بسم الله الرحمن الرحيم

Palestine Polytechnic University

College of Information Technology &
Computer Engineering

Project name:

Mobile Hospital information system

(VitalIndex)

Team members:

Mohammad Hammad Omar Sweiti Eid Hamamda

Supervisor name:

Dr. Liana Tamimi

2025-2026

الإهداء

بعد سنواتٍ مُضينةٍ من التعب

ندرس الكتب، وننهلُ من معين العلم كأساً يرفع الله به درجاتٍ للذين اتخذوا العلم سبيلا ...

بوصيةِ المعلم الأول ... رسولنا الكريم سيد البشرية محمد بن عبد الله ...

نقف هنا ... لنهدي هذا الجهد الجليل لكلّ من استحقه

وبداية العرفانُ أصيلة لا تتحول

سندُ الأيامِ ومتكأي الدائم ... لوالدي العزيز

رفيقة الحياة وبلسمها الشافي ... والدتي الغالية

لرفاق أوقات القهوة والفراغ المليء بالأحاديث الشيّقة

للوطن الذي يستحق أن تنبيه سواعدٌ طموحه ...

للذين نزفوا دماءهم فداء للحرية ... شهدائنا الأبطال

للذين غيّبتهم زنازين العدو عن أهلهم وأحبتهم ... أسرانا الأشاوس ...

لكل من كان له فضلٌ علي ... علمني حرفاً وأخذ بيدي وأرشدني للنور ...

ولكل من يحب العلم ويسعى اليه.

Certification and Anti-Plagiarism Declaration

This is to declare that the graduation project produced under the supervision of Liana Tamimi having the title Mobile Hospital information system was prepared by student/s below in partial fulfilment of the requirements for the degree of Bachelor in Information Technology and no part hereof has been reproduced illegally (in particular: cut and paste) which can be considered as Plagiarism.

All referenced parts have been used to support and argue the idea and have been cited properly. I/We certify that I/we have not commit any plagiarism, cheating, or any other academic integrity violation. I/We will be responsible and liable for any consequence if violation of this declaration is proven.

Any use of Artificial Intelligence (AI) tools in the preparation of this project has been clearly declared and documented in the related appendix at the end of the report, in accordance with the PPU guidelines.

Date: 2 - 6 - 2025

Graduation project groups student(s):

Name: Mohammad Hammad **Signature:**

Name: Omar Sweiti **Signature:**

Name: Eid Hamamda **Signature:**

Abstract:

The Mobile Hospital Information System is a secure and scalable mobile application designed to streamline healthcare data management through role-based access to patient information. Developed collaboratively by the Colleges of Medicine and Information Technology, the system provides tailored interfaces for administrators, doctors, nurses, and students. It enhances clinical efficiency by digitizing medical documentation, facilitating real-time access to patient records, and supporting decision-making processes. Robust security features, including encryption and access control, ensure data confidentiality and integrity. This project bridges the gap between healthcare and technology, promoting improved patient care and operational efficiency in medical environments.

Table of Content

List of Table:

List of figures

Chapter 1
Introduction

1.1 Overview

The rapid advancement in technology and the digital transformation within the healthcare sector have created a pressing need for efficient and streamlined information management systems in medical institutions. In response, the Mobile hospital information system Project seeks to provide a comprehensive mobile-based hospital information system (HIS) that enhances the accessibility and organization of patient records.

Developed collaboratively by the College of Medicine and Health Sciences and the College of Information Technology and Computer Engineering,Mobile hospital information system simplifies administrative, medical, and documentation processes by centralizing patient data and assigning role-based access. This system empowers healthcare providers to access critical patient information with accuracy and efficiency, ultimately improving patient care quality.

The Mobile hospital information system Project will not only resolve the operational challenges for healthcare providers but also keep up the pace with the global trends of digital transformation in health care, giving rise to data-based solutions that revolutionize patient care. Using a strong mobile platform, the system allows healthcare professionals access to crucial patient data where and when they need it to improve decision-making order care faster especially in real-time situations. It also minimizes reliance on outdated paper-based systems, which are prone to loss, damage and errors in favor of a centralized, secure and scalable digital solution. The transition reduces operational efforts and simultaneously increases the quality and reliability of healthcare delivery.

1.2 Problem Statement for the Mobile hospital information system Project

Healthcare institutions face significant challenges in managing patient information due to reliance on outdated, paper-based systems and fragmented digital solutions. These inefficiencies result in delayed access to critical data, increased errors, and administrative burdens, ultimately affecting the quality of patient care.

1.3 motivation

Project Motivation

The inspiration behind this project lies in the pressing need to modernize healthcare data management and overcome inefficiencies in existing systems. Traditional methods of handling patient records, particularly paper-based systems, often lead to disorganization, slow access to information, and an increased risk of errors—issues that can significantly impact healthcare quality and safety.

1. **Challenges in Healthcare Efficiency**
   As hospitals continue to grow, managing vast amounts of medical data becomes increasingly complex. Administrative burdens can divert attention from patient care, highlighting the need for more efficient data handling methods.
2. **Concerns About Data Accuracy**
   Manual record-keeping is prone to mistakes, which can compromise patient safety and care outcomes. Ensuring accurate and timely documentation remains a challenge when relying on handwritten or fragmented systems.
3. **Evolving Demands in Medical Education**
   With greater emphasis on clinical exposure in modern medical training, educational institutions face the challenge of providing students with meaningful, real-world learning experiences—while still protecting patient privacy and adhering to ethical standards.
4. **Increasing Importance of Data Security**
   Protecting patient information has become a growing concern in the digital age. Healthcare institutions must address the need for secure access to sensitive data, ensuring that only authorized individuals can view and manage specific information.
5. **The Need for Immediate Access to Patient Data**
   In critical care situations, timely access to patient records can be life-saving. Current delays in retrieving important medical information can hinder effective decision-making and response time, underscoring the need for more immediate access to patient histories and treatment plans.

1.4 Importance of the Project and Digital Data in Healthcare

The Mobile Hospital Information System Project plays a pivotal role in modernizing healthcare operations by centralizing and streamlining data management, which

reduces redundant administrative tasks and allows healthcare professionals to devote more attention to direct patient care. Tailored access for each user role enhances coordination and resource allocation across departments. Doctors and nurses benefit from real-time access to patient information, administrators maintain data consistency, and students gain educational exposure without compromising data security. Integration with existing systems supports the creation of unified patient records and reliable access to healthcare data across institutions.

The adoption of digital data management in hospitals and academic institutions like the College of Medicine and Health Sciences has significantly improved operational efficiency and enhanced the support for medical education. Digitized records increase data accessibility and accuracy, reduce human error, and ensure consistent documentation of medical history, diagnoses, and treatment plans. This contributes to more informed and timely medical decisions.

In academic settings, secure access to anonymized real-world medical cases helps students observe professional practices in data handling and patient confidentiality. The transition to digital systems reflects a broader alignment with global healthcare standards, encouraging better collaboration, research, and outcomes across the medical field.

1.5 Project Objectives

The objectives of the Mobile hospital information system Project are as follows:

1. **Enhance Patient Care Quality:**
   Provide healthcare providers with instant access to accurate and up-to-date patient records, enabling doctors and nurses to make timely and informed medical decisions, especially in critical care situations.
2. **Improve Data Management Efficiency:**
   Centralize patient data in a secure, role-based system to reduce administrative overhead and allow healthcare staff to focus more on patient care rather than manual record-keeping tasks.
3. **Facilitate Interoperability:**
   Ensure seamless integration with existing hospital information systems, allowing doctors and nurses to access unified data across departments for better collaboration and coordination of care.
4. **Support Medical Education:**
   Offer controlled access to real-world patient data for students, providing them with practical learning experiences in a secure and educational environment that bridges the gap between classroom theory and real-world applications.

5. **Leverage Digital Transformation:**
   Replace traditional paper-based processes with a mobile-based digital platform, reducing the workload for nurses and doctors and improving their ability to document and retrieve data in real-time.

1.6 Scope of project

The Mobile hospital information system Project is designed to serve the following users within healthcare institutions:

- **Doctors:** To access and update patient records for accurate diagnosis and effective treatment planning.
- **Nurses:** To monitor, document, and manage patient vitals and treatment progress in real-time.
- **Students:** To access anonymized patient data for practical learning and medical education purposes

1.7 Proposed Solution

Why Mobile?

We chose to make our project mobile because mobile devices are the ideal replacement for traditional paperwork in today's fast-paced, technology-driven world. Digital mobile platforms like Mobile hospital information system streamline processes by offering real-time access to critical data and addressing the inconveniences of paperwork, such as its bulkiness, susceptibility to damage, difficulty in sharing, and inefficiency in updating. With the ability to centralize and secure information, mobile solutions provide unparalleled convenience, allowing healthcare professionals to access and update patient records anytime and anywhere. This approach not only enhances efficiency but also aligns with the global trend toward digital transformation, making mobile technology an indispensable tool for modern healthcare management.

The Mobile hospital information system Project serves as an alternative to traditional Specifically, it offers **alternatives to:**

- **Paper-Based Record-Keeping.**

Old-fashioned paper records are bulky, easy to lose or damage, and not the best way to get quick access and updates. Mobile hospital information system is a computerized solution that stores centralized data, offers real-time access, and allows instant patient data updates.

- **Use of websites.**

The phone application is an alternative to websites because it allows quick access to the program and fashion information and because it does not require the Internet to access the existing information.

- **An alternative to using a computer:**

An example of this is AviCenna is the current hospital information system (HIS) used in Palestine. While it has served as a reliable platform for managing patient data, it primarily operates as a desktop-based system, limiting accessibility for healthcare providers on the go. **The Mobile hospital information system Project** offers a mobile-based solution, providing healthcare professionals with real-time access to patient information anytime and anywhere. This mobile capability enhances the flexibility and efficiency of patient care, ensuring that vital data is always at the fingertips of doctors, nurses, and other healthcare providers.

