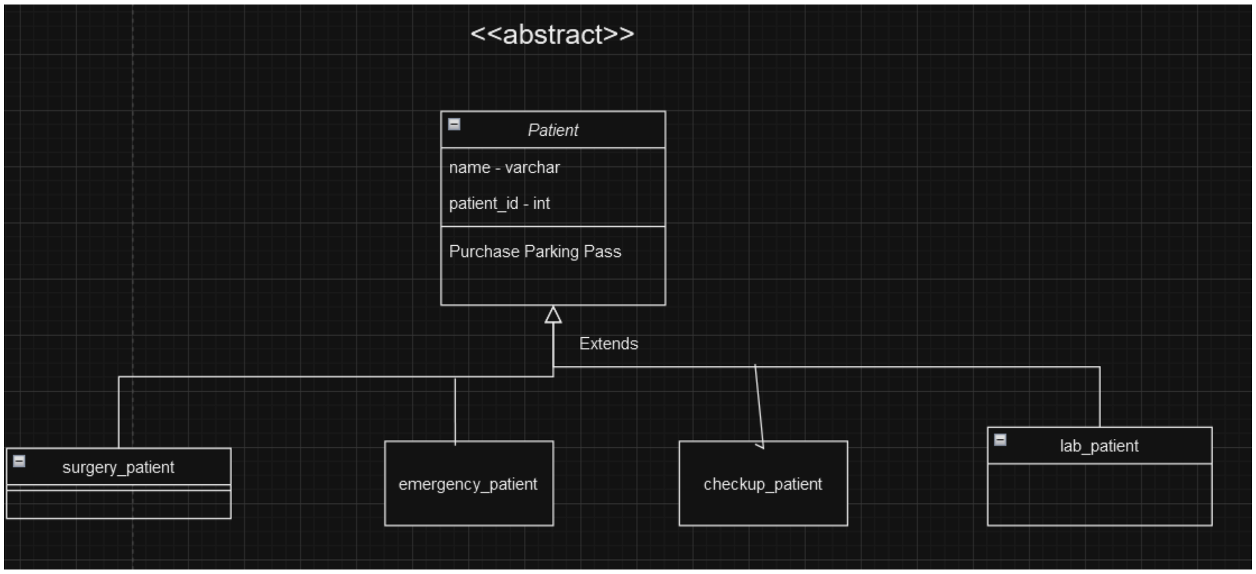
Use Case Diagram



Class Diagram

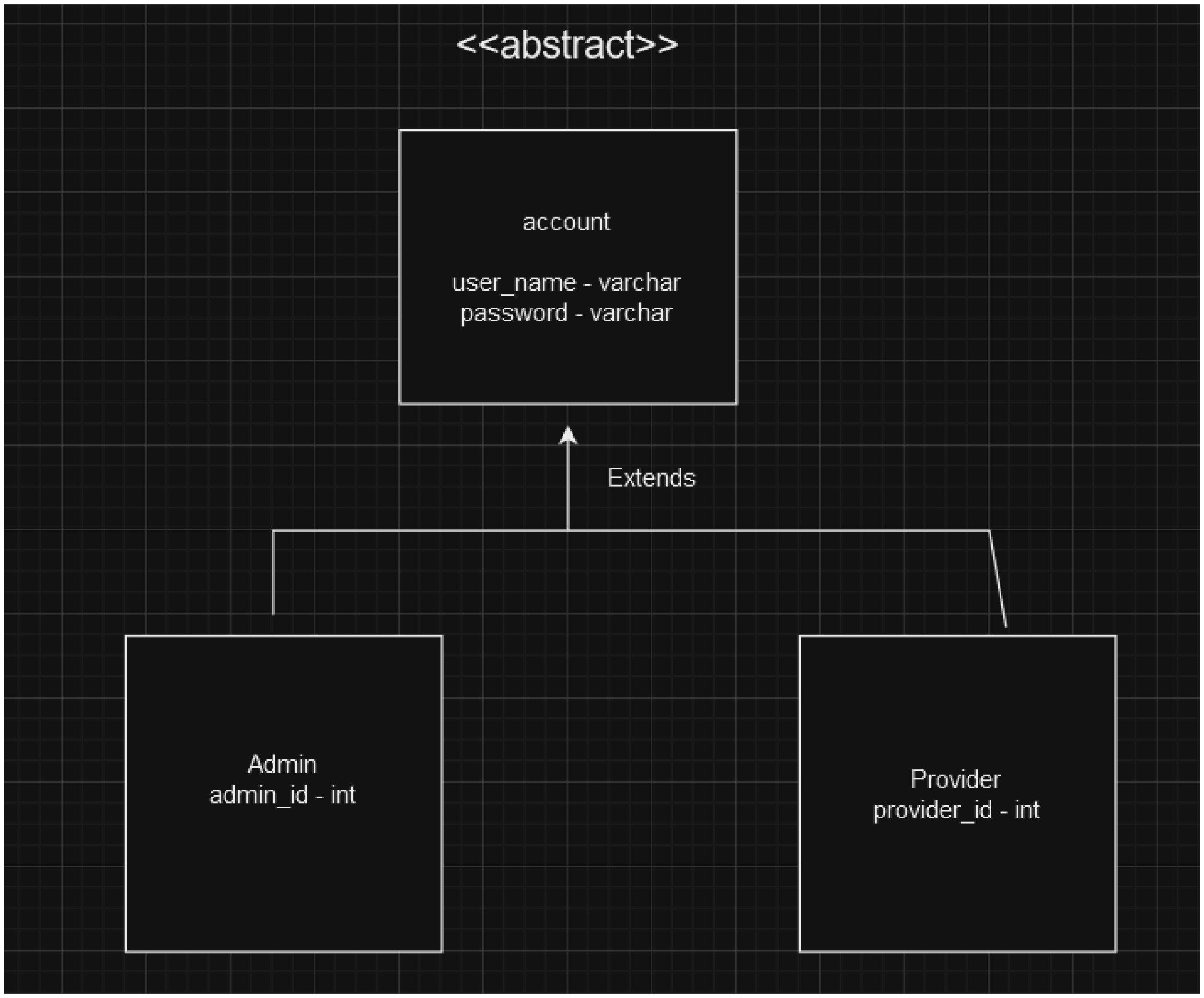
Patient Class

We have our patient class which is an abstract class. There will be four types of patients. lab_patient, checkup_patient, surgery_patient, and emergency_patient.