1.8 Timeline/ Project Scheduling

Tasks table:

Task number	Task name	The time required(week)
1	System definition and planning	4
2	Determine the project requirement and data	3
3	Description of the project requirement	4
4	System design	5
5	System development and programming	6
6	Integration and system testing	7
7	Documenting the application	During the working period

Table 1.8.1: Tasks table

Timeline figure:

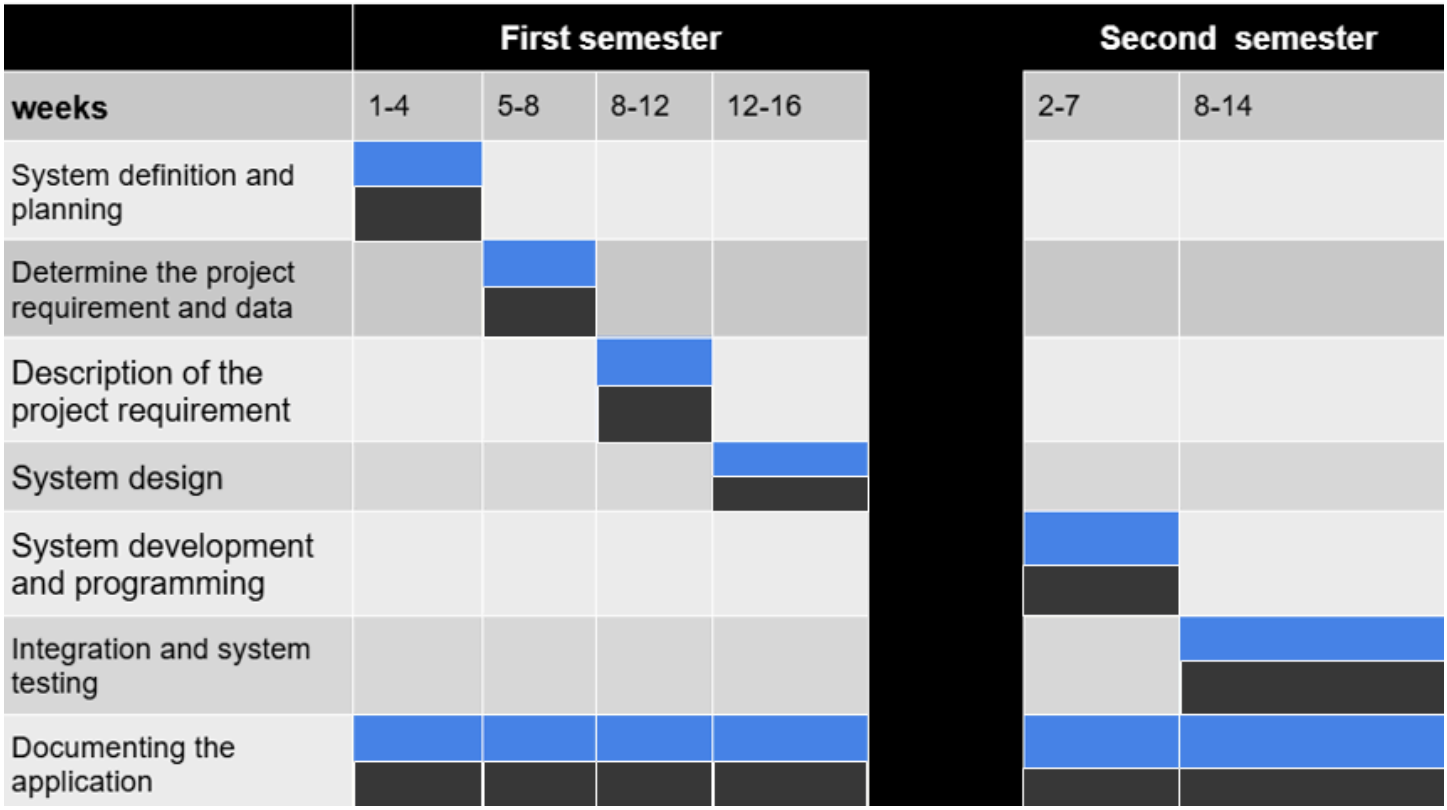


Figure 1.8.1: Timeline figure and chart

Chapter 2
Requirements Specification

2.1 Overview

This chapter outlines the requirements specification for the HIS system, detailing functional and non-functional needs to ensure efficient patient data management and seamless integration with hospital workflows.

2.2 Context Diagram

The context diagram shows the interactions between a system and the actors or between the system and other software systems as shown in figure.

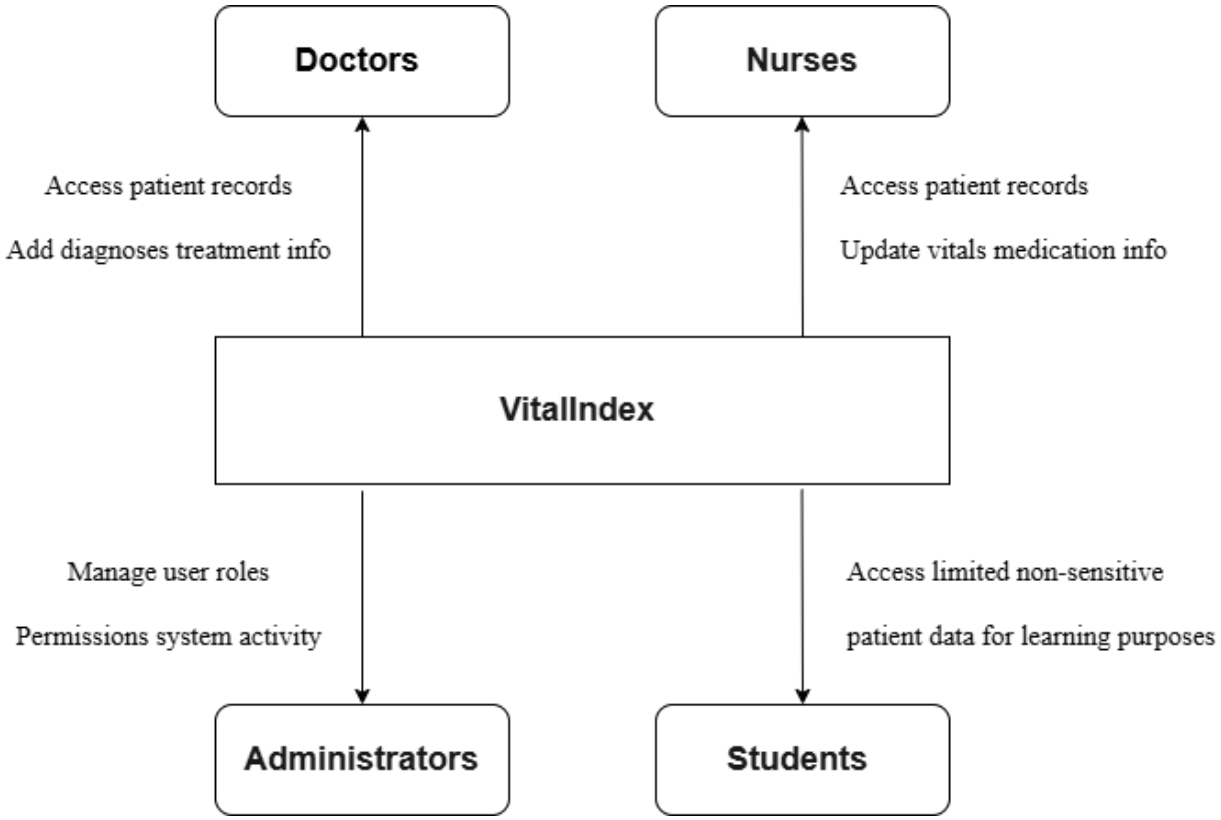


Figure 2.2.1: Context diagram

2.3 Functional requirements

The objective of these requirements is to define the functional aspects of the system, how it interacts with inputs and outputs, and how it behaves in certain cases. It includes the requirements of the system and user.

For Administrators:

1. **Create Actor:**
    - Administrators must be able to create new actors by entering personal and role-specific details. Validation should ensure data integrity, and all required fields must be filled before creation.
2. **Edit Actor:**
    - Administrators must be able to update existing actor profiles by modifying their personal and role-specific details. Validation should ensure data integrity, and changes must not affect active records improperly.
3. **Delete Actor:**
    - Administrators must be able to delete actor profiles with confirmation. Deletion must not proceed if the actor is linked to active records.
4. **Generate System Activity Reports:**
    - Administrators must be able to generate system activity reports by selecting specific parameters, such as date range or user type. These reports should provide insights into user actions and system performance.
5. **Traceable Access Logs:**
    - Administrators must have access to activity logs that record all system usage and actions. These logs should be filterable by criteria such as user or date, enabling detailed auditing and accountability.
6. **Assign Roles to Users:**
    - Administrators must be able to assign roles to users, defining access and permissions according to their responsibilities within the system.
7. **Modify User Roles:**
    - Administrators must be able to update or modify existing roles and permissions assigned to users. Only authorized personnel should be able to make these changes, and configurations must remain valid.

For Doctors:

1. **Create Patient:**
    - Doctors must be able to register new patients by entering required demographic and contact details. The system must validate all fields for completeness and prevent duplicates.
2. **Update Patient Basic Info:**
    - Doctors must be able to update basic patient information, such as demographics and contact details. The system must ensure data integrity and prevent unauthorized changes.
3. **Create Medical Record:**
    - Doctors must be able to create new medical records for patients, entering diagnoses, treatments, and other relevant clinical data. The system must validate entries for completeness and accuracy.
4. **Update Medical Record:**
    - Doctors must be able to update existing medical records, including diagnoses and treatment plans. All updates must be validated for completeness and accuracy.
5. **Create Medical Report:**
    - Doctors must be able to generate comprehensive medical reports summarizing patient data for clinical and administrative use. The system should flag incomplete data and allow corrections before report generation.
6. **Update Medical Report**:
    - Doctors must be able to update existing medical reports as needed, with validation to ensure accuracy and completeness.
7. **Access Full Patient Records:**
    - Doctors must be able to access complete patient records to support medical decision-making. Access must be restricted to authorized users based on their roles.

For Nurses:

1. **Create Patient:**
    - Nurses must be able to register new patients by entering required demographic and contact details. The system must validate all fields for completeness and prevent duplicates.
2. **Update Patient Basic Info:**
    - Nurses must be able to update basic patient information, such as demographics and contact details. The system must ensure data integrity and prevent unauthorized changes.
3. **Create Medical Record:**
    - Nurses must be able to create new medical records for patients, entering diagnoses, treatments, and other relevant clinical data. The system must validate entries for completeness and accuracy.
4. **Update Medical Record:**
    - Nurses must be able to update existing medical records, including diagnoses and treatment plans. All updates must be validated for completeness and accuracy.
5. **Access Full Patient Records:**
    - Nurses must be able to access complete patient records to support medical decision-making. Access must be restricted to authorized users based on their roles.

For Students:

1. **View Available Case Studies:**
    - Students must be able to view a list of available case studies for educational purposes. All data must be anonymized to protect patient privacy.
2. **Save Case Studies:**
    - Students must be able to save selected case studies for later review or download, in compliance with privacy and access policies.

2.4 Non-functional Requirements:

Non-Functional Requirements define a system's quality attributes, operations, and constraints rather than its behaviors. They ensure the system performs efficiently, securely, and reliably under various conditions while meeting user and legal expectations for sustainability and usability.

1. Performance

Performance refers to the system’s ability to execute its functions swiftly and handle increasing workloads effectively.

- The system should be capable of performing the usual functions of allowing access to patient files, retrieving and modifying the data .
- The system should be capable of efficient management of large amounts of patient data, providing an acceptable load time of information strings as the data grows.

2. Reliability and Availability

Reliability is the system’s ability to consistently perform without failure, while availability measures its uptime and accessibility during critical moments.

- The system shall ensure 99.9 % availability so as to be accessible for healthcare providers and healthcare administrators during critical times.
- In case of one part of the network or the system going down, the other part should be able to maintain the system by including a failover mechanism.
- To avoid any possibility of data corruption, there should be regular automated backups at least more than once a day, in addition to incrementals done once daily .

3. Security
Security involves protecting sensitive data from unauthorized access, ensuring confidentiality, integrity, and compliance with legal standards.

- In order to protect data and private information, the system has to adhere to appropriate legal provisions in the healthcare industry.
- Appropriate practices such as encryption of sensitive data during transport and storage to augment patient confidentiality.
- The system will require the use of an additional factor to authenticate the user when logging into the system so as to secure the account and stop any unauthorized access.

4. Scalability
Scalability is the system's ability to handle increased demands, such as a larger user base or growing data volumes, without requiring significant re-engineering.

- This system is designed to be scalable such that the user base, the number of hospital subunits and the quantity of data can increase without major re-engineering the system.
- Such growth shall be in the integration of more healthcare information systems as may be required and ensure that the system grows with the expanding institutions.

5. Maintainability
Maintainability refers to the ease with which the system can be updated, modified, or enhanced to adapt to new requirements or resolve issues.

- The System shall be designed with separation of functions or parts such that there shall be minimal re-engineering of the entire system to address change in programming, additional features and upgrade of any part of the system.
- Such changes shall be reflected in the relevant documentation and will provide appropriate information about the system including its features, maintenance activities, control of the user, and updates of the software.

6. Privacy

Privacy refers to the protection of sensitive information for all users, including patients, administrators, doctors, nurses, and students, ensuring confidentiality, security, and compliance with legal standards.

- The system shall anonymize patient data for educational purposes, ensuring that no identifiable information is accessible during academic use while protecting other user data from unauthorized access.
- Role-based access control shall be implemented to restrict access to information based on the user's role, ensuring that users can only view and manage data relevant to their responsibilities.
- Encryption shall be applied to all sensitive data, including personal and professional information of users, during storage and transmission to prevent unauthorized access or breaches.

2.5 Use case

The use case describes the user's view of the application.

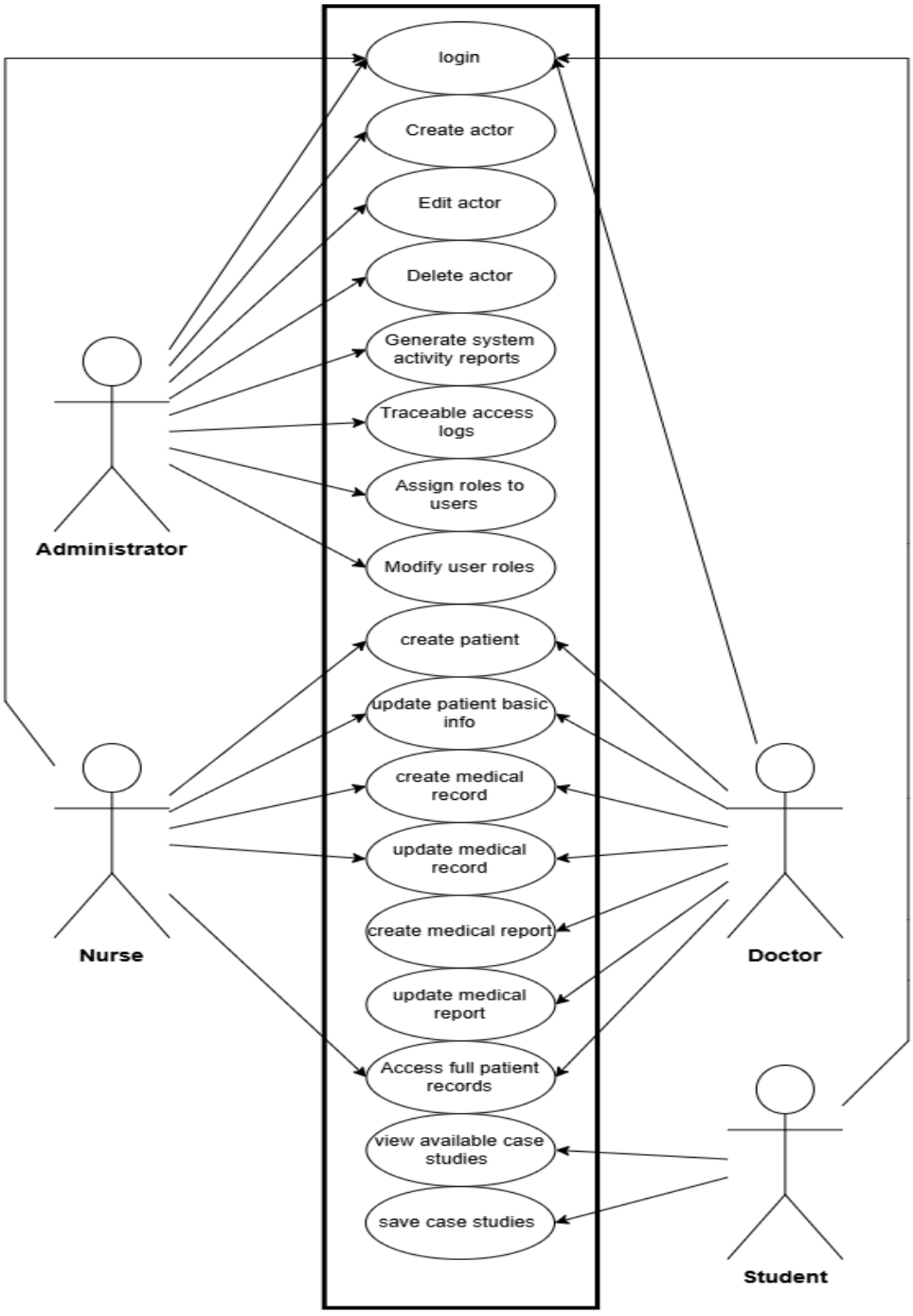


Figure 2.5.1: Use case diagram

2.6 Use Case Description

A use case is a written description of how users will perform tasks on your mobile app.

- **For All Actors**

Use case	Login
Actor	Administrator, Nurse, Doctor, Student
Goal	Allow users to securely access the system
Preconditions	User must have valid credentials
Scenario	1. User enters username and password. 2. The system authenticates credentials. 3. If valid, the user is granted access.
Exceptions	Invalid credentials result in access denied and an error message.

Table 2.6.1: Login

- **For Administrators**

Use case	Create Actor
Actor	Administrator
Goal	Allows the Administrator to create a new actor by entering personal and role-specific details
Preconditions	Administrators must be logged in and have permissions to create new actors.
Scenario	4. Admin navigates to the "Create Actor" page. 5. Fills out actor details. 6. Clicks submit. 7. System validates the data and saves the actor.
Exceptions	If the system fails to store the data due to an error, it displays an appropriate error message.

Table 2.6.2: Create Actor

Use case	Edit Actor
Actor	Administrator
Goal	Allows the Administrator to modify details of existing actors.
Preconditions	Administrator must be logged in and have permissions to edit actor profiles.
Scenario	1. Admin selects an actor. 2. Modifies the actor's details. 3. Clicks submit. 4. The system validates the data and updates the actor.
Exceptions	If the actor does not exist or is linked to active records that should not be modified, the system shows an error.

Table 2.6.3: Edit Actor

Use case	Delete Actor
Actor	Administrator
Goal	Allows the Administrator to delete an actor profile, ensuring they are not linked to any active records.
Preconditions	Administrator must be logged in with appropriate permissions. The actor must not be associated with active records.
Scenario	1. Admin selects an actor. 2. Confirms deletion. 3. The system validates and deletes the actor if conditions are met.
Exceptions	If an actor cannot be deleted, an appropriate error message is shown.

Table 2.6.4: Delete Actor

Use case	Generate System Activity Reports
Actor	Administrator
Goal	Allows the Administrator to generate reports based on system activity, filtered by parameters like date range or user type.
Preconditions	Administrator must be logged in with reporting permissions.
Scenario	1. Admin selects report type. 2. Set parameters . 3. Submit the request. 4. The system generates and displays the report.
Exceptions	If report generation fails due to system issues, an error message is shown.

Table 2.6.5: Generate System Activity Reports

Use case	Traceable Access Logs
Actor	Administrator
Goal	Allows the Administrator to access system logs detailing all user actions, filtered by user or date.
Preconditions	The system must have active logging enabled. Administrator must be logged in with permissions to view access logs.
Scenario	1. Administrator accesses the "Audit Logs" section. 2. Admin selects log criteria. 3. System displays all activities based on search filters. 4. Administrator reviews logs for suspicious or erroneous activities.
Exceptions	If the logs are corrupted or cannot be retrieved, an error message is shown.

Table 2.6.6: Traceable Access Logs

Use case	Assign Roles to Users
Actor	Administrator
Goal	Allows the Administrator to assign roles to users, determining their access and permissions.
Preconditions	Administrator must be logged in with permission to assign roles.
Scenario	1. Admin selects a user. 2. Assigns a role and sets permissions. 3. Submits the changes. 4. The system updates the user's role.
Exceptions	If the user does not exist, an error is displayed.

Table 2.6.7: Assign roles to users

Use case	Modify Users Roles
Actor	Administrator
Goal	Allows the Administrator to modify existing user roles and permissions.
Preconditions	Administrator must be logged in and authorized to modify roles.
Scenario	1. Admin selects a user. 2. Modifies the user's role. 3. Clicks submit. 4. The system updates the role.
Exceptions	If the user does not exist, an error is displayed.

Table 2.6.8: Modify users roles

- **For Doctors**

Use case	Create Patient
Actor	Doctor, Nurse
Goal	Register a new patient in the system
Preconditions	User must be logged in with permission
Scenario	1. User navigate to add new patient screen. 2. Enter patient details. 3. Submit. 4. System validates and saves.
Exceptions	Data invalid or duplicate; error message shown.

Table 2.6.9: Create patient

Use case	Update Patient Basic Info
Actor	Doctor, Nurse
Goal	Update non-medical patient information
Preconditions	Patient exists and user must be logged in with permission
Scenario	1. User select a patient. 2. System navigate to modify patient screen. 3. Edit patient details. 4. Submit. 5. System updates info.
Exceptions	Data invalid or duplicate; error message shown.

Table 2.6.10: Update patient basic info

Use case	Create Medical Record
Actor	Doctor, Nurse
Goal	Create a new medical record for a patient
Preconditions	Patient exists and user must be logged in with permission
Scenario	1. User select a patient. 2. System navigate to modify patient screen. 3. User select add new medical record. 4. Enter medical record data. 5. Submit. 6. System saves record.
Exceptions	Data invalid ; error message shown.

Table 2.6.11: Create medical record

Use case	Update Medical Record
Actor	Doctor, Nurse
Goal	Update existing medical record.
Preconditions	Medical record exists, actor logged in with permission.
Scenario	1. User select a patient. 2. System navigate to modify patient screen. 3. User select a medical record. 4. System navigate to modify medical record screen. 5. Edit medical record data. 6. Submit. 7. System updates record.
Exceptions	Data invalid ; error message shown.

Table 2.6.12: Update medical record

Use case	Create Medical Report
Actor	Doctor
Goal	Generate a medical report for a patient.
Preconditions	Doctor must be logged in with permissions to generate reports.
Scenario	1. User select a patient. 2. System navigate to modify patient screen. 3. User select create medical report. 4. Enter medical report data. 5. Submit. 6. System saves report.
Exceptions	Data invalid ; error message shown.

Table 2.6.13: Create medical report

Use case	Update Medical Report
Actor	Doctor
Goal	Update an existing medical report.
Preconditions	Doctor must be logged in with permissions to generate reports.
Scenario	1. Doctor navigate to reports screen. 2. Doctor select report. 3. Doctor select update report. 4. System navigate to update medical report screen 5. Edit medical report data. 6. Submit. 7. System updates report.
Exceptions	Data invalid ; error message shown.

Table 2.6.14: Update medical report

- **For Students**

Use case	View Available Case Studies
Actor	Students
Goal	Allow students to view case studies for educational purposes.
Preconditions	Student logged in with permission
Scenario	1. Student navigate to case studies screen. 2. Select case study. 3. System navigate to view case study screen.
Exceptions	If the student is unauthorized, access is denied.

Table 2.6.15: View available case studies

Use case	Save Case Study
Actor	Students
Goal	Save case studies for later review.
Preconditions	Student logged in with permission.
Scenario	1. Student navigate to case studies screen. 2. Select case study. 3. System navigate to view case study screen. 4. Student select save case study. 5. System add case to saved case studies.
Exceptions	If the student is unauthorized, access is denied.