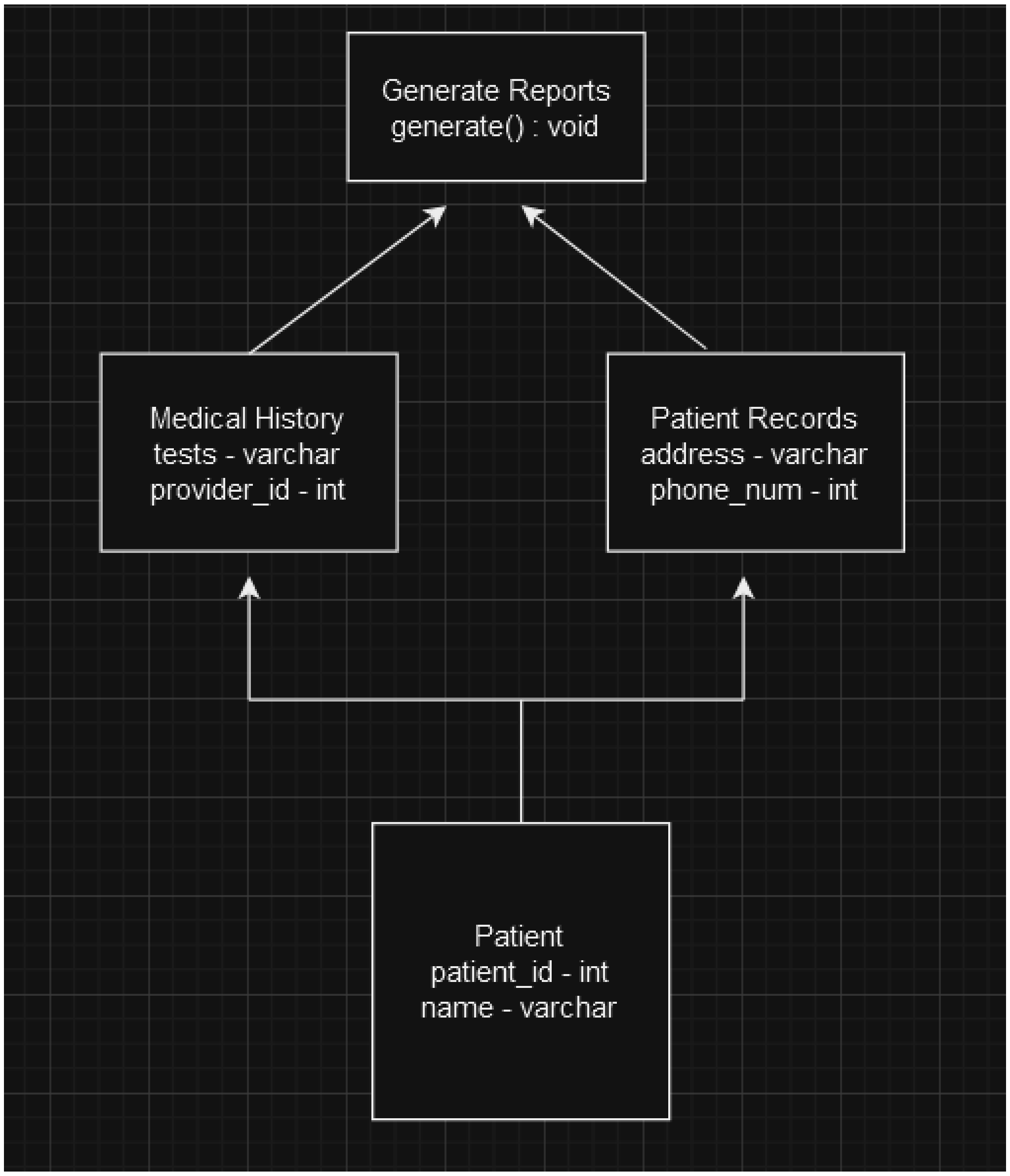
Account Class

There will be another abstract class names account. This class will create the accounts that can be used to view patient records. Any provider using the system must be approved by Admin before hand. Admin will be the only users able to add and remove access.


<<abstract>>
account
user_name - varchar
password - varchar
Extends
Admin
admin_id - int
Provider
provider_id - int