Table 2.6.16: Save case study

Chapter 3
System design

3.1 Overview

This chapter includes an explanation of the project's design, tools, and structure, as well as a detailed description of the system's components to provide a comprehensive understanding of the entire system.

3.2 Architecture Design

Our project is organized into two main architectural designs: one for the frontend and one for the backend. Each follows a layered approach, ensuring clear separation of concerns, modularity, and ease of maintenance.

3.2.1 Frontend Architecture

The frontend is structured in multiple layers, as shown in the diagram below:

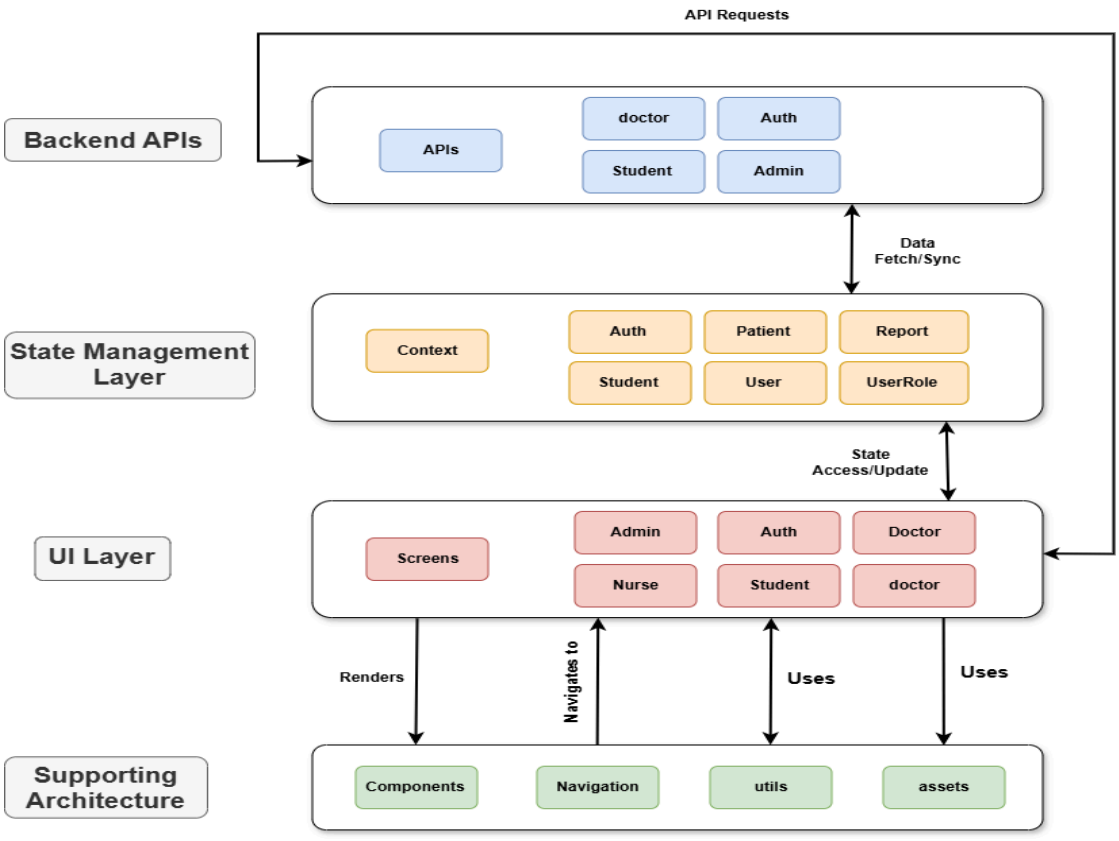


Figure 3.2.1: Frontend Architecture

Layers and Components

- **Backend APIs:**
  The frontend communicates with backend APIs to fetch and synchronize data. These APIs expose endpoints for different user roles (Admin, Doctor, Nurse, Student) and handle authentication and data management.
- **State Management Layer:**
  This layer manages the application's state using contexts for authentication, user roles, patient data, reports, and more. It ensures that data is accessed and updated consistently across the UI.
- **UI Layer:**
  The UI layer consists of screens and modules for each user type (Admin, Doctor, Nurse, Student, Auth). It renders the user interface and interacts with the state management layer to display and update data.
- **Supporting Architecture:**
  This includes reusable components, navigation logic, utility functions, and assets. These elements support the UI and provide shared functionality across the application.

Key Advantages

- **Separation of Concerns:** Each layer has a specific responsibility, making the codebase easier to maintain and extend.
- **Reusability:** Shared components and utilities can be used across different parts of the application.
- **Scalability:** The layered approach allows for easy addition of new features and user roles.

3.2.2 Backend Architecture

The backend is designed using a layered architecture based on Django REST Framework, as illustrated below:

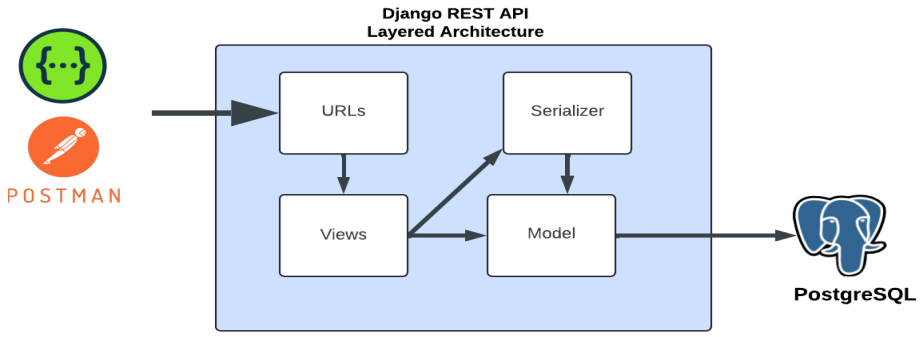


Figure 3.2.2: Backend Architecture

Layers and Components

- **API Interface:**
  Tools like Swagger and Postman are used to interact with and test the API endpoints.
- **URLs:**
  Define the routing of HTTP requests to the appropriate views.
- **Views:**
  Handle incoming requests, invoke business logic, and return responses. They act as controllers in the MVC pattern.
- **Serializers:**
  Responsible for transforming data between Django models and JSON format, ensuring validation and consistency.
- **Models:**
  Define the structure of the database tables and encapsulate the business data.
- **Database (PostgreSQL):**
  Stores all persistent data for the application.

Key Advantages

- **Modularity:** Each layer handles a distinct part of the request/response cycle, making the system easier to test and maintain.
- **Data Integrity:** Serializers validate data before it is saved to the database.
- **Scalability:** The architecture supports adding new endpoints and logic with minimal impact on existing code.

3.3 Conceptual Schema of the Database

We have eleven entities in our site, each of which has a number of attributes, as shown in Figure 3.2.1 and 3.2.2

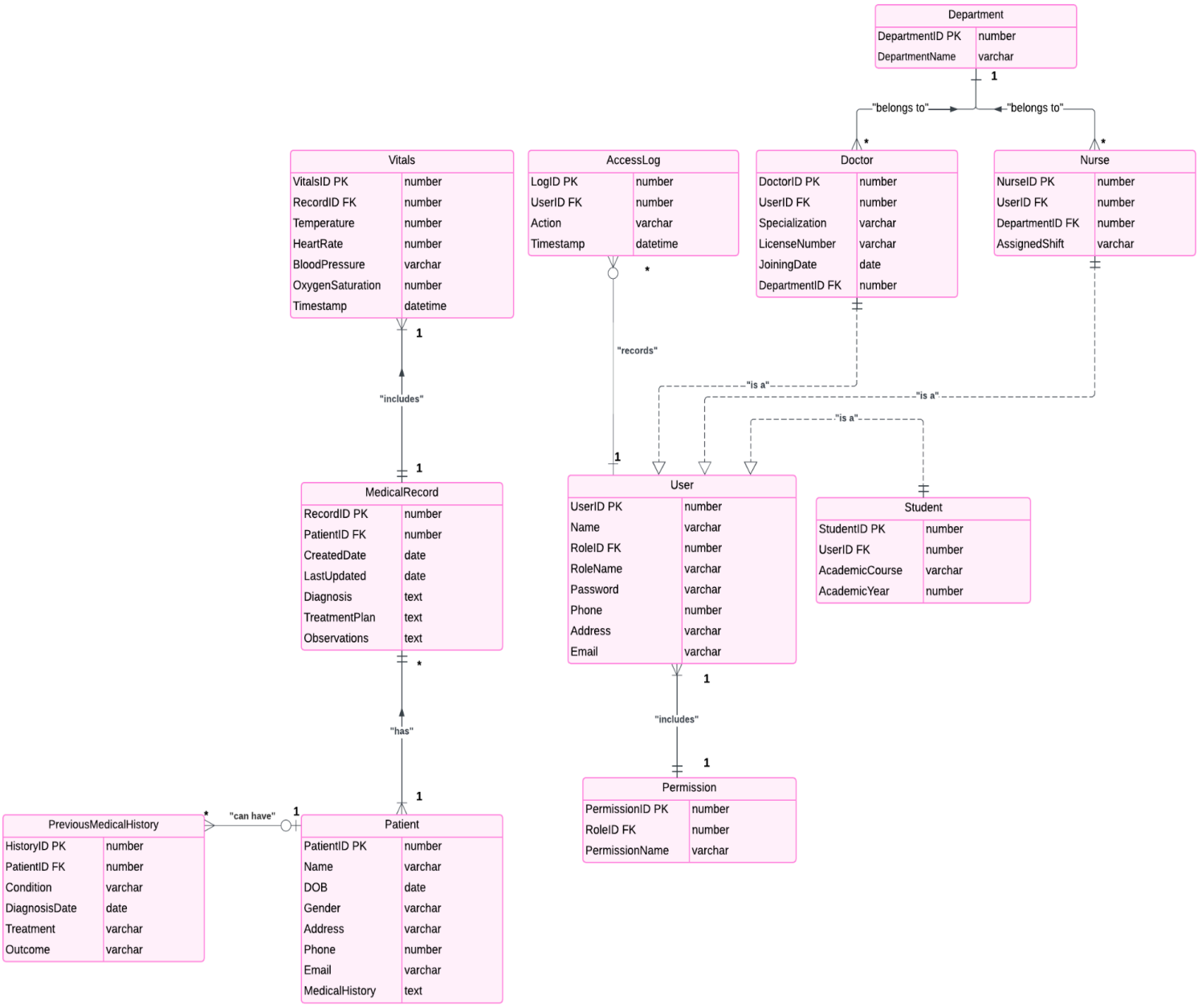


Figure 3.3.1: ER Model

3.4 Normalized Relational Database

3.4.1 Mapping

Figure 3.3.1 shows the database schema after mapping, with entities transformed into tables. Each table includes a primary key and relevant attributes, while relationships from the ER diagram are represented using foreign keys and join tables. The schema reflects all entities and their connections as defined in the original model.

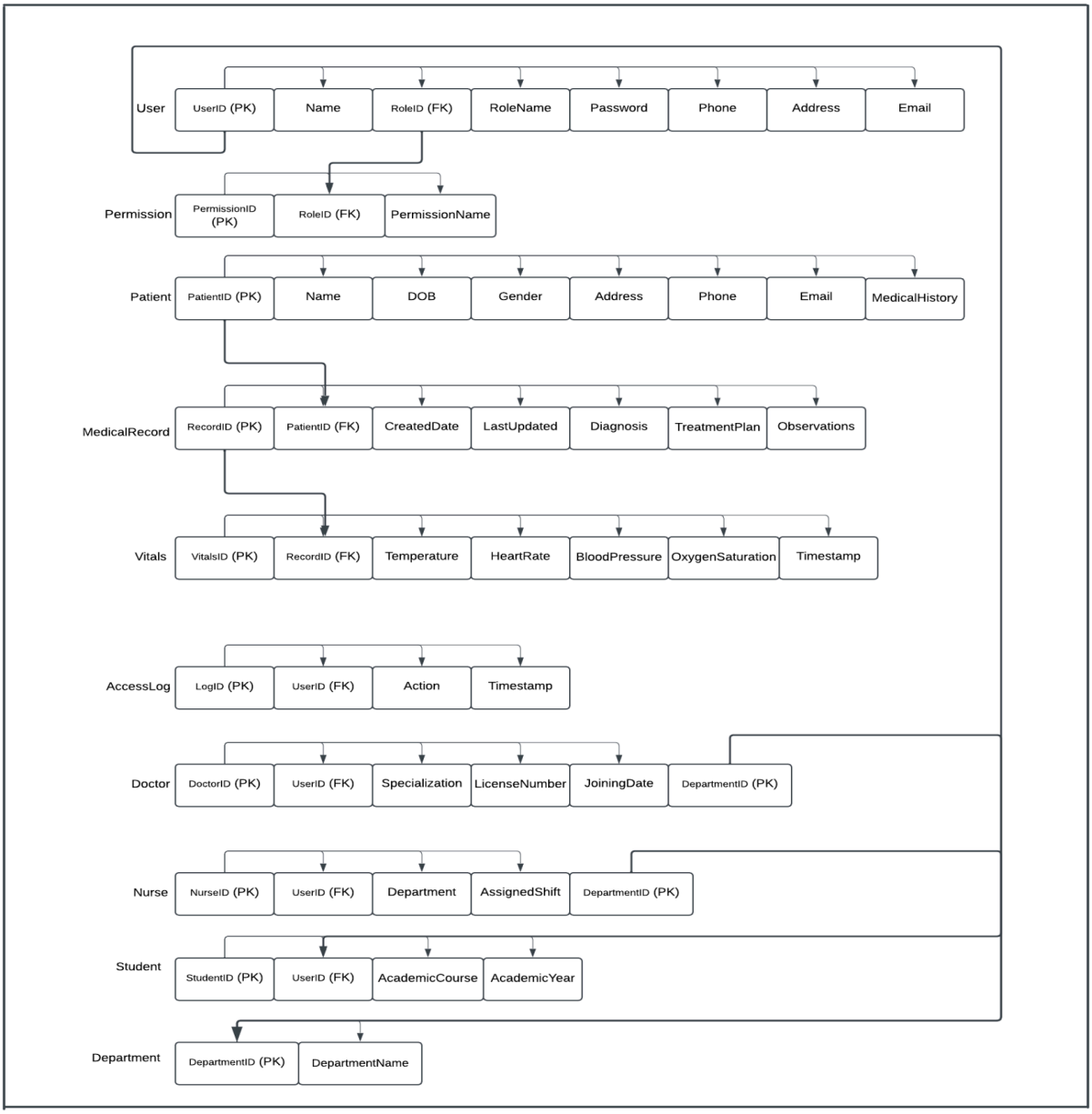


Figure 3.4.1: Relational model after mapping

3.5 Description of Database Table

The system is associated with a database consisting of several tables interconnected with each other through common relationships between them as the following:

- **Database Tables:**

Table name	Description
Patient	Stores patient demographic and medical history information.
User	Contains user account details and their association with roles.
Permission	Defines specific permissions associated with roles.
MedicalRecord	Stores patient medical records, including diagnosis, treatment plan, and observations.
Vitals	Stores vital signs data associated with medical records.
AccessLog	Logs user actions and timestamps for auditing purposes.
Doctor	Stores information specific to doctors, including specialization, license number, and joining date.
Nurse	Stores information specific to nurses, including department and assigned shift.
Student	Stores information specific to students, including academic course and academic year
Department	Stores department details such as department name and unique identifier.
PreviousMedicalHistory	Stores past medical records of patients, including condition, diagnosis date, treatment, and outcome.

Table 3.5.1: Database Tables

- **Patient table:**

Field Name	Data Type	Size	Null
PatientID	Integer	10	No
Name	Varchar	100	No
DOB	Date	-	No
Gender	Varchar	5	No
Address	Varchar	100	Yes
Phone	Varchar	15	Yes
Email	Varchar	50	Yes
MedicalHistory	Text	-	Yes

Table 3.5.2:Patient table

- **User table:**

Field Name	Data Type	Size	Null
UserID	Integer	10	No
Name	Varchar	50	No
RoleID	Integer	10	No
RoleName	Varchar	50	No
Password	Varchar	50	No
Phone	Varchar	15	Yes
Address	Varchar	100	Yes
Email	Varchar	50	No

Table 3.5.3: User table

- **Permission table:**

Field Name	Data Type	Size	Null
PermissionID	Integer	10	No
PermissionName	Varchar	50	No

Table 3.5.4:Permission table

- **MedicalRecord table:**

Field Name	Data Type	Size	Null
RecordID	Integer	10	No
PatientID	Integer	10	No
CreatedDate	Date	-	No
LastUpdated	Date	-	No
Diagnosis	Text	-	Yes
TreatmentPlan	Text	-	Yes
Observations	Text	-	Yes

Table 3.5.5:MedicalRecord table

- **Vitals table:**

Field Name	Data Type	Size	Null
VitalID	Integer	10	No
RecordID	Integer	10	No
Temperature	Integer	10	Yes
HeartRate	Integer	10	Yes
BloodPressure	Varchar	10	Yes
OxygenSaturation	Integer	10	Yes
Timestamp	Datetime	-	No

Table 3.5.6:Vitals table

- **AccessLog table:**

Field Name	Data Type	Size	Null
LogID	Integer	10	No
UserID	Integer	10	No
Action	Varchar	100	No
Timestamp	Datetime	-	No

Table 3.5.7:AccessLog table

- **Doctor Table**

Field Name	Data Type	Size	Null
DoctorID	Integer	10	No
UserID	Integer	10	No
Specialization	Varchar	50	No
LicenseNumber	Varchar	50	No
JoiningDate	Date	-	No

Table 3.5.8:Doctor table

- **Nurse Table**

Field Name	Data Type	Size	Null
NurseID	Integer	10	No
UserID	Integer	10	No
Department	Varchar	50	No
AssignedShift	Varchar	50	No

Table 3.5.9:Nurse table

- **Student Table:**

Field Name	Data Type	Size	Null
StudentID	Integer	10	No
UserID	Integer	10	No
AcademicCourse	Varchar	100	No
AcademicYear	Integer	15	No

Table 3.5.10:Student table

- **Department Table**

Field Name	Data Type	Size	Null
DepartmentID	Integer	10	No
DepartmentName	Varchar	50	No

Table 3.5.11:Department table

- **PreviousMedicalHistory Table**

Field Name	Data Type	Size	Null
HistoryID	Integer	10	No
PatientID	Integer	10	No
Condition	Varchar	100	No
DiagnosisDate	Date	10	No
Treatment	Varchar	100	No
Outcome	Varchar	100	No

Table 3.5.12:PreviousMedicalHistory table

Chapter 4
Implementation

4.1 Overview

In this chapter, we will discuss the techniques used to build backend frontend deploy and another tools

4.2 Technology And Tools Used

Backend :

- **Django:** back-end server side web framework.

Figure 4.2.1: django Backend

- **Restfreamwork:** a powerful and flexible toolkit for building Web APIs.

Figure 4.2.2: Restfreamwork Backend

- **PostgreSQL:** is a powerful, open source object-relational database system.

Figure 4.2.3: PostgreSQL Backend

- **JWT:** JSON Web Token, it is an open standard used to share security information between two parties - a client and a server

Figure 4.2.4: Jwt Backend

Frontend :

- **Figma:** to create user interface designs.

Figure 4.2.5: Figma Frontend

- **Expo:** is a framework and platform for building React Native apps — used by developers to create mobile apps for iOS, Android, and the web.

Figure 4.2.6: Expo Frontend

- **React Native:** is a programming language used mainly to make websites interactive.

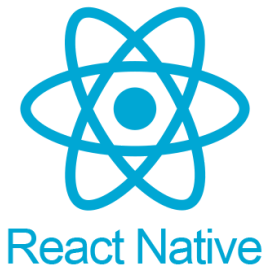


Figure 4.2.7: React Native Frontend

- **Jest:** is a JavaScript testing framework designed to ensure correctness of any JavaScript codebase.

Figure 4.2.8: Jest Frontend

- **Axios:** it is a library to fetch API's.

Figure 4.2.9: Axios Frontend

Deploy and Tools :

- **visual studio code:**is an integrated development environment developed by Microsoft for build code.

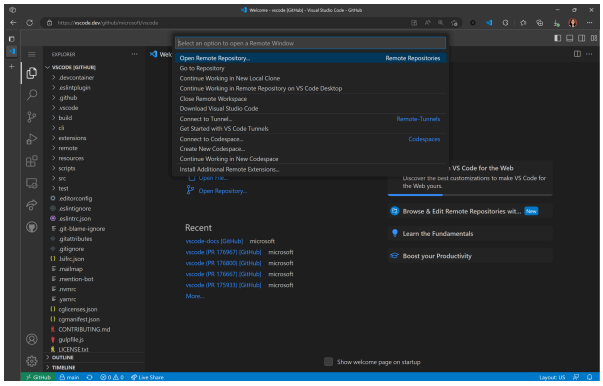


Figure 4.2.10: VS code Deploy

- **GitHub:** To collaborate easily between team members.

Figure 4.2.11: Git & Github Deploy

- **Swagger and Postman:** test , try and make documentation for all endpoints.

Figure 4.2.12: Swagger and Postman Deploy

- **Render:**is a cloud hosting platform that helps developers deploy APIs, databases

Figure 4.2.13: Render Deploy

- **Android Studio:** Google's IDE for Android, supports coding, debugging, and testing with Kotlin/Java and API integration. It features an emulator and Gradle for building apps.

Figure 4.2.14: Android Studio Deploy

4.3 code

Layered API Structure with View, Serializer, and URL Routing Integration

Backend Code , django framework


```python
from django.urls import path
from medical_records.api.views import (
    CreateMedicalRecord,
    CreateVitals,
    MedicalRecordDetail,
    VitalsDetail,
    GetAllMedicalRecords,
    MedicalRecordByPatient,
    MedicalRecordByPatientName,
)

urlpatterns = [
```

Route: to map the URL pattern to the corresponding API view (controller) and handle HTTP methods.

```python
class MedicalRecordUpdateSerializer(serializers.ModelSerializer):
    vitals = VitalsInlineSerializer(many=True, required=False)

    class Meta:
        model = MedicalRecord
        fields = [
            'record_id', 'diagnosis',
            'treatment_plan', 'observations', 'is_public', 'vitals'
        ]

    def update(self, instance, validated_data):
        vitals_data = validated_data.pop('vitals', None)

        for attr, value in validated_data.items():
            setattr(instance, attr, value)
        instance.save()

        if vitals_data is not None:
            instance.vitals.all().delete()
            for vital in vitals_data:
                Vital.objects.create(medical_record=instance, **vital)

        return instance
```

MedicalRecordUpdateSerializer: to handle update logic for MedicalRecord and manage related Vitals data.

```python
class CreateMedicalRecord(APIView):
    permission_classes = [IsAuthenticated, IsAdminOrDoctorOrNurse]

    @swagger_auto_schema(request_body=MedicalRecordSerializer, tags=['medical_records'])
    def post(self, request):
        try:
            data = request.data.copy()
            data.pop('patient', None)

            patient_id = data.get('patient_id')
            if not patient_id:
                return Response({"detail": "patient_id is required."}, status=status.HTTP_400_BAD_REQUEST)

            if not Patient.objects.filter(pk=patient_id).exists():
                return Response({"detail": "Patient does not exist."}, status=status.HTTP_404_NOT_FOUND)

            serializer = MedicalRecordSerializer(data=data, context={'request': request})
            if serializer.is_valid():
                serializer.save()
```

CreateMedicalRecord: to handle HTTP POST requests for creating a new MedicalRecord and validating patient associations.