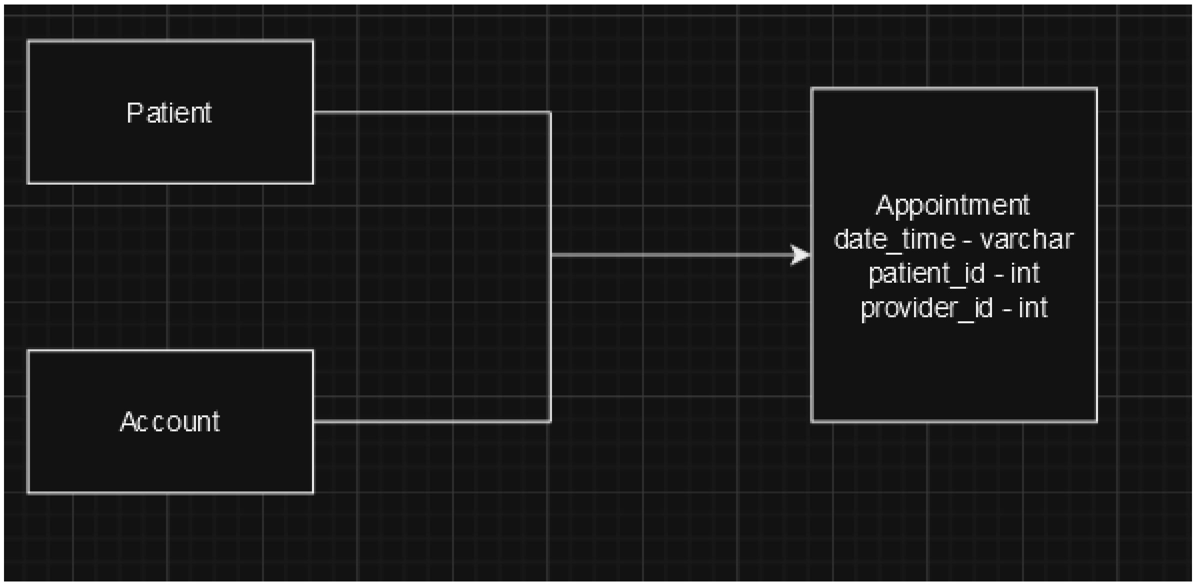
Records Class

We will then have a records class that will be able to be accessed by patients. This class will have medical records, this will include tests the patient has had and the provider that is their primary care provider. The class will also have patient records. This will have the patients address and phone number.

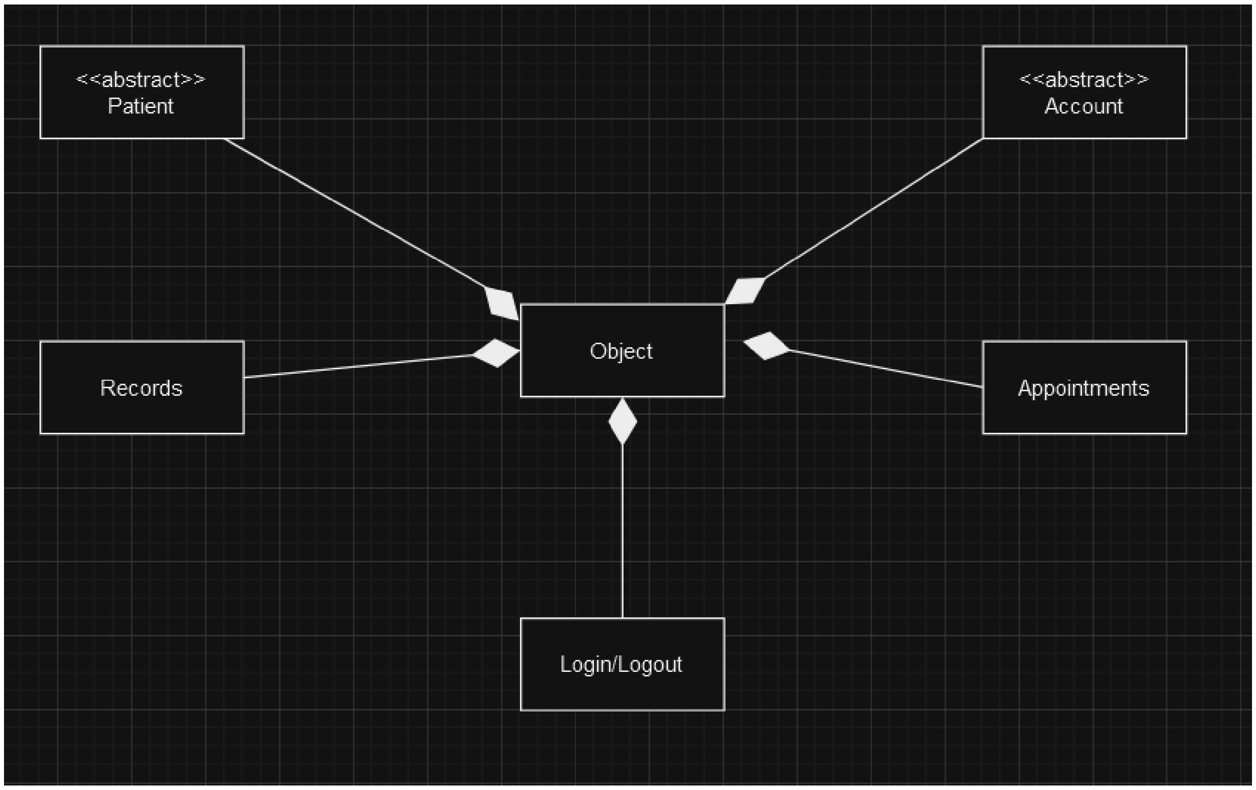


Appointment Class

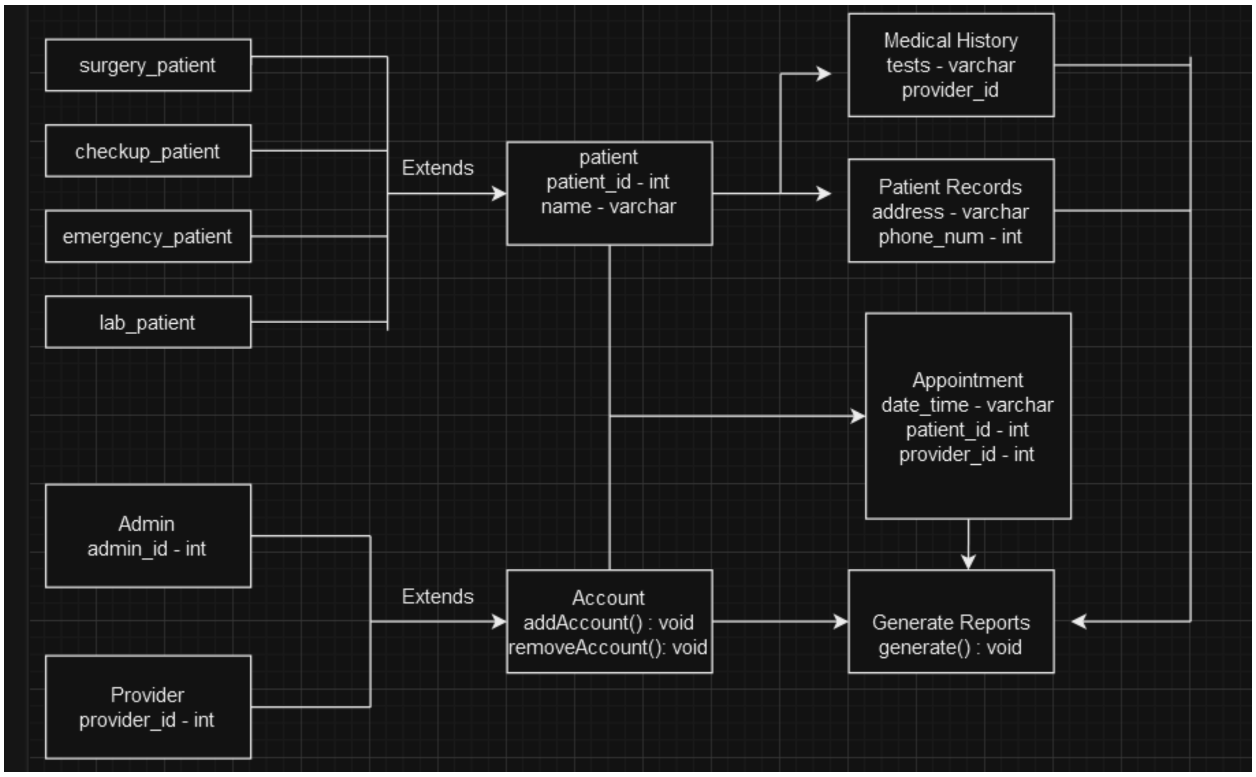
The appointment class will be how our providers schedule appointments between providers and patients. Admin will be able to access and edit these appointments.



Patient Management System Composition



System


surgery_patient
checkup_patient
emergency_patient
lab_patient
Extends
patient
patient_id - int
name - varchar
Medical History
tests - varchar
provider_id
Patient Records
address - varchar
phone_num - int
Appointment
date_time - varchar
patient_id - int
provider_id - int
Admin
admin_id - int
Provider
provider_id - int
Extends
Account
addAccount() : void
removeAccount(): void
Generate Reports
generate() : void

Title: System Sequence Diagram and Activity Diagram

Madeline Abbott and Benjamin Hogan

Sequence Diagrams

Use Case 1: Updating Patient Records

Actor:

User: The person who interacts with the system (ex. doctor, nurse, admin, etc.),

Object:

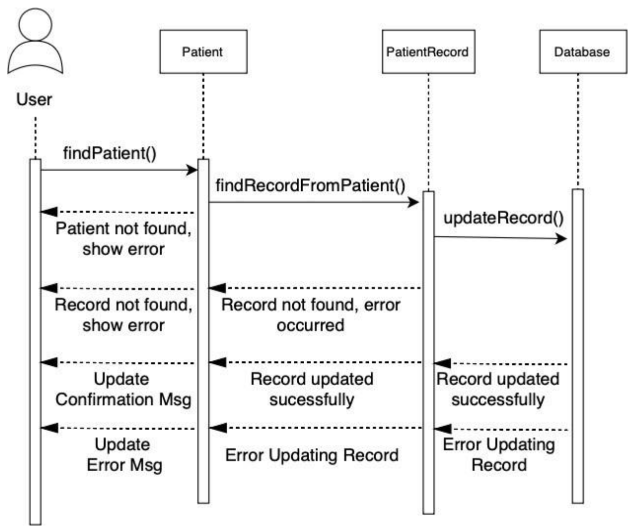
Patient: the patient whose records are being managed.

PatientRecord: the record of the patient.

Database: the storage system for patient records.

Steps of Updating a Patient Record:

1. The User opens the “Manage Patient Records” section in the interface.
2. The User identifies the Patient object based on the input of patientID.
3. The system sends the request to the PMS server and finds the corresponding PatientRecord.
4. The server returns a response finding the record (either a success or error) to the interface.
5. The system processes the request and communicates with the Database by either inserting, updating, or deleting a PatientRecord.
6. The server returns a response updating the record (either a success or error) to the interface.
7. The changes are saved to PatientRecords via the Database.



Use Case 2: Scheduling an Appointment

Actor:

User: The person who interacts with the system (ex. doctor, nurse, admin, etc.),

Object:

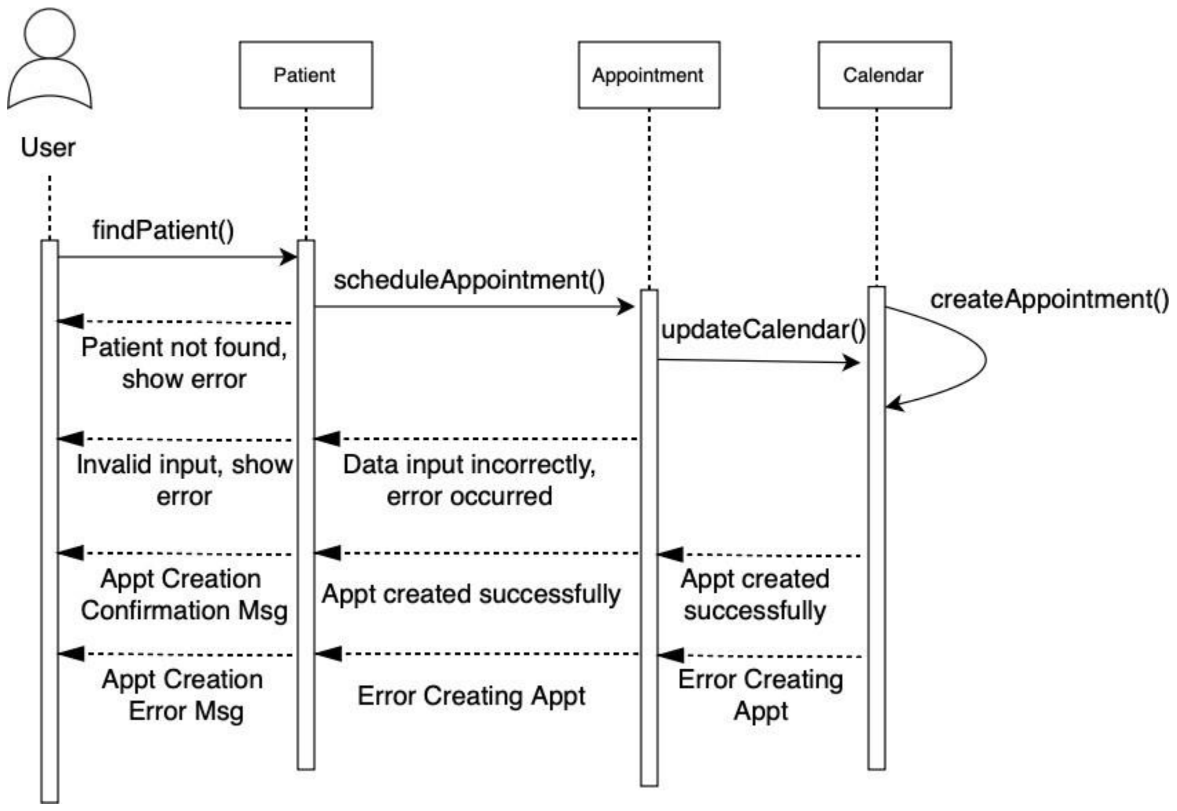
Patient: The person for whom the appointment is being scheduled.