4.4 Interfaces

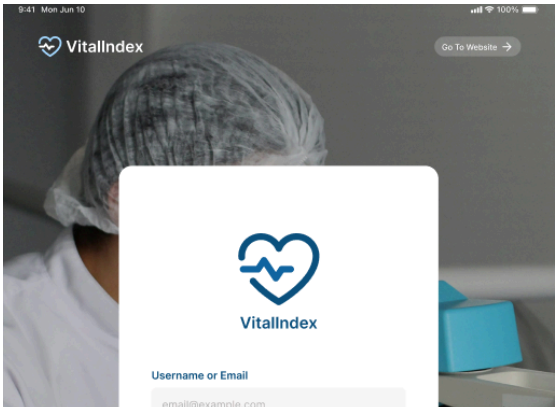


Figure 4.4.1: Login page

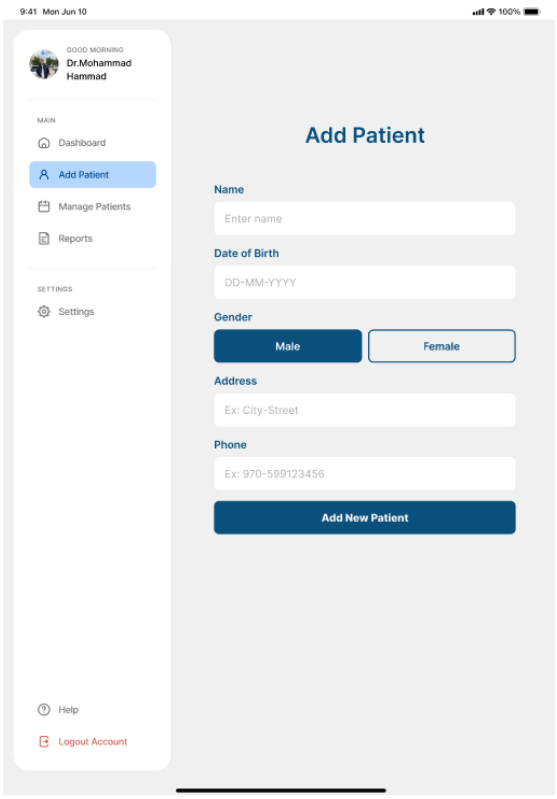


Figure 4.4.2: Add patient page

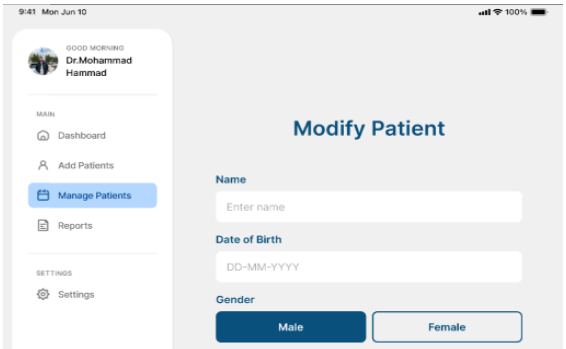


Figure 4.4.3: Modify patient page

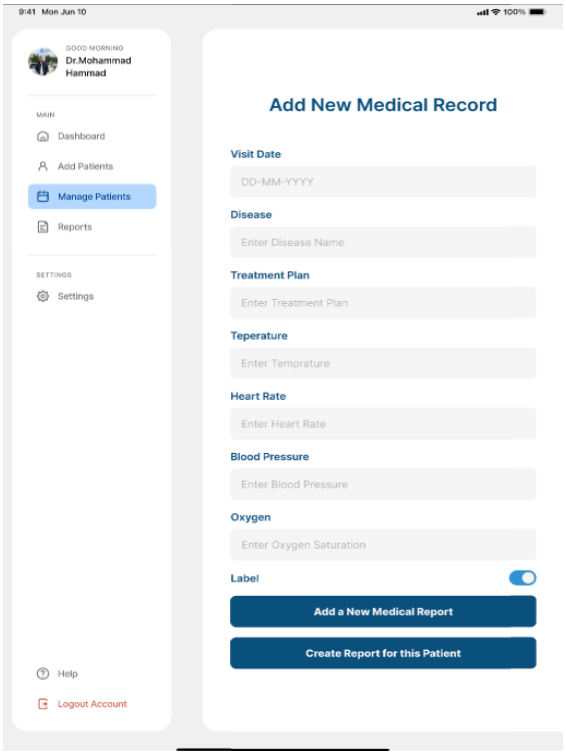


Figure 4.4.4: Add New Medical Record page

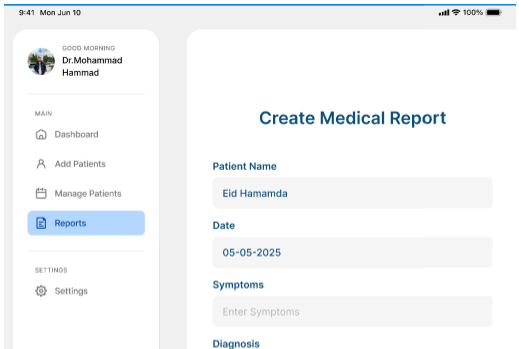


Figure 4.4.5: Create Medical Report page

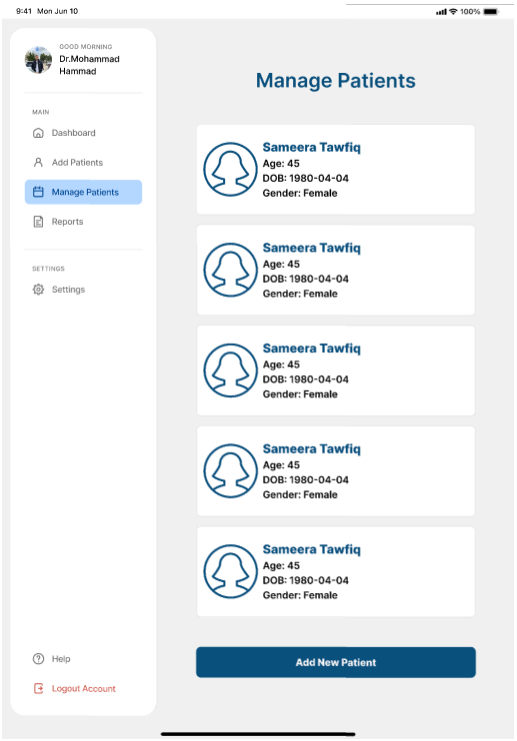


Figure 4.4.6: Manage patients page

Chapter 5
Testing

5.1 Overview

This chapter will discuss the unit testing for backend APIs and functional requirements testing. To ensure that our app is working as expected and helps manage new changes in specification or implementation.

5.2 Validation

All information entered in all fields in the mobile application is checked to ensure that the data entered by the user matches all conditions as follows:

- Customize the field to match the entry.
- The process will not be executed if wrong data is entered.
- The operation is not performed if the fields are empty.
- Ensure that there are actual users in the database.

5.3 Django Report API Test Explanation

5.3.1 Overview

This test suite validates the Create, Read, Update, and Delete (CRUD) operations for the Report API in a Django REST Framework (DRF) application. It simulates authenticated doctor actions and checks both success and failure cases.

Imports (Test Environment Setup)

```
from django.test import TestCase
from rest_framework.test import APIClient
from django.urls import reverse
from users.models import User
from patients.models import Patient
from staff.models import Doctor, Department
from reports.models import Report
from datetime import date
from rest_framework_simplejwt.tokens import RefreshToken
```

- **TestCase**: Django test class, wraps each test in a transaction and rolls back after.
- **APIClient**: DRF client to simulate HTTP requests with auth.
- **reverse**: Resolves URL patterns by name.
- **Model imports**: Used to create test data (User, Patient, Doctor, Report, etc).
- **RefreshToken**: Generates JWT for authenticated requests.

Test Class

```
class ReportAPITest(TestCase):
```

- Defines a unit test suite for testing report-related API endpoints.

setUp Method

```python
def setUp(self):
    self.client = APIClient()
    self.department = Department.objects.create(name='Cardiology')

    self.user = User.objects.create_user(
        username='docuser', password='password', email='doc@example.com', role='Doctor'
    )

    self.doctor = Doctor.objects.create(
        user=self.user,
        specialization="Cardiology",
        license_number="DOC123456",
        joining_date=date(2022, 5, 10),
        department=self.department
    )

    refresh = RefreshToken.for_user(self.user)
    self.client.credentials(HTTP_AUTHORIZATION=f'Bearer {refresh.access_token}')

    self.patient = Patient.objects.create(
        first_name='Ali',
        last_name='Zidan',
        date_of_birth='1990-01-01',
        gender='Male',
        address='Hebron',
        phone='1234567890',
        email='ali@example.com'
    )

    self.report = Report.objects.create(
        doctor=self.doctor,
        patient=self.patient,
        report_title="Initial Report",
        report_type="case_study",
        report_content="Initial content"
    )
```

- The setUp method initializes an authenticated API client, creates a doctor with a linked user and department, then sets up a patient and a related report. This prepares the test environment for validating report API functionality.

```python
def test_create_report_success(self):
    payload = {
        "report_title": "Heart Case",
        "report_type": "case_study",
        "report_content": "Details...",
        "doctor_id": self.doctor.doctor_id,
        "patient_id": self.patient.id
    }
    response = self.client.post(reverse('create-report'), data=payload)
    self.assertEqual(response.status_code, 201)
    self.assertEqual(response.data['report_title'], "Heart Case")

def test_update_report_success(self):
    payload = {
        "report_title": "Updated Report Title",
        "report_content": "Updated content"
    }
    url = reverse('report-detail', args=[self.report.report_id])
    response = self.client.put(url, data=payload)
    self.assertEqual(response.status_code, 200)
    self.assertEqual(response.data['report_title'], "Updated Report Title")

def test_delete_report_success(self):
    url = reverse('report-detail', args=[self.report.report_id])
    response = self.client.delete(url)
    self.assertEqual(response.status_code, 204)
    self.assertFalse(Report.objects.filter(report_id=self.report.report_id).exists())

def test_create_report_invalid_data(self):
    payload = {
        "report_title": "",
        "report_type": "invalid_type",
        "report_content": "Some content"
    }
    response = self.client.post(reverse('create-report'), data=payload)
    self.assertEqual(response.status_code, 400)
    self.assertIn('report_title', response.data)
    self.assertIn('report_type', response.data)
```

- These tests validate the Report API: creating a report with valid data returns 201 Created; updating it returns 200 OK; deleting it returns 204 No Content; and submitting invalid data returns 400 Bad Request with specific error fields.

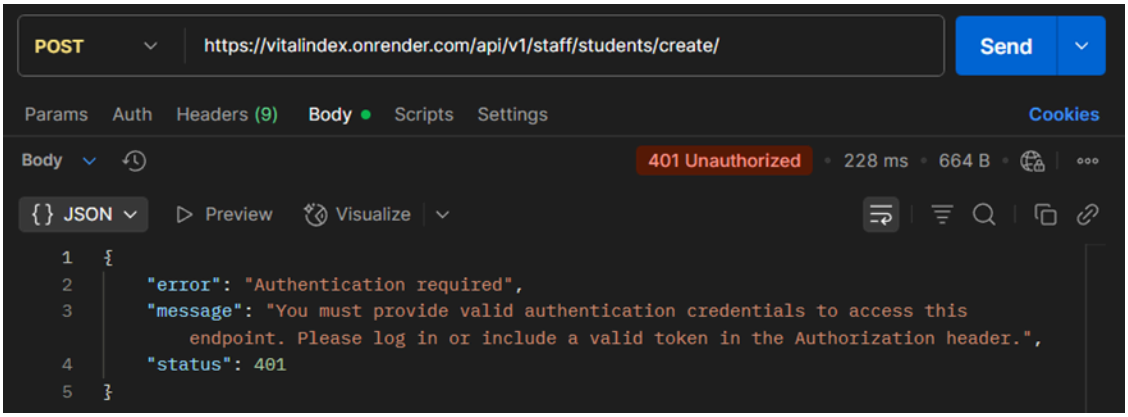
```python
notifications = Notification.objects.filter(user=self.user)
self.assertTrue(notifications.exists(), "No notification created for doctor")
latest_notification = notifications.latest('created_at')
self.assertIn("New report published", latest_notification.content)
self.assertFalse(latest_notification.is_read)
```

- This test block verifies that a notification is properly created when a new report is published by a Doctor.

5.4 Endpoint testing

1- Staff

Creating new student without logging in

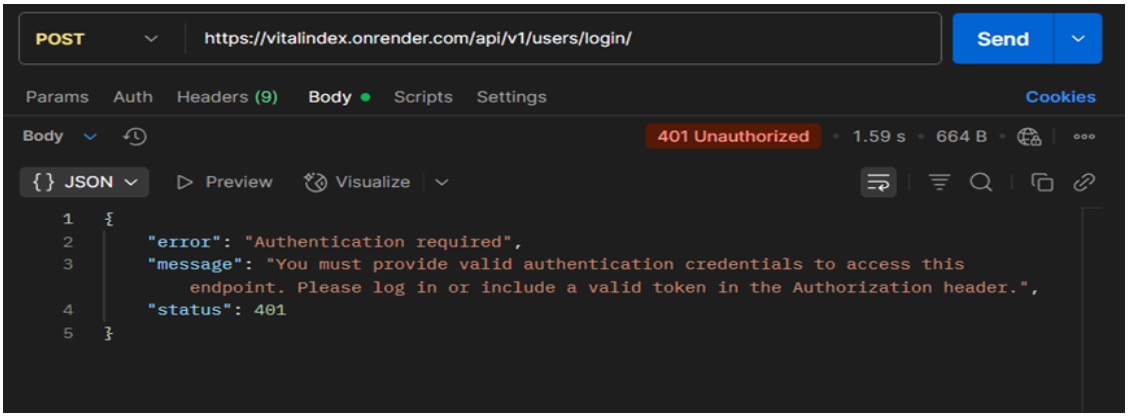


Expected output and status is : Authentication required , 401 Unauthorized

Actual output and status is : Authentication required , 401 Unauthorized

Authentication:

Register with password not match

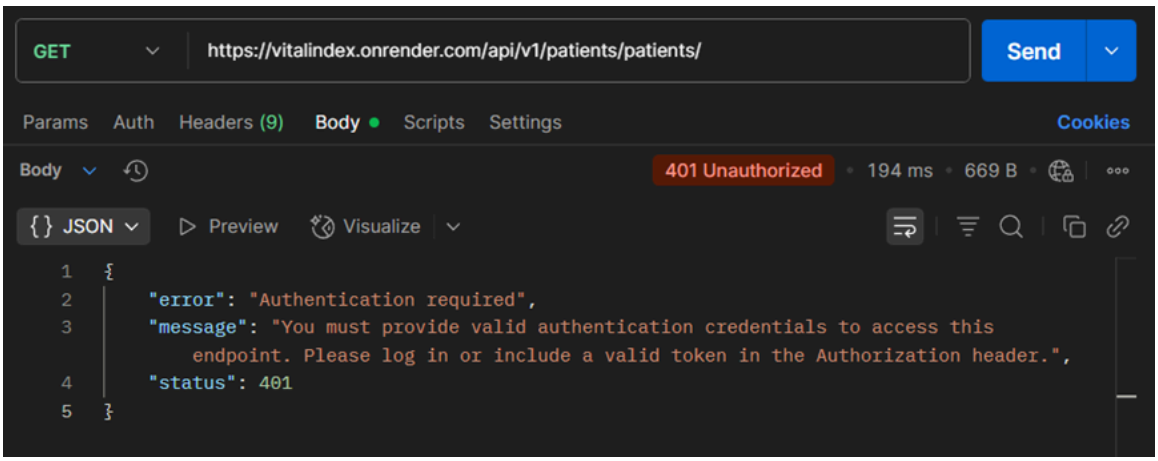


Expected output and status is : Authentication required , 401 Unauthorized

Actual output and status is : Authentication required , 401 Unauthorized

2- Patients

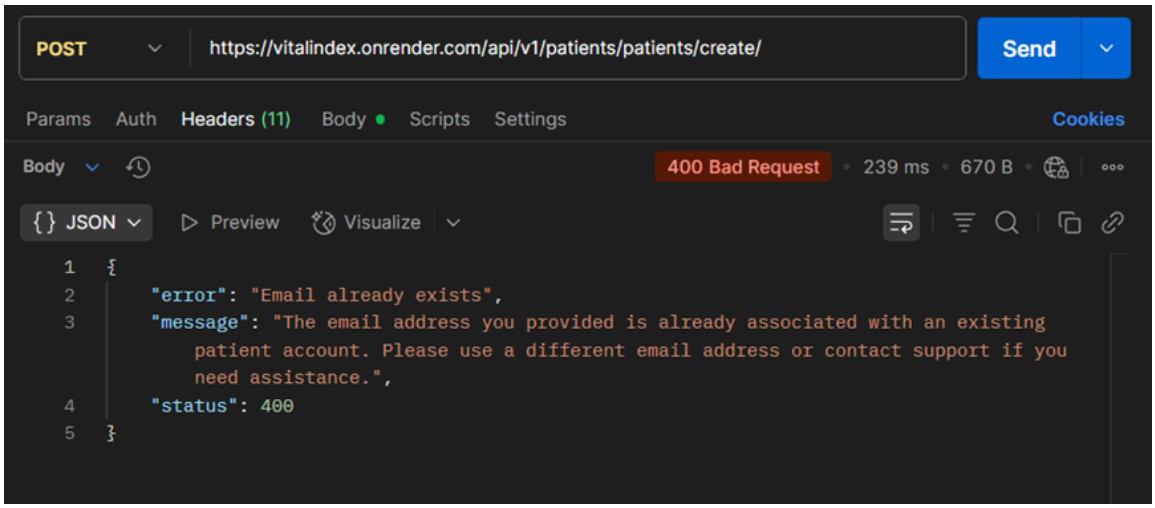
Getting patient without logging in



Expected output and status is : Authentication required , 401 Unauthorized

Actual output and status is : Authentication required , 401 Unauthorized

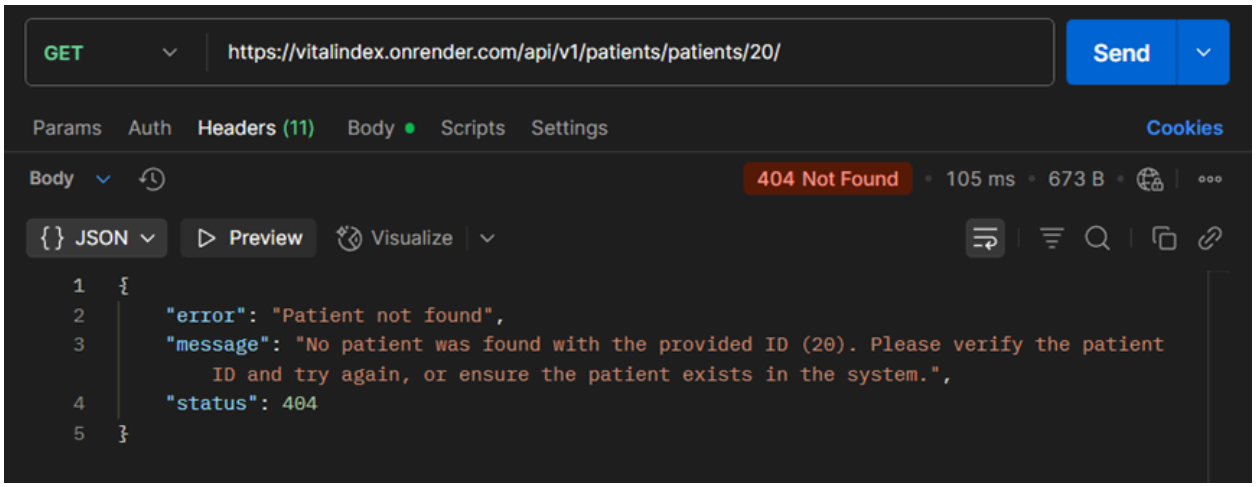
Creating patient with already taken email