Appointment: Represents the transaction of scheduling an appointment including the time, date, and reason.

Calendar: The tool used to represent dates and times, as well as schedule appointments.

Steps of Scheduling an Appointment:

- The User selects the Patient which will have an Appointment by using their patientID.
- The User selects a time and date for the Appointment.
- The system checks the Calendar to validate the time and date for the Appointment.
- The system either returns an error or confirmation message for the Appointment.
- The Appointment is saved in the Calendar.


User
Patient
Appointment
Calendar
findPatient()
scheduleAppointment()
updateCalendar()
createAppointment()
Patient not found, show error
Invalid input, show error
Data input incorrectly, error occurred
Appt Creation Confirmation Msg
Appt created successfully
Appt created successfully
Appt Creation Error Msg
Error Creating Appt
Error Creating Appt

Activity Diagrams

Use Case 1: Creating a Patient and Patient Record

States:

Initial State: The user opens the system and begins the patient creation process.

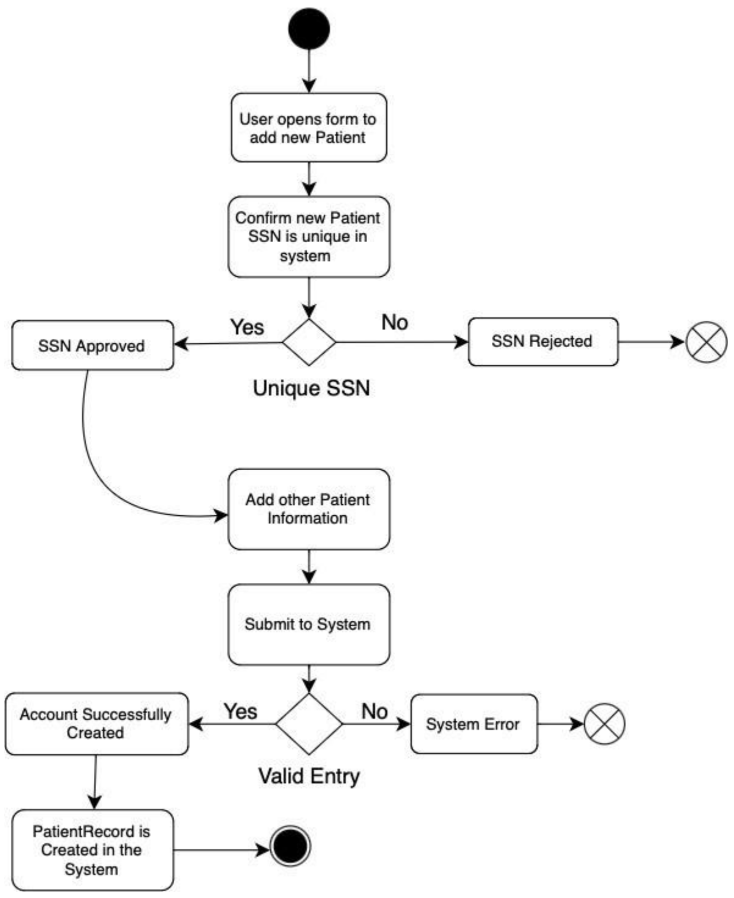
Final State: 1. The user receives confirmation of the new patient creation, and a new patient is added to the database. 2. The user receives an error message with an explanation, and a new patient is not added to the database.

Actions:

The user is logged in and selects the “Add New Patient” option. The user is given a form and inputs patient information including name, date of birth, SSN, address, contact information, active medications, known conditions, and other notes. User submits Patient details. System

validates the entered information. If the validation is successful, the system creates a

PatientRecord. System saves the new PatientRecord to the database. System returns a success or error message to the interface accordingly.


User opens form to add new Patient
Confirm new Patient SSN is unique in system
Yes
No
SSN Approved
SSN Rejected
Unique SSN
Add other Patient Information
Submit to System
Yes
No
Account Successfully Created
System Error
Valid Entry
PatientRecord is Created in the System

Use Case 2: Deleting a Patient and Patient Record

States:

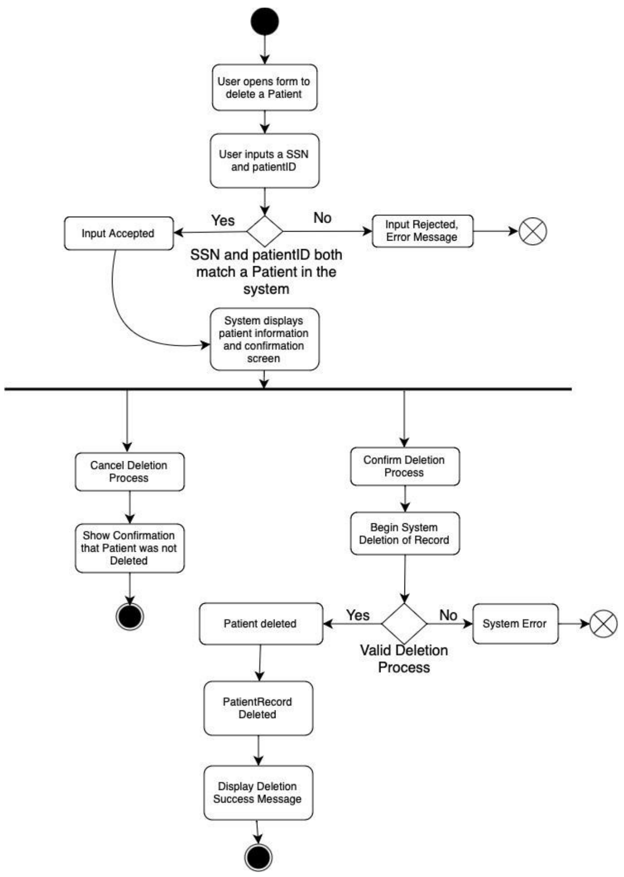
Initial State: The user opens the system and begins the patient deletion process.

Final State: 1. The user receives a confirmation message, and the patient and their record are successfully deleted. 2. The user receives an error message, and the patient and their record remain in the system. 3. The user cancels the deletion, and the patient and their record remain in the system.

Actions:

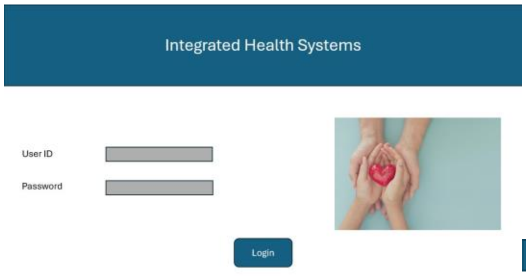
The user is logged in and selects the "Delete Patient Record" option. The user is given a form to confirm the patient they want to delete, where the user inputs the patient ID and their social security number. If both attributes do not match a patient, the system returns an error. Otherwise, the system displays the patient information along with a textbox that says "Are you sure you want to delete this patient's information?" along with two buttons, each saying "yes" or "no." If the user selects no, a confirmation message that the record was not deleted will be displayed. Otherwise, the system will delete the patient and their record. If the process is successful, the system will return a confirmation message of the deletion. Otherwise, the

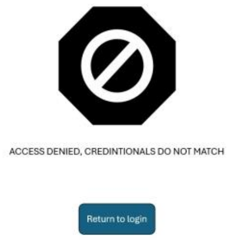
system will return an error message that assures that the patient and their record were not deleted.


User opens form to delete a Patient
User inputs a SSN and patientID
Yes
No
Input Accepted
Input Rejected, Error Message
SSN and patientID both match a Patient in the system
System displays patient information and confirmation screen
Cancel Deletion Process
Confirm Deletion Process
Show Confirmation that Patient was not Deleted
Begin System Deletion of Record
Patient deleted
Yes
No
System Error
Valid Deletion Process
PatientRecord Deleted
Display Deletion Success Message

User Interface Specification

Use Case – Non registered person attempts to log in

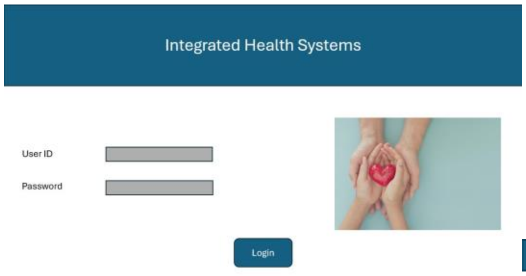


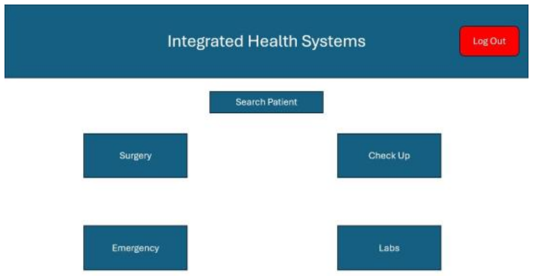


User Login Screen

User is shown access was denied with a button to return

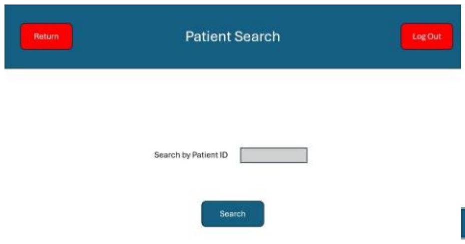
Use Case – User Search for Patients Chart

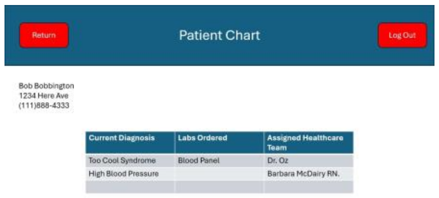




User Logs In

User gets to home screen, selects search patient





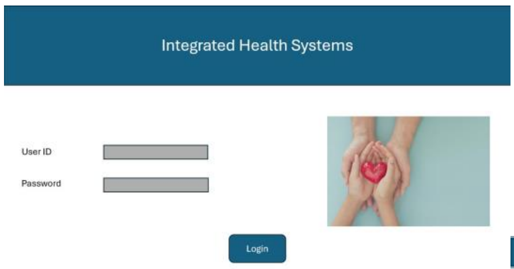
Bob Bobbington
1234 Here Ave
(111)888-4333

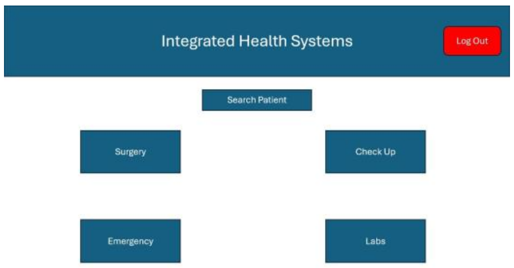
Current Diagnosis	Labs Ordered	Assigned Healthcare Team
Too Cool Syndrome	Blood Panel	Dr. Oz
High Blood Pressure		Barbara McDairy RN.

User enters patient ID

Patients Chart is pulled up

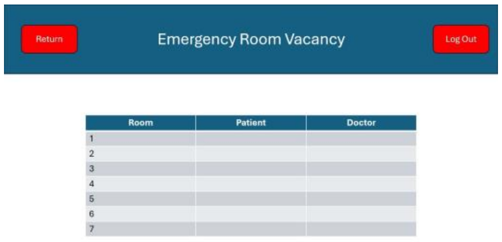
User Case – Assigning Patient to ER Room





User Logs in

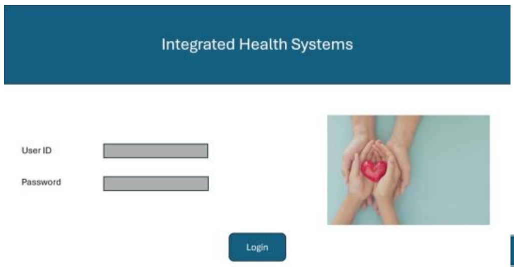
User selects emergency button

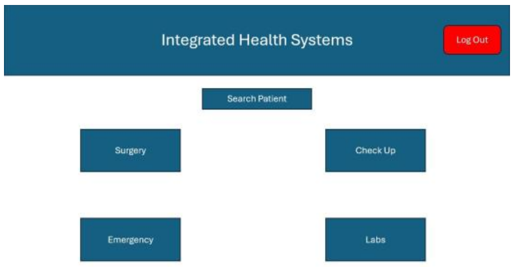


Room	Patient	Doctor
1		
2		
3		
4		
5		
6		
7		

User fills in data for emergency room table

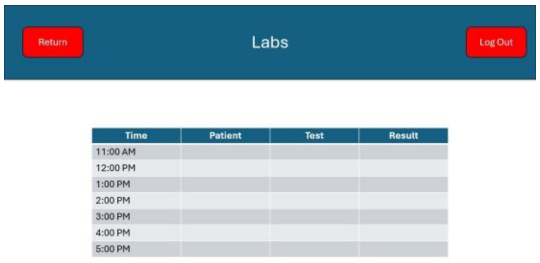
User Case – Adjusting Lab Schedule





User Logs in

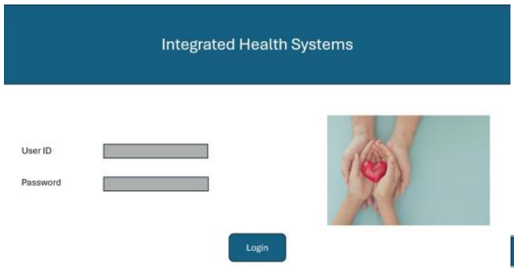
User selects labs button

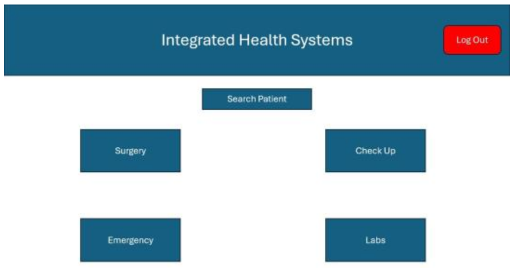


Time	Patient	Test	Result
11:00 AM			
12:00 PM			
1:00 PM			
2:00 PM			
3:00 PM			
4:00 PM			
5:00 PM			

User can make adjustments to labs schedule

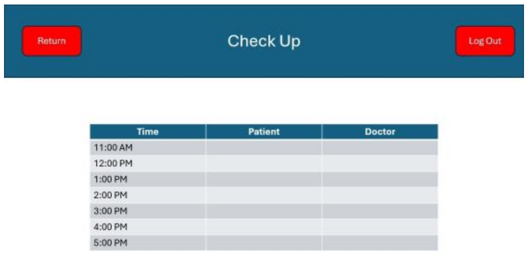
User Case – Adjusting Check Up Schedule





User Logs In

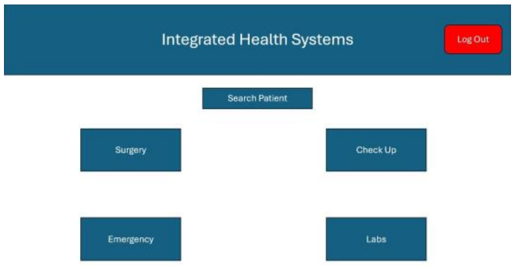
User selects check ups



Time	Patient	Doctor
11:00 AM		
12:00 PM		
1:00 PM		
2:00 PM		
3:00 PM		
4:00 PM		
5:00 PM		

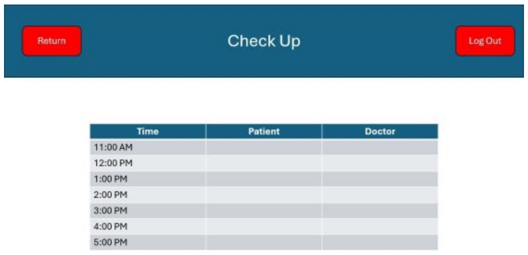
User adjusts check up schedule

User Case – Log Out



Each screen after log in will have a log out button at the top right

User Case – Return to prior page



Time	Patient	Doctor
11:00 AM		
12:00 PM		
1:00 PM		
2:00 PM		
3:00 PM		
4:00 PM		
5:00 PM		

Most pages will allow user to return to previous page with a button at top left.

User Effort Estimation

Usage Scenario	Navigation	Clicks	Keystrokes
Unauthorized User	Login screen, access denied	3	<50
User search for patient, user can enter/change patient information.	Login, home, patient search, patients chart	>=9	>100
Assigning patient to emergency room.	Login, home, emergency room	>=5	>50
Adjusting lab schedule	Login, home, labs	>=5	>50
Adjusting checkups schedule	Login, home, check ups	>=5	>50
Logout(User can log out from any page after login screen)	Login, logout button	>3	>50
Return to previous page(User can return to previous page after home screen)	Login, home, return any point past this	>4	>50

Patient Management System

Traceability Matrix

System Requirements

No.	Priority Weight (1-5: 1: lowest, 5: highest)	Description
REQ1	5	Health Care providers can log in and log out from the system
REQ2	5	Only Authorized providers able to login and access schedules/patient information
REQ3	5	Providers able to search for individual patients
REQ4	5	Only authorized providers able to edit patient schedules
REQ5	5	Only authorized providers are able to delete patient appointments.
REQ6	5	Only providers able to edit patient information
REQ7	4	Providers able to see up to date schedules as they are adjusted

REQ8	3	Providers able to adjust reasons for appointments

REQ9	3	Providers able to remove patients and their information

Use Cases

No.	Description
UC1	Login: Provider logs in to personal account
UC2	Schedule Appointment: Provider schedules an appointment
UC3	Search Patients: Provider able to search patients they have access to
UC4	Edit Patient Information: Provider able to edit information underneath the patients' records.
UC5	View Schedule: Provider able to view schedules that directly involve that provider.
UC6	Log Out: Provider logs out of account

Traceability Matrix

RQ	PW	UC1	UC2	UC3	UC4	UC5	UC6
RQ1	5	X	X	X	X	X	5
RQ2	5		X			X	
RQ3	5			X			
RQ4	5		X				
RQ5	5				X		
RQ6	5				X		
RQ7	4					X	
RQ8	3		X				
RQ9	3			X			
	Max PW	5	5	5	5	5	5

	Total PW	**5**	**18**	**13**	**15**	**14**	**5**

Title: System Architecture and System Design

Madeline Abbott and Benjamin Hogan

System Architecture and System Design **Architectural Styles**

Our project uses both a layered architecture and a client-server architecture.

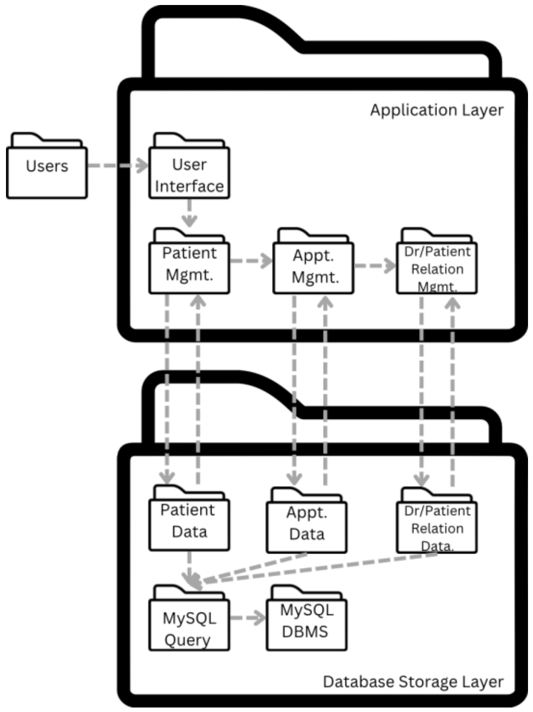
Layered Architecture

- Presentation Layer: The user interface, built with HTML, CSS, and JavaScript
- Business Logic Layer: PHP handling patient management operations (CRUD operations, scheduling, etc.)
- Data Access Layer: PHP interacting with the SQL database to store and retrieve data.

Client-server architecture

- The client is the web browser where users interact with the system.
- The server is the machine hosting the PHP scripts and SQL database.

UML Package Diagram



Mapping Subsystems to Hardware

Our system will run on multiple computers. On the client-side, web browsers on users' machines render the HTML/CSS and run JavaScript for interactive elements. On the server-side, a web server hosts the PHP scripts and connects to the locally hosted SQL database.

Persistent Data Storage

Our system requires persistent data to store patient records, appointments, and relationships with doctors. Persistent objects include things like Patients, Doctors, Admins, CheckUps, Labs, Surgeries, and DoctorPatient (the relationship between doctors and patients). **Network Protocol**

Our system uses HTTP/HTTPS as the communication protocol. The client sends HTTP requests to the server. Then, the server processes the requests using PHP and interacts with the database before returning responses.

Global Control Flow

Execution Orders

Our system is event driven, meaning that it persistently waits for events, and each user can complete actions in a unique sequence. Users can perform actions like login, patient record editing, or appointment creation. The system responds to these actions in any order, depending on user needs.

Time Dependency

Our system does not have any necessary real-time constraints because it processes actions on an event-response basis.

Concurrency

Since our system handles multiple users simultaneously, the web server and database handle concurrent requests internally. The PHP scripts don't require explicit threading since the web server manages concurrent connections.

Hardware Requirements

- **Client-Side**
    - A device with a modern web browser
    - Internet access to connect to the server

Server-Side

- A machine running a web server (ex. Apache) with PHP installed
- A local or remote SQL database server (ex. MySQL)
- Minimum 2 GB RAM and 10 GB disk space
- A dual-core processor

Weeks 1-2:

Set up a development environment (PHP, MySQL, HTML/CSS/JS).

Design database schema and set up MySQL database.

Weeks 3-4:

Develop the frontend interface with HTML, CSS, and JavaScript.

Implement basic PHP scripts for CRUD operations on patient records.

Weeks 5-7:

Develop the ability to schedule appointments.

Add features for managing medical histories and submitting reports.

Weeks 8-10:

Record midterm report.

Implement user authentication and authorization.

Ensure data security and integrity.

Weeks 11-13:

Test the system for bugs and usability issues, including adding in test cases.

Refine and optimize features based on feedback.

Week 14:

Prepare final documentation and presentation.

Week 15:

Record demo and finalize project for submission.