Expected output and status is : Email already exists , 400 , Bad Request

Actual output and status is : Email already exists , 400 , Bad Request

Getting non-existing patient

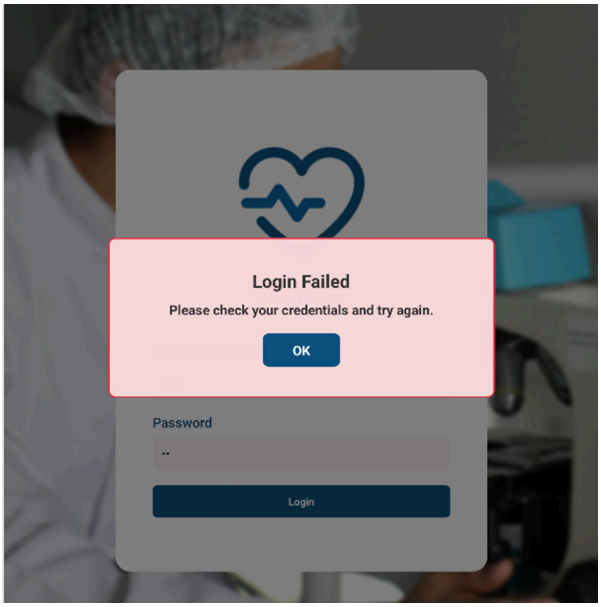


Expected output and status is : Patient not found , 404 , Not Found

Actual output and status is : Patient not found , 404 , Not Found

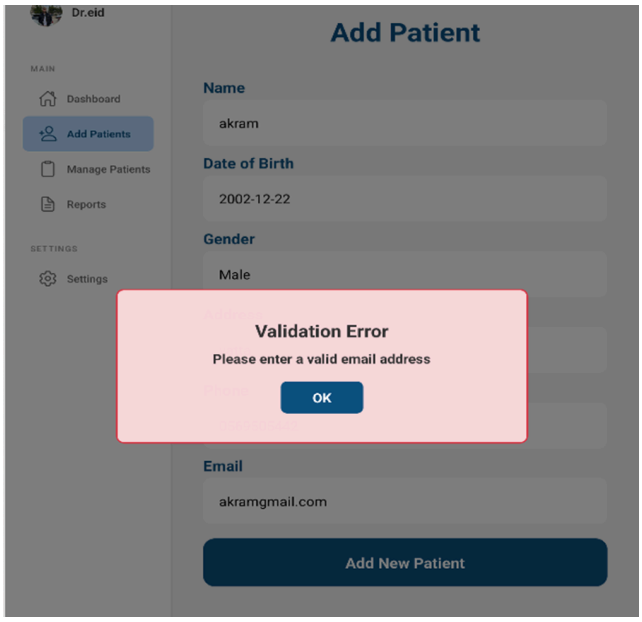
5.5 QA Manual testing

- Logging with non-valid user name or password



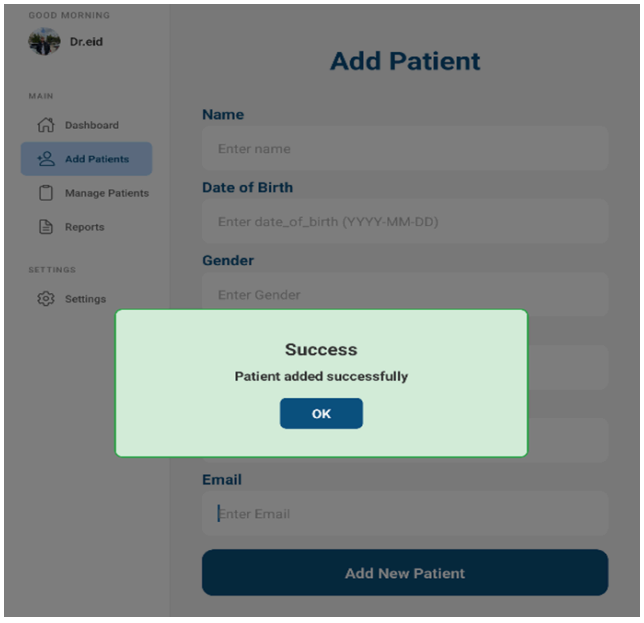
Expected : Login Failed
Actual : Login Failed

- Creating new patient with non-valid email



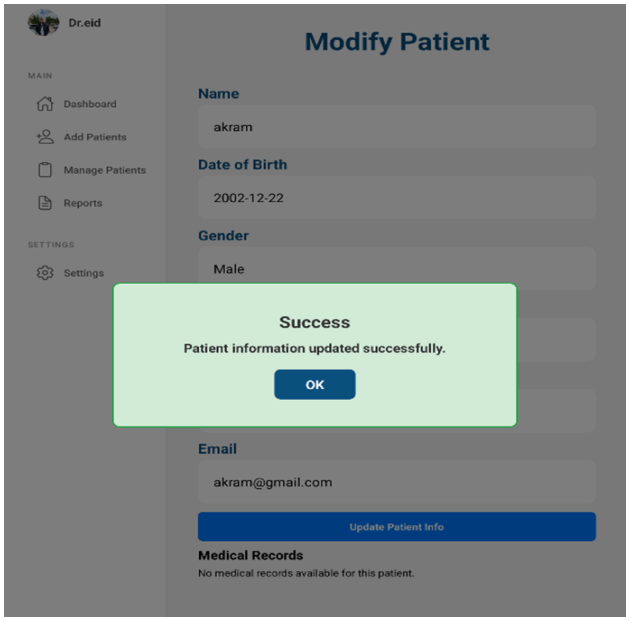
Expected : Validation Error
Actual : Validation Error

- Creating new patient



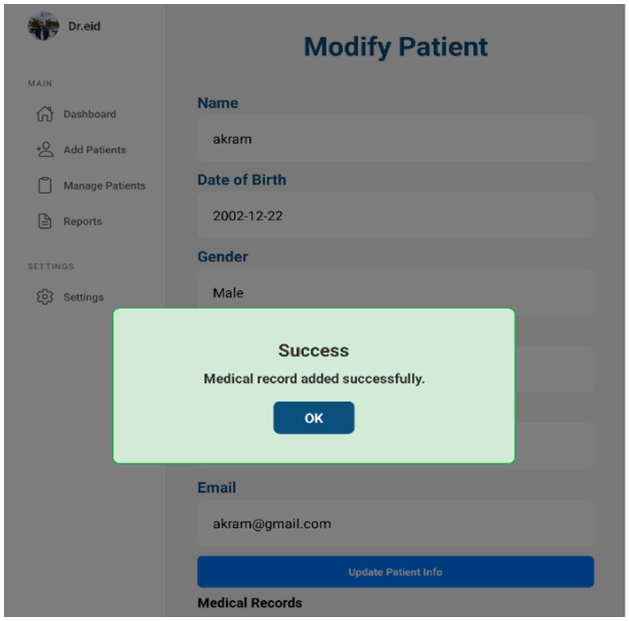
Expected :Success
Actual : Success

- Update patient basic information



Expected :Success
Actual : Success

- Create new medical record



Expected :Success
Actual : Success

Chapter 6
Conclusion and Future Work

6.1 Conclusion

In the end, the Mobile hospital information system is a landmark improvement in the management of health-related data and information by providing a mobile-friendly design that efficiently allows access to patient information and improves medical decision-making. With its onboarding of role-based access, data sanitation, and real-time access to patient data, the system directly addresses presentations in modern health care. The project also promotes efficiency in operations, and importantly, provides an opportunity for those students seeking an education in medicine to more responsibly engage with real-world patient data. Because of its scalable and secure infrastructure, Mobile hospital information system has established a new frontier of digital transformation in the management of health-related data, while delivering a better, more effective, reliable, and patient-centered approach to medical information management.

6.2 Future Work

Future directions for the Mobile hospital information system system may involve enhancing its interoperability with other healthcare information systems and electronic health records (EHR) for seamless data exchange throughout medical systems. Equally, use of artificial intelligence and machine learning capabilities may improve predictive analytics, leading to clinicians appropriately diagnosing and recommending treatment plans for diseases. Likewise, expanded function through telemedicine elements supports remote consultations and telehealth for underdeveloped areas. Not only would these enhancements improve the Mobile hospital information system system but also contribute to continued digitization of healthcare services.

References:

1. Django. (n.d.). Django: Back-end server-side web framework. [online] Available at: https://www.djangoproject.com/.

2. Django REST framework. (n.d.). A powerful and flexible toolkit for building Web APIs. [online] Available at: https://www.django-rest-framework.org/.

3. PostgreSQL. (n.d.). Powerful, open source object-relational database system. [online] Available at: https://www.postgresql.org/.

4. JWT.io. (n.d.). JSON Web Token (JWT): A compact and self-contained way to transmit information securely. [online] Available at: https://jwt.io/.

5. Figma. (n.d.). Design tool for creating user interfaces. [online] Available at: https://www.figma.com/.

6. Expo. (n.d.). Framework for building React Native apps. [online] Available at: https://expo.dev/.

7. React Native. (n.d.). Create native apps for Android and iOS using React. [online] Available at: https://reactnative.dev/.

8. Jest. (n.d.). Delightful JavaScript Testing Framework. [online] Available at: https://jestjs.io/.

9. Axios. (n.d.). Promise based HTTP client for the browser and node.js. [online] Available at: https://axios-http.com/.

10. Visual Studio Code. (n.d.). Integrated development environment by Microsoft. [online] Available at: https://code.visualstudio.com/.

11. GitHub. (n.d.). Code hosting platform for version control and collaboration. [online] Available at: https://github.com/.

12. Swagger. (n.d.). Design, build, document, and use REST APIs. [online] Available at: https://swagger.io/.

13. Postman. (n.d.). Platform for API development and testing. [online] Available at: https://www.postman.com/.

14. Render. (n.d.). Cloud platform to build and host applications and databases. [online] Available at: https://render.com/.

15. Android Studio. (n.d.). Official IDE for Android development. [online] Available at: https://developer.android.com/studio.

Appendix

نموذج إقرار وتوثيق استخدام أدوات الذكاء الاصطناعي في مشروع التخرج

اسم الطالب / أسماء الطلبة: mohammad hammad Omar Sweiti Eid Hamamda	**عنوان المشروع:** Mobile Hospital information system (VitalIndex)
الرقم / الأرقام الجامعية: 201075 211183 211023	**اسم المشرف الأكاديمي:** Liana Tamimi
الكلية / الدائرة: College of Information Technology & Computer Engineering	**السنة/الفصل الدراسي:** 2025 / الثاني

الغرض من الاستخدام	اسم الأداة	أماكن الاستخدام في التقرير (فصول / صفحات)	الأمر الأساسي المستخدم (Prompt)	هل تم تعديل الناتج ؟	نسبة الاستخدام التقديرية	ملاحظات
توليد نصوص	☐ ChatGPT	the introduction, chapter 1.4 Page 9	"Read this file and write to me Importance of the Project and Digital Data in Healthcare"	Yes	3%	We did not write the text as it is, but rather summarized it in our own terms
	☐ Gemini					
	☐ Claude					
	☐ Microsoft Copilot					
	☐ other: ____					
تدقيق لغوي	☐ Grammarly					
	☐ Quillbot					
	☐ Word Editor AI					
	☐ other: ChatGPT	Full report	"Read this file and check all spelling errors and correct these errors without	Yes	8%	Spelling errors have been checked by us because the file is written in English.

			changing the words in the file"			
This paragraph has been abbreviated due to its length for a small text.	2%	Yes	"Read this text and Summarize this for me in a short paragraph without changing the content of the existing text"	the introduction, chapter 1.7 Page 12	☐ ChatGPT	**تلخيص محتوى**
					☐ SMMRY	
					☐ Notion AI	
					☐ Scholarcy	
					☐ other: ____	
	0%	No			☐ Excel Copilot	**تحليل بيانات**
					☐ Python AI Assistants	
					☐ Tableau AI	
					☐ other: ____	
	0%	No			☐ DALL·E	**رسم مخططات / أشكال**
					☐ Canva AI	
					☐ Visio AI	
					☐ Lucidchart AI	
					☐ other: ____	
					☐ GitHub Copilot	**كتابة أكواد برمجية**
					☐ Replit AI	
					☐ Codeium	
:(class CreateMedicalRecord(APIView permission _classes = [IsAuthenticated, [IsAdminOrDoctorOrNurse	5%	Yes	How to add body to ui swagger	BackEnd	☐ other: ChatGPT	

						swagger_auto_schema(request_body=@ MedicalRecordSerializer, (['tags=['medical_records ﹤ :(def post(self, request serializer = MedicalRecordSerializer(data=request.d (ata :()if serializer.is_valid (serializer.save return Response (serializer.data, (status=status.HTTP_201_CREATED return Response(serializer.errors, status=status.HTTP_400_BAD_REQUE (ST _____
توثيق مراجع	☐ EndNote		No			
	☐ Mendeley					
	☐ Zotero					
	☐ ChatGPT					
	☐ other: ____					
أخرى (حدد): _____						

إقرار فريق مشروع التخرج:

نقر بأن استخدام أدوات الذكاء الاصطناعي تم بشكل مسؤول وبما يتوافق مع السياسات الجامعية، وقد راجعنا وحررنا كافة المخرجات بما يعكس فهمنا الشخصي.

توقيع الطالب / الطالب: **التاريخ: 2 - 6 -2025**

توقيع الطالب / الطالب:

توقيع الطالب / الطالب: