DIGITAL HEALTH RECORDS SYSTEM

A project submitted in partial fulfilment of the requirement for the award of the degree.

BACHELOR OF TECHNOLOGY (HONORS)

IN

SOFTWARE ENGINEERING

HARARE INSTITUTE OF TECHNOLOGY

Submitted by

NGONIDZASHE MANGUDYA
(H180202M)

Under the supervision of
MR. MUKOSERA

CERTIFICATE OF DECLARATION

This is to certify that work entitled **DIGITAL HEALTH RECORDS SYSTEM** is submitted in partial fulfilment of the requirements for the award of Bachelor of Technology (Hons) in Software Engineering, Harare Institute of Technology. It is further certified that no part of research has been submitted to any university for the award of any other degree.

(Mangudya Ngonidzashe)
Student's Signature .....
Date ..... /...../.....

(Mr Mukosera)
Supervisor's Signature .....
Date ..... /...../.....

(Mentor)
Mentor's Signature .....
Date ..... /...../.....

(Chairman)
Chairman's Signature .....
Date ..... /...../.....

ABSTRACT

Ngonidzashe Mangudya: Digital Health Records for Zimbabwe
(Under the direction of Mr Mukosera)

The current landscape of patient information management in healthcare organisations is rife with inefficiencies and shortcomings that are undermining the quality of care delivered to patients. Despite a plethora of methods that have been developed to store and manage patient information, there remains a scarcity of objective evidence demonstrating their impact on healthcare quality and outcomes. This lack of evidence has made it difficult for healthcare organisations to adopt new information management technologies and methods, leaving many to rely on outdated and cumbersome pen-and-paper methods that waste valuable time and resources. To address these challenges, we propose a low-cost digital health records system that is designed to capture, securely store, and manage all patient-related information and activities. The proposed system leverages the power of digital technology to enable healthcare organisations to seamlessly manage patient information, streamline workflows, and provide more informed and efficient care. By incorporating advanced security features and strict privacy protocols, our system ensures that patient information always remains safe and secure, without compromising accessibility or ease of use. Perhaps most importantly, our system places a strong emphasis on patient engagement and empowerment, enabling patients to stay informed about their care and take an active role in their health management. Overall, the low-cost digital health records system represents a game-changing solution that has the potential to transform the healthcare industry. By improving the quality and efficiency of patient information management, the proposed system enables healthcare organisations to deliver better care, reduce costs, and improve overall health outcomes for patients.

Keywords
Digital health records system; Patient information management; Healthcare quality; Patient engagement; Security and privacy protocols

ACKNOWLEDGEMENTS

I want to extend my heartfelt appreciation to my friends and family who provided invaluable support during the various stages of my project. They helped me in gathering requirements, assessing the feasibility of the solution, and offering their suggestions on designing a user-friendly interface that ensures an excellent user experience. I would also like to express my special thanks to my project supervisor, Mr. Mukosera, for providing me with guidance and support throughout the entire development process.

TABLE OF CONTENTS

DIGITAL HEALTH RECORDS SYSTEM 1
CERTIFICATE OF DECLARATION 3
ABSTRACT 4
ABBREVIATIONS 11
CHAPTER ONE: INTRODUCTION 12
1.1 Background 12
1.2 Problem Statement 12
1.3 Objectives 13
1.4 Hypothesis 13
1.5 Justification 14
1.6 Proposed Tools 14
1.7 Feasibility Study 14
1.8 Project Plan 17
CHAPTER TWO: LITERATURE REVIEW 19
2.1 Introduction 19
2.2 Related Work 19
i. Electronic health records in chiropractic practice 19
ii. Adopting EMRs and Information Technology in Primary Care 20
iii. Electronic Health Record Implementation in Developing Countries 22
Conclusion 23
CHAPTER THREE: ANALYSIS 24
3.1 Description of the system 24
3.2 Information Gathering Tool 24
Case Studies 24
Interviews 24
3.3 Data Analysis 25
Context Diagram 25
3.4 Functional Analysis 26
Functional Requirements (Proposed System Use Case Diagram) 26

3.5 Non-functional Requirements 27
System Performance 27
Security 27
Deliverable 28
Usability 28
CHAPTER FOUR: DESIGN 29
4.1 Introduction 29
Purpose 29
Scope 29
Overview 29
4.2 System Overview 30
4.3 Activity Diagrams 32
4.4 Sequence Diagrams 33
4.5 Architectural Design 36
System Model 36
4.6 Modular Decomposition 38
4.7 Database Design 38
4.8 Overview of the User Interface 42
CHAPTER FIVE: IMPLEMENTATION AND TESTING 51
5.1 Coding Standards 51
Internal Documentation Standards 51
Indentation 51
Classes, Subroutines, Functions and Methods 52
Variable Names 52
Version Control 52
5.2 Implementation Strategy 54
Components To Be Implemented 54
Development Approach 55
5.3 Code Sample 56
5.4 Software Test Plan 65
Features Tested 65

Features Not Tested 65
5.5 Testing Strategy 66
Item Pass/Fail Criteria 66
5.6 Test Case Specification 66
5.7 Test Reports 69
Incidents 69
Defects 69
5.8 Deployment 69
Desktop Client 69
Mobile Application 70
Backend API 70
CHAPTER SIX: CONCLUSION AND RECOMMENDATIONS 71
6.1 Results 71
Discussion of Results 71
6.2 Summary 72
6.3 Recommendations 72
6.4 Future Work 72
Bibliography 74

LIST OF TABLES

Table 1: Components of the system 36
Table 2: Components to be implements 54
Table 3: Test Case Specification 66

LIST OF FIGURES

Figure 1: Project Plan 17
Figure 2: Context Diagram 26
Figure 3: Use case diagram for Digital Health Records System 27
Figure 4: Level 0 DFD Digital Health Records System 30
Figure 5: Level 1 DFD Digital Health Records System 31
Figure 6: Patient Check-In Activity Diagram 32
Figure 7: Patient Mobile App Activity Diagram 33
Figure 8: Patient Check-In Sequence Diagram 34
Figure 9: Dispensary Sequence Diagram 35
Figure 10: Patient Mobile App Sequence Diagram 36
Figure 11: Architectural Diagram 38
Figure 12: Database Models 41
Figure 13: Mobile App Screenshots 1 42
Figure 14: Mobile App Screenshots 2 43
Figure 15: Desktop App Login Screen 43
Figure 16: System Admin Management Dashboard 44
Figure 17: System Admin Register New Admins 44
Figure 18: System Admin Register New Health Institutions 45
Figure 19: Change Password 45
Figure 20: Health Institution Employees Management 46
Figure 21: Health Institution Register New Employees 46
Figure 22: Nurse Dashboard 47

Figure 23: Nurse Point of Service 47
Figure 24: Nurse Point of Service Prescribe Medication 48
Figure 25: Pharmacist Dispensary 48
Figure 26: Dispensary Payments (using Ecocash) 49
Figure 27: Approved Medicines Management 49
Figure 28: Health Institution Medicines Management 50
Figure 29: Project Repository Branches 53
Figure 30: Project Repository Merged Changes 53
Figure 31: Feature Driven Development Methodology 56

ABBREVIATIONS

EHR - *Electronic Health Record*

IT - *Information Technology*

HIPAA- *Health Insurance Portability and Accountability Act*

API - *Application Programming Interface*

SQL - *Structured Query Language*

UI - *User Interface*

UX - *User Experience*

REST - *Representational State Transfer*

CPU - *Central Processing Unit*

GB - *Gigabyte*

HDD - *Hard Disk Drive*

IDE - *Integrated Development Environment*

iOS - *iPhone Operating System*

macOS - *Macintosh Operating System*

Mb - *Megabyte*

OS - *Operating System*

RAM - *Random Access Memory*

SSD - *Solid State Drive*

VPS - *Virtual Private Server*

CHAPTER ONE: INTRODUCTION

1.1 Background

Traditionally paper has been standardised and in use for so long in many countries as a way of recording patients' health information i.e., from birth up until death, their health records are on various papers. This was very successful at the time since it required less upfront setup costs i.e., all that was required was paper and file cabinets to store the papers. This also meant fewer costs for training personnel. Doing this on paper has led to many complications within the healthcare environment, for example, storing this information in file cabinets means there is a lack of security, and no data is backed up. The emergence of electronic methods of storing information has resulted in significant advances in the ease and flexibility in which information can be stored, manipulated, backed up and analysed to quickly identify any patterns that would be of importance to stakeholders in the healthcare industry.

1.2 Problem Statement

In today's technologically advanced era, it is astonishing that many healthcare facilities still rely on a paper-based record-keeping system. This antiquated method of documenting patient information has a myriad of drawbacks that can lead to serious consequences. One of the most prominent issues with paper records is that they can be easily misplaced or destroyed, leading to missing or distorted information. Inaccurate information can have severe implications for patient care, and in some cases, may even result in malpractice.

The lack of proper data to support medical decisions is another major issue with paper-based record-keeping systems. This can lead to wrong diagnoses or incorrect treatment plans for patients, which can have serious consequences. Furthermore, the current system makes it challenging to have a seamless flow of information from one healthcare facility to another, which can create loopholes in how patients are handled and treated across different healthcare facilities.

Recording data on paper also poses a significant problem when it comes to analysing the information. The analysis of data is crucial in healthcare, and relying on paper records makes this process time-consuming and often results in false information due to human error. This, in turn, can lead to erroneous conclusions and flawed decisions.

In conclusion, the current paper-based record-keeping system in healthcare is plagued with a host of problems that can have significant consequences for patients. With the advancement of technology, it is essential for healthcare facilities to adopt modern record-keeping systems that are reliable, secure, and efficient. This will not only enhance patient care but also improve the overall efficiency and effectiveness of the healthcare system.

1.3 Objectives

- To develop a desktop platform for health facilities to:
    - securely collect, store and retrieve patient information (visits, tests, prescriptions, etc)
    - generate analytical reports at any level for any specified period.
    - Provide secure APIs for sharing of patient information with other health care providers.
- To develop a mobile platform for patients to:
    - Access their information.
    - Receive notifications of anything related to their health (e.g., reminders to take medication as prescribed)

1.4 Hypothesis

- The healthcare facilities have a working moderate internet connection.
- The healthcare facilities are well equipped with desktop computers or laptops.
- The healthcare personnel have the applicable knowledge of how to operate computers.

1.5 Justification

- Analytical reports can be generated which saves time (most clinics take more than 2 days to compile these reports) and results in quicker responses to issues.
- Having data in one place means it can be easily shared with multiple healthcare providers which eliminate the loopholes in how differently a patient can be treated at different healthcare facilities.
- The system notifies patients about most things about their health including reminders to take medication which also improves communication with patients after they have been discharged.
- The collection of such information in high volumes means we can now leverage innovative technologies such as big data.
- The system allows for the backing up of data which significantly reduces the probability of losing the data in case of any inconveniences.

1.6 Proposed Tools

- Desktop Platform - Flutter
- Mobile Platform - Flutter
- Backend Service - Python Django
- Database - PostgreSQL
- Deployment - DigitalOcean VPS (Docker Containers)
- Version Control - Git, GitHub

1.7 Feasibility Study

It is the detailed analysis that considers all the critical aspects of the proposed project to determine whether it is practical and will be successful. It takes the project's critical elements into account including technical, economic and operational aspects to determine the likelihood of success. The purpose of the feasibility study is to investigate the depth and provide information that justifies the development of the Digital Health Records System. The feasibility of this project should be evaluated throughout every stage of the development lifecycle.

Technical Feasibility

Assessment of the details (technology assumptions, architecture and design) of how the project will be delivered. During this study technical requirements are weighed against the technical capabilities of the resources available.

Software Requirements

- Desktop Platform - Flutter
- Mobile Platform - Flutter
- Backend Service - Python Django
- Database - PostgreSQL
- Deployment - DigitalOcean VPS (Docker Containers)
- Version Control - Git, GitHub

System Requirements

- Windows 10 or later
- macOS 10.11 or later
- Linux (that supports snaps)
- Android 6+ or iOS 12+
- Android Studio IDE
- PyCharm Professional IDE
- PostgreSQL Server & PgAdmin 4

Hardware Requirements (Desktop)

- 2GHz + CPU processor
- Minimum of 8 GB RAM
- Minimum of 100GB SSD (or HDD) to run the applications.

Hardware Requirements (Mobile)

- 1GB RAM minimum
- 1GHz CPU Processor
- 500MB disk space minimum

Operational Feasibility

Measured all the proposed aspects of the solution to see how well they solved the problem and how well they addressed or satisfied the requirements. This study looked at things like affordability, usability and reliability.

- Users with basic computer skills can operate the applications.
- An in-app help guide helps users find their way around the application.
- The system shows how time can be drastically saved in generating analytical reports.

Economic Feasibility

It analysed the costs of developing and maintaining the system against the gains from using such a system to determine whether the project is capable of generating significant financial returns.

Costs

Development Software (Free) - $0
Data Persistence Services (Free) - $0
Internet Data - $60 / month
Deployment Platform - $28 * 2 / month

Forecasted Benefits

Approx. Current Costs = (Approx. $4 / patient on paper) x (Approx. 100 patients per month) = $400

Approx. Forecasted Savings = $400 - $116 = $284 would be saved

Return On Investment

Cost Per Month = $116

Savings (Benefits) Per Month = $284

ROI = ($284 / $116) * 100 = $245%

The overall cost of development and maintenance of the system is outweighed by the savings or benefits that would be obtained from cutting costs of acquiring paper per month, therefore using the high rate of return on investment the study concludes that the project is economically feasible.

1.8 Project Plan

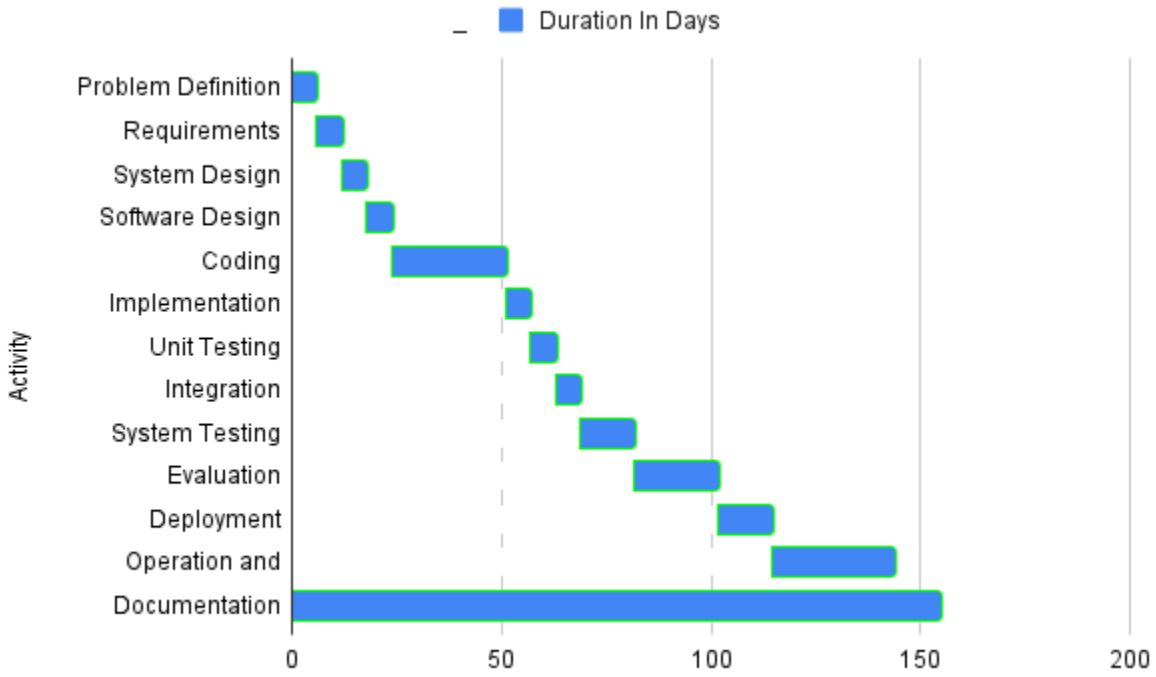

Figure 1: Project Plan

CHAPTER TWO: LITERATURE REVIEW

2.1 Introduction

This chapter examines the systems, subjects, studies, and methodologies linked to electronic health records. The main purpose of this section is to perform a study and analysis of the work related to this project and the mechanisms used in the previous studies. Here, known flaws with the earlier work will be identified, and the best way to incorporate fresh ideas into the project using the earlier work as a guide will be used to lead it in the direction. In order to describe, summarise, and critically evaluate these works in relation to the research problem under investigation, this chapter primarily examines books, academic articles, and any other materials pertinent to the subject under study.

2.2 Related Work

This section's goal is to examine prior work on critical analysis, pinpoint significant issues, and determine the best strategy for resolving the current issue and incorporating fresh ideas into the project.

i. Electronic health records in chiropractic practice

(Taylor, David. (2017). A Literature Review of Electronic Health Records in Chiropractic Practice: Common Challenges and Solutions. Journal of Chiropractic Humanities. 24. 10.1016/j.echu.2016.12.001.)

In the article, "A Literature Review of Electronic Health Records in Chiropractic Practice: Common Challenges and Solutions," David Taylor explores the use of electronic health records (EHR) in chiropractic practice. Taylor notes that while there is increasing evidence of EHR having a positive impact on healthcare delivery, it is not without its challenges.

One of the key benefits of EHR in chiropractic practice is improved efficiency. The use of EHR can streamline the process of entering and accessing patient information, which can save time

and reduce errors. In addition, EHR can help to standardise treatment plans and aid communication between healthcare providers.

Another advantage of EHR in chiropractic practice is increased accuracy. Digital records make it easier to store, organise, and analyse data, which can help practitioners to identify patterns and make more informed decisions about patient care. Moreover, EHR can improve patient satisfaction by reducing wait times and providing easy access to their health records.

Despite these benefits, there are challenges associated with EHR in chiropractic practice. One such challenge is the cost of implementing and maintaining the technology, which can be prohibitive for some practitioners. Additionally, EHR systems can be difficult to use, and their implementation requires significant changes to workflow.

Regulatory compliance is another challenge that chiropractors face with the use of EHR. This is because EHR systems must meet requirements set by government and industry regulators, such as the Health Insurance Portability and Accountability Act (HIPAA).

In conclusion, the article recommends that chiropractors should adopt EHR systems, but should also be careful to choose the right system and implement it effectively. By doing so, chiropractors can take advantage of the benefits of EHR while minimising the associated challenges.

ii. Adopting EMRs and Information Technology in Primary Care

(Carbone, Sarah & Peckham, Allie & Bhatia, Dominika & Allin, Sara & Marchildon, Gregory. (2020). Adopting EMRs and Information Technology in Primary Care. NAO Rapid Review No. 18.)

The adoption of Electronic Medical Records (EMRs) and Information Technology (IT) in primary care has become increasingly important in recent years. In their article "Adopting EMRs and Information Technology in Primary Care," authors Sarah Carbone, Allie Peckham,

Dominika Bhatia, Sara Allin, and Gregory Marchildon explore the impact of EMR and IT adoption on primary care, as well as the factors that influence successful adoption.

The authors conducted a rapid review of the literature on EMR and IT adoption in primary care, focusing on studies published between 2015 and 2019. The review revealed several key findings. First, the adoption of EMRs and IT in primary care can improve the quality of care delivered to patients, as well as patient outcomes. Specifically, EMRs and IT can improve communication between healthcare providers, facilitate better coordination of care, reduce medical errors, and improve patient engagement.

However, the authors note that successful adoption of EMRs and IT in primary care is not guaranteed. Several factors can influence adoption, including organisational culture, resource availability, and technical expertise. In addition, the authors note that the cost of implementing EMRs and IT can be a barrier for some primary care providers, particularly those in under-resourced communities.

To address these challenges, the authors suggest several strategies for improving the adoption of EMRs and IT in primary care. These strategies include providing training and technical support for primary care providers, ensuring that EMRs and IT are user-friendly and meet the needs of primary care providers and patients, and increasing funding and resources to support adoption.

Overall, the authors argue that the adoption of EMRs and IT in primary care has the potential to improve the quality of care delivered to patients and enhance patient outcomes. However, successful adoption requires careful consideration of organisational culture, resource availability, and technical expertise. By providing support and resources to primary care providers, policymakers can help to promote successful adoption of EMRs and IT, improving the delivery of primary care and ultimately enhancing the health of communities.

iii. Electronic Health Record Implementation in Developing Countries

(Adeyinka, Tirenioluwa. (2021). Electronic Health Record Implementation in Developing Countries: A Systematic Review.)

Electronic Health Records (EHRs) have become increasingly popular in developed countries, where they are widely used to store and manage patient information. However, their implementation in developing countries has been more limited, and there is a need to explore the challenges and opportunities associated with EHR implementation in these contexts. In her article "Electronic Health Record Implementation in Developing Countries: A Systematic Review," author Tirenioluwa Adeyinka explores the literature on EHR implementation in developing countries, highlighting the challenges and opportunities associated with this process.

The author conducted a systematic review of the literature on EHR implementation in developing countries, focusing on studies published between 2015 and 2020. The review revealed several key findings. First, the author notes that EHR implementation in developing countries faces a few challenges, including a lack of resources, technical expertise, and infrastructure. In addition, the author notes that cultural and organisational factors can also play a role in the success or failure of EHR implementation.

Despite these challenges, the author notes that there are also opportunities associated with EHR implementation in developing countries. For example, EHRs can improve the quality of care delivered to patients, facilitate better coordination of care, and reduce medical errors. In addition, EHRs can help to improve data collection and analysis, which can be used to inform healthcare policy and decision-making.

To address the challenges associated with EHR implementation in developing countries, the author suggests several strategies. These include increasing funding and resources for EHR implementation, providing training and technical support for healthcare providers, and working to address cultural and organisational barriers to adoption.

Overall, the author argues that EHR implementation in developing countries has the potential to improve the delivery of healthcare and enhance patient outcomes. However, successful

implementation requires careful consideration of the challenges and opportunities associated with this process, as well as the development of strategies to address these issues. By working to overcome the challenges associated with EHR implementation, policymakers and healthcare providers in developing countries can help to improve the health of their populations and enhance the overall quality of care delivered.

Conclusion

Digital health record systems have the potential to enhance healthcare delivery by improving its quality, efficiency, and safety. However, the above-mentioned articles also underscore the challenges associated with implementing such systems, including:

- Lack of resources, technical expertise, and infrastructure
- Cultural and organisational barriers
- High cost of implementation
- Data security and privacy concerns

Fortunately, the highlighted challenges can be addressed to leverage the benefits of implementing digital health records while minimising obstacles. For example, implementing a low-cost system tailored to the needs of developing countries that requires little technical proficiency to operate could be a viable approach. Additionally, policymakers, healthcare providers, and patients can collaborate to address organisational and cultural barriers.

CHAPTER THREE: ANALYSIS

3.1 Description of the system

An all-inclusive digital health record-keeping solution that makes less sensitive patient-related information freely and quickly accessible for the purposes of better healthcare provision, better decision-making (via data analytics), and securing patient data.

3.2 Information Gathering Tool

Case Studies

To learn and understand through comparisons and cross-examination, case studies were utilised to describe processes, experiences, and practices. This greatly aided in obtaining the greatest knowledge and vision for building the proposed system and being aware of any limits that might be encountered during the system life cycle through cross-examination of numerous articles.

Interviews

Conducted in-person interviews to gather comments and viewpoints on the proposed system. This worked well since I was able to gather the target stakeholders' direct feedback and learn more about what they anticipated from the system.

Some of the following questions were given to determine whether the suggested system was required in the healthcare industry:

- How much time do you take to compile reports?
- How many patients does the healthcare organisation treat on a regular basis?
- How much time is spent on managing patient records, scheduling appointments, and other administrative tasks?

- Are patient records currently being managed using paper-based methods or outdated software systems?
- Is it difficult to access patient records from multiple locations or departments within the organisation?
- Are there any financial or productivity benefits that can be realised from implementing an EHR, such as improved billing accuracy or reduced duplication of effort?
- Does the healthcare organisation have plans for expanding or adding new services that would require a more efficient way of managing patient data?
- How important is patient data security and confidentiality to the healthcare organisation?
- Does the healthcare organisation have the resources, including staff and technology infrastructure, to implement and maintain an EHR system?
- Are there any existing EHR systems in use by other healthcare organisations in the same geographic area or specialty that can be leveraged?

3.3 Data Analysis

Context Diagram

The level 0 data flow diagram (DFD) that is displayed below models a system's processes by graphically representing how data flows through it. Making a broad overview of the system without getting into great depth is a first step that can be expanded upon later.

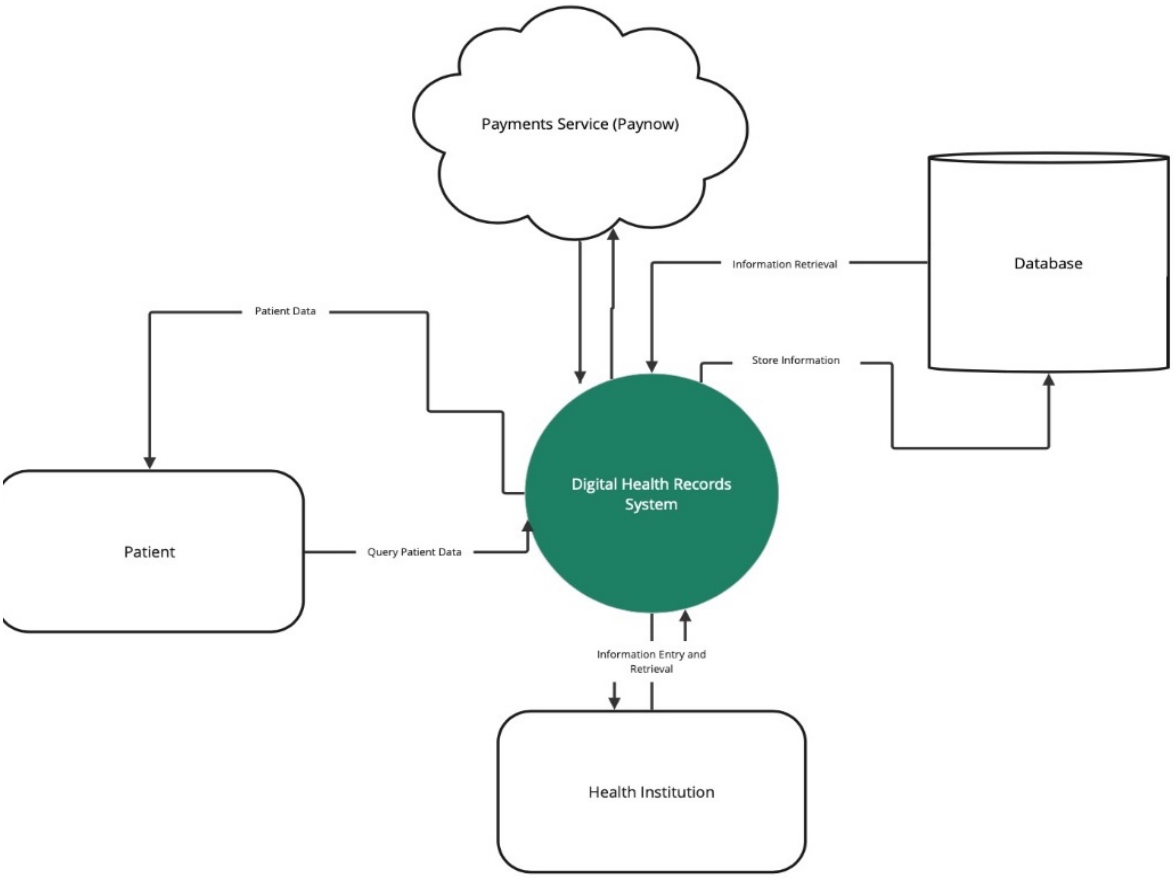

Figure 2: Context Diagram

3.4 Functional Analysis

Functional Requirements (Proposed System Use Case Diagram)

The interaction between the system's primary and secondary users and the system by itself is depicted in the use case diagram below. It also displays how well the system works.

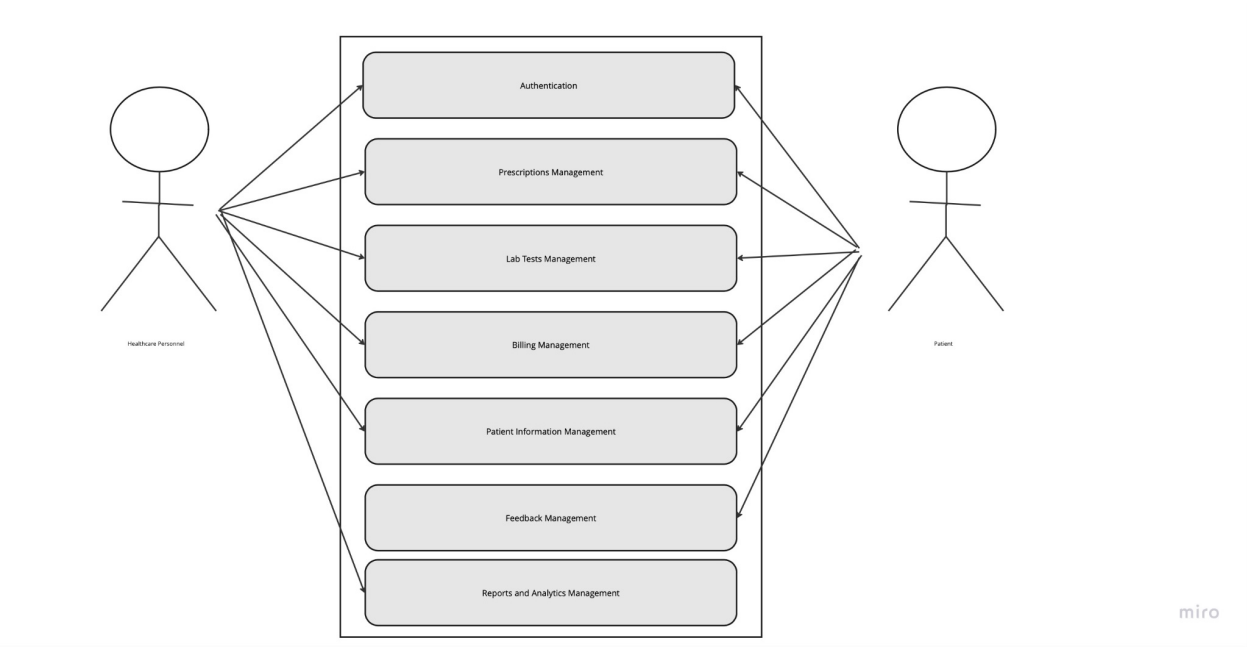

Figure 3: Use case diagram for Digital Health Records System

3.5 Non-functional Requirements

System Performance

Given that the system is based on a solid and cutting-edge technology stack that includes Flutter, Python Django, and Redis, it is anticipated to operate at a high level of performance.

Security

On the system, malware protection tools are used. The system will also implement a stringent access control policy for both primary and secondary users to safeguard data security and privacy. System restoration points and routine backup procedures are available in the event of a system failure to guarantee the security and longevity of user data.

Deliverable

Due to the allocation of time and resources to tasks in accordance with importance and priority, the system will be delivered within the specified timeline and budget.

Usability

The system will be user-friendly and easy to navigate, with clear and concise instructions for healthcare providers.

CHAPTER FOUR: DESIGN

4.1 Introduction

Purpose

The architecture and system design of the Digital Health Records System is described in this software design paper.

Scope

The software's main goals were to provide patients with a platform for organising their data and appointments as well as securely storing and accessing patient information at any time.

Overview

The system overview, architectural design, modular breakdown, and user interface design are all included in this chapter.

4.2 System Overview

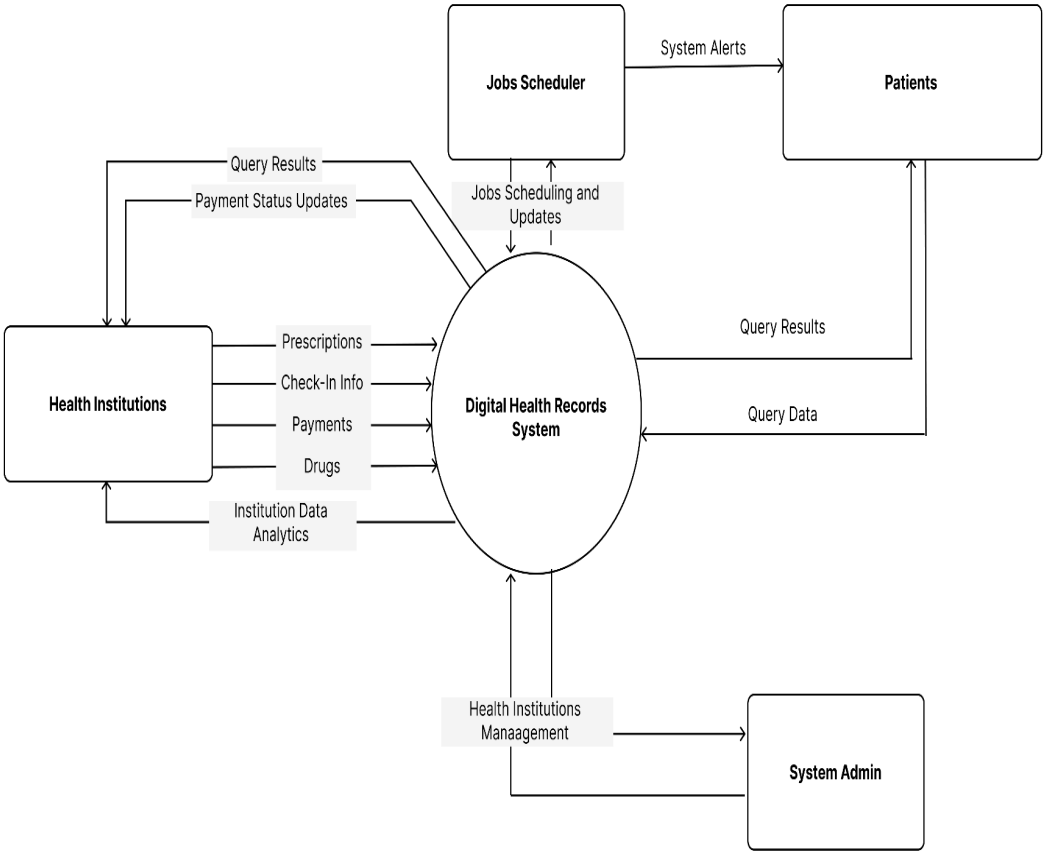

Figure 4: Level 0 DFD Digital Health Records System

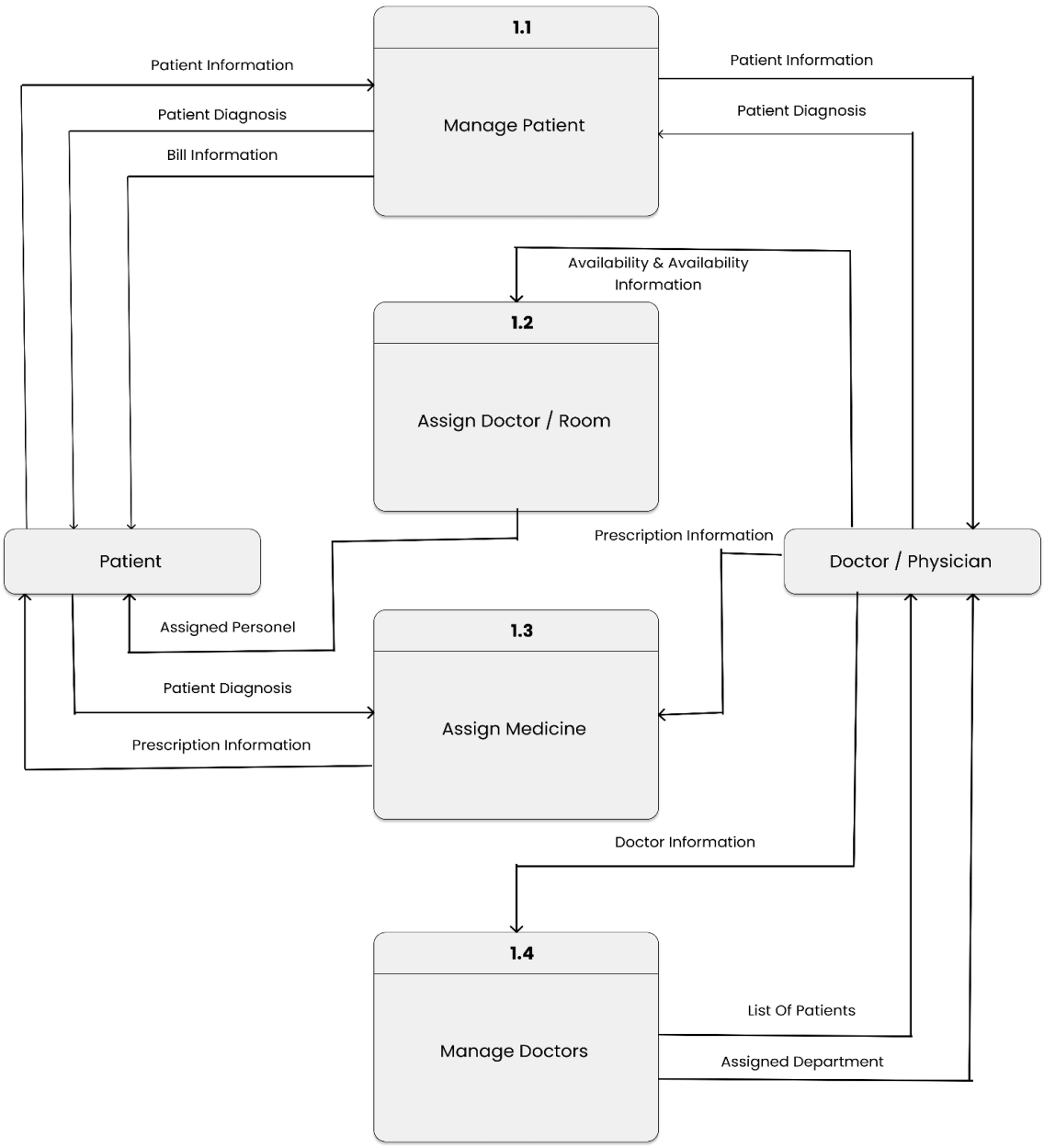

Figure 5: Level 1 DFD Digital Health Records System

4.3 Activity Diagrams

Activity Diagram for Patient Check-In

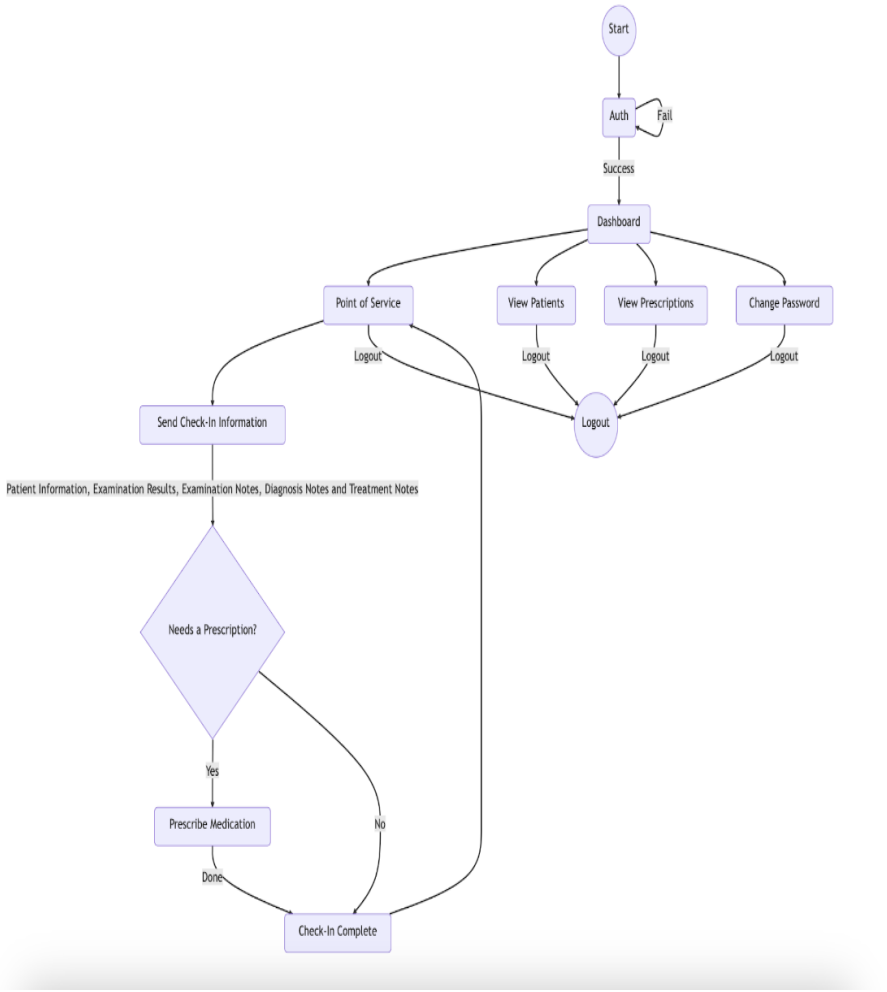

Figure 6: Patient Check-In Activity Diagram

Activity Diagram for Patient Mobile App

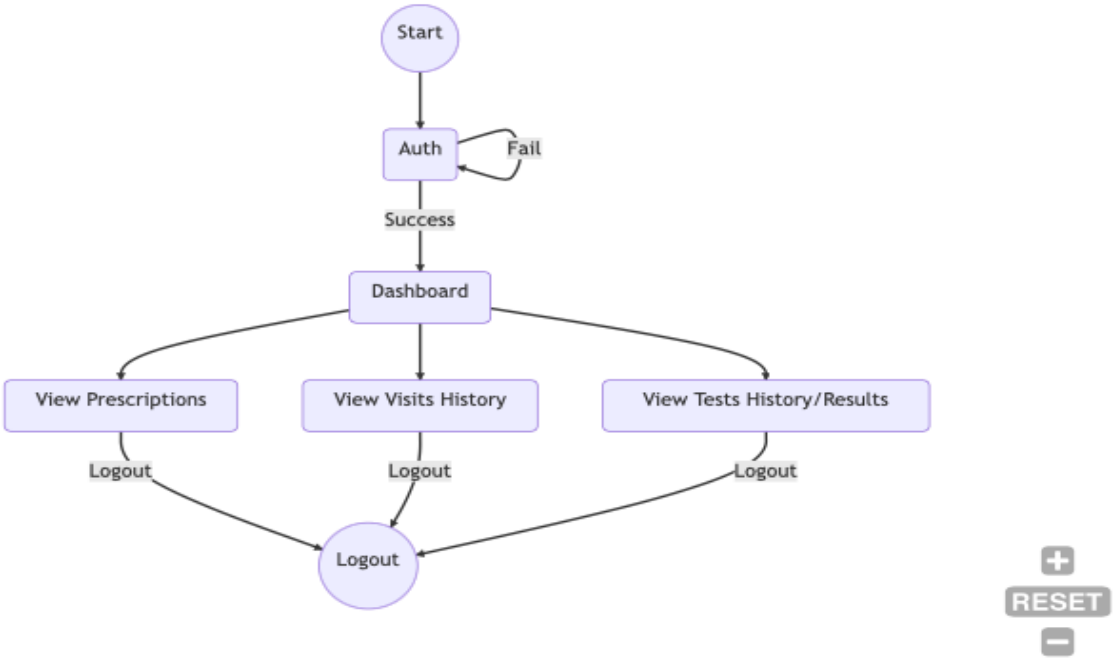

Figure 7: Patient Mobile App Activity Diagram

4.4 Sequence Diagrams

Object interactions are arranged in temporal sequence in a sequence diagram. The scenario's objects are shown, along with the series of messages that must be exchanged for the scenario to run properly.

Patient Check-In Sequence

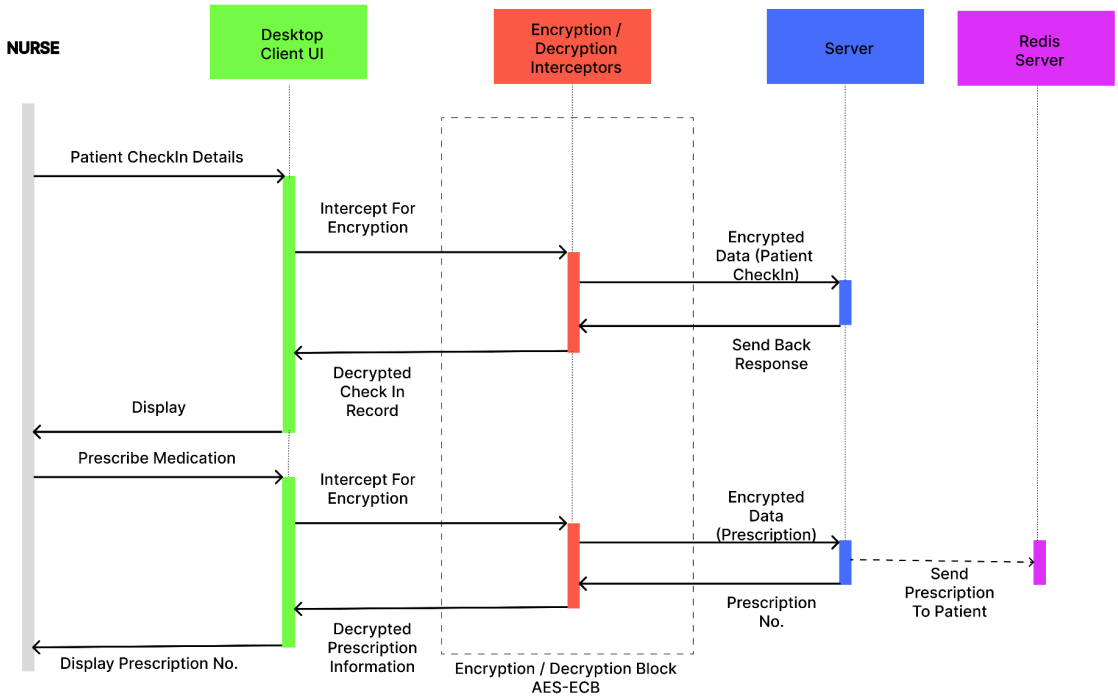

Figure 8: Patient Check-In Sequence Diagram

Dispensary Sequence

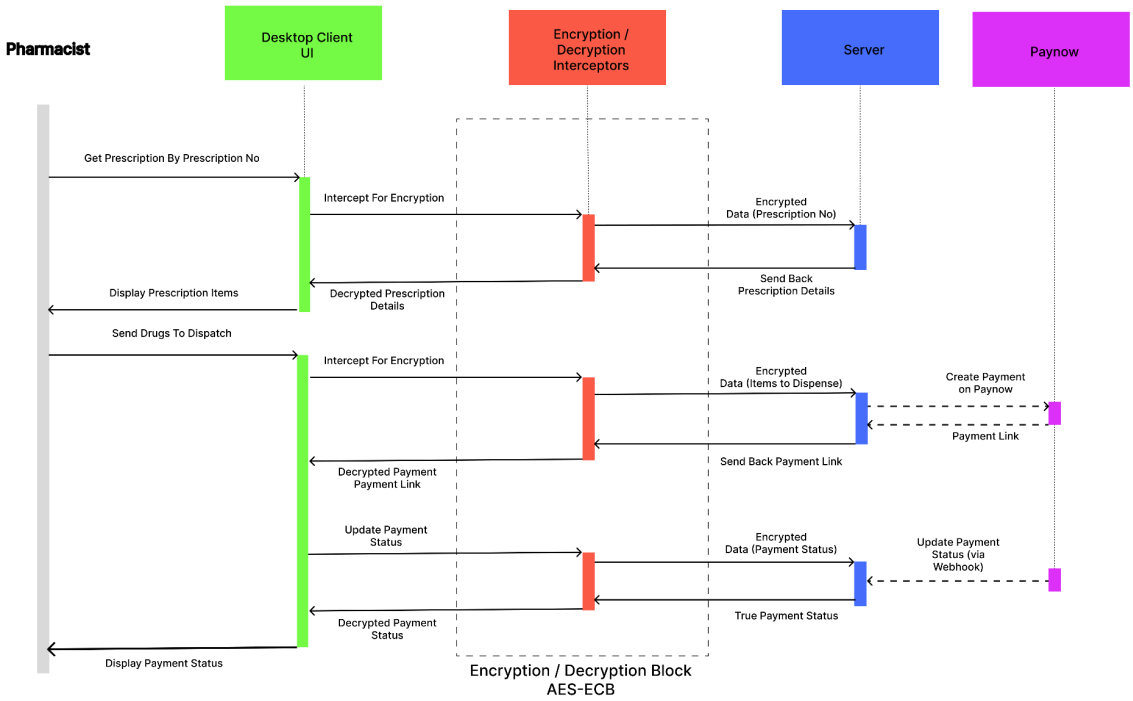

Figure 9: Dispensary Sequence Diagram

Patient Interactions Sequence

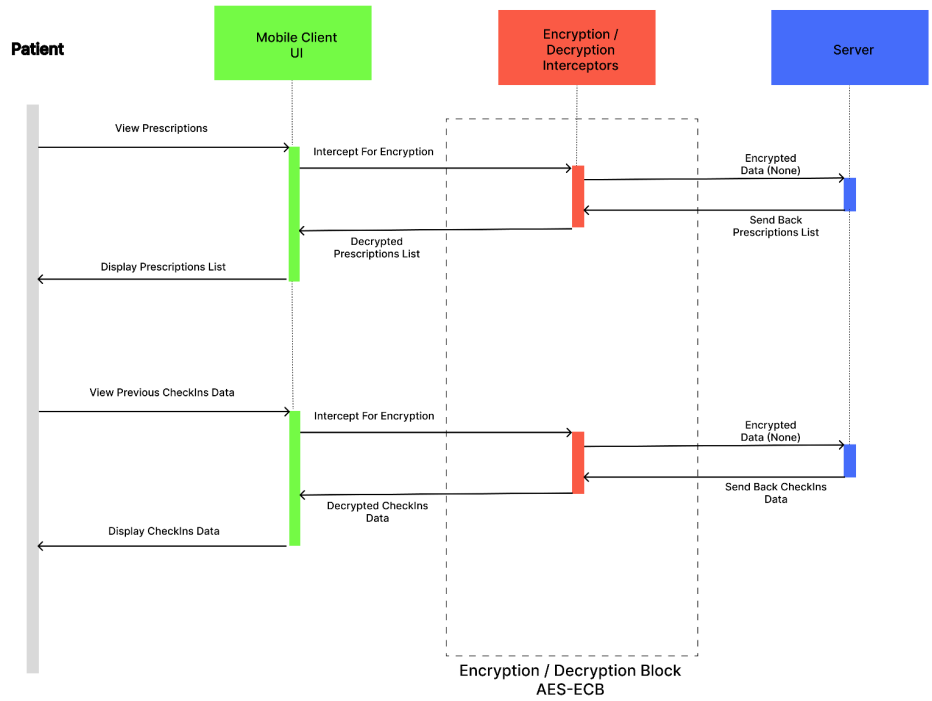

Figure 10: Patient Mobile App Sequence Diagram

4.5 Architectural Design

System Model

The REST (Representational State Transfer) architecture was utilised in a client-server format for the Digital Health Records System. A RESTful API (Application Programming Interface) reads and writes data using HTTP methods like GET, POST, PUT, and DELETE, which suggests that the client application and server application can be developed independently and separately.

Table 1: Components of the system

Name of Item	Function	Source

Backend API	Implementing business rules for handling users, patient data, payments, lab processes, and schedules is your responsibility.	In-house development
Patient Application UI	Liable for showing and rendering material that solely relates to that user	In-house development
Doctor / Clinic System UI	Oversees gathering and displaying patient data	In-house development
Database	Save patient data, such as visits, identities, fees, tests, etc.	In-house development
Queue Management	In charge of controlling the processing queue for time-consuming jobs	In-house development

Architectural Diagram

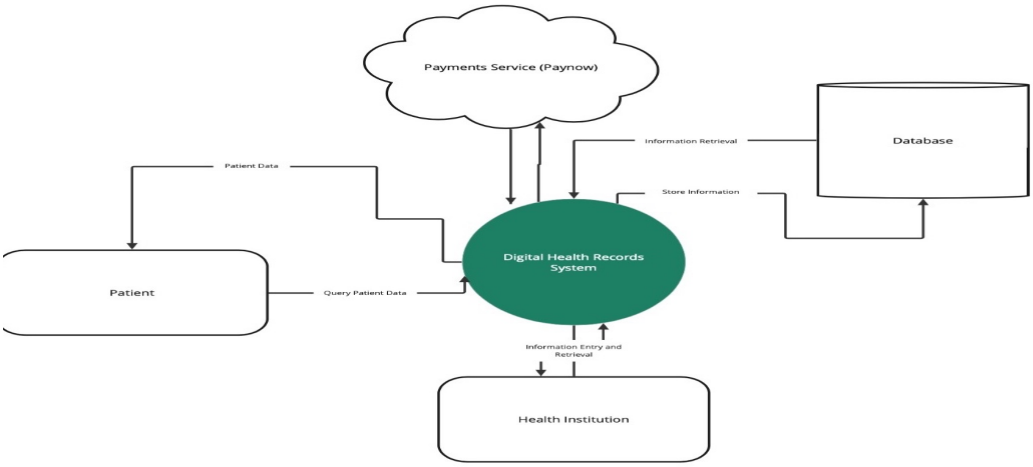

Figure 11: Architectural Diagram

4.6 Modular Decomposition

The Digital Health Records System was created using functional decomposition. It consists of 5 parts:

- Backend API
- Desktop Interface
- Mobile Application Interface
- Database
- Queue Management

4.7 Database Design

The Digital Health Records System makes use of the SQL database PostgreSQL. The selection of a SQL persistence layer enables data querying and data fusion via relationships between the data.

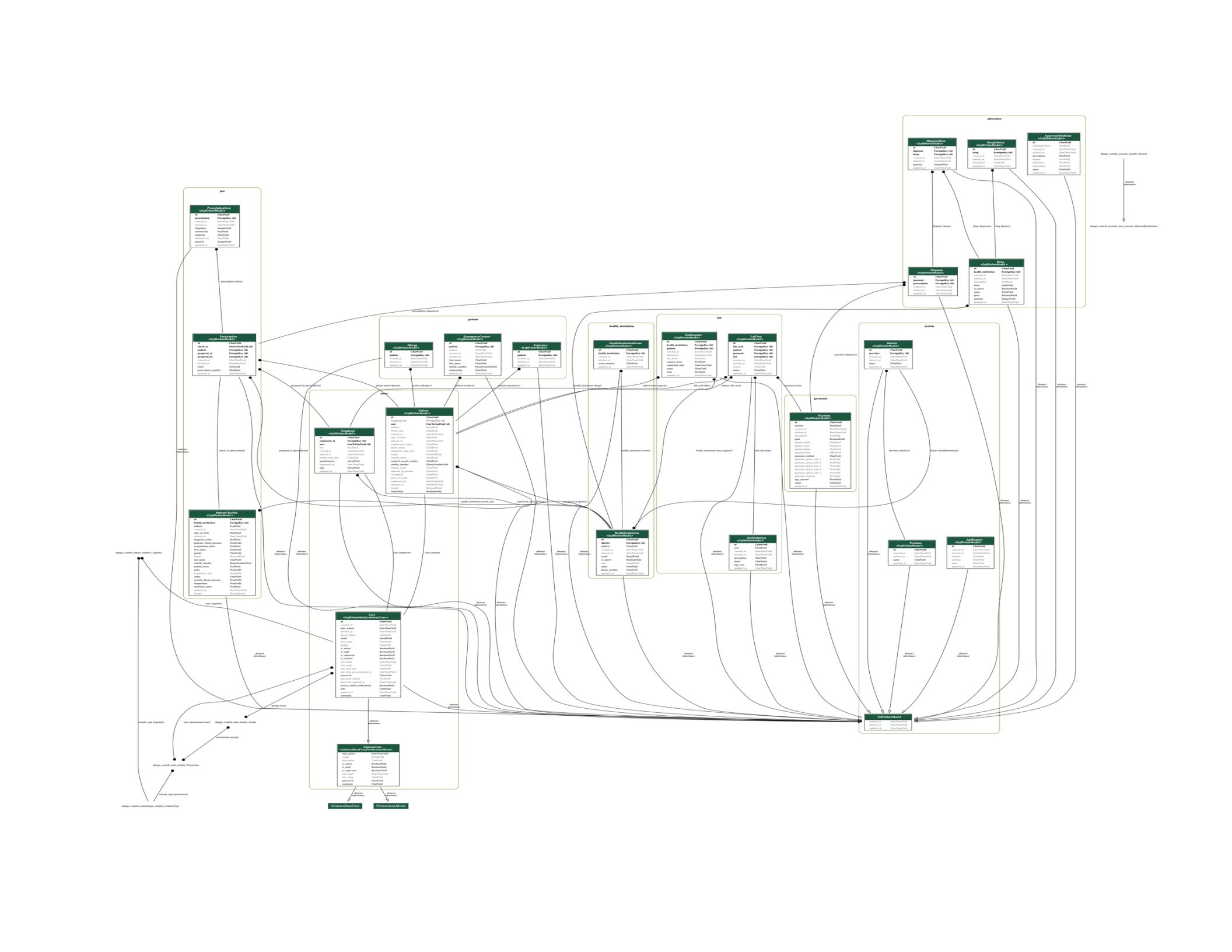
pharmacy
DispenseItem
DrugHistory
ApprovedMedicine
Dispense
Drug
gen
PrescriptionItem
Prescription
patient
Allergy
EmergencyContact
Insurance
health_institution
HealthInstitutionRoom
lab
TestRequest
LabTest
payments
Payment
system
District
Province
AgriRequest
Employee
Patient
PatientCheckIn
HealthInstitution
TestSubstition
User
AbstractUser
PermissionsMixin
SoftDeleteModel
prescription (items)
prescription (dispenses)
dispense (items)
drug (dispenses)
drug (history)
prepared_by (prescriptions)
patient (prescriptions)
patient (allergies)
patient (contacts)
patient (insurances)
health_institution (drugs)
patient (test_requests)
lab_tech (labs)
patient (lab_tests)
payment (items)
payment (dispenses)
province (districts)
district (healthinstitutions)
check_in (prescriptions)
prepared_at (prescriptions)
health_institution (rooms)
health_institution (test_requests)
test (lab_tests)
health_institution (check_ins)
registered_at (patients)
registered_at (patients)
user (employee)
user (patient)
user (logentry)
groups (user)
user_permissions (user)
permissions (group)
content_type (logentry)
content_type (permission)
abstract inheritance

Figure 12: Database Models

4.8 Overview of the User Interface

The mock-up designs that were utilised to create the system's user interface are depicted in the diagrams below. The clinic, hospital, or physician's point of view is used to demonstrate the system's capabilities.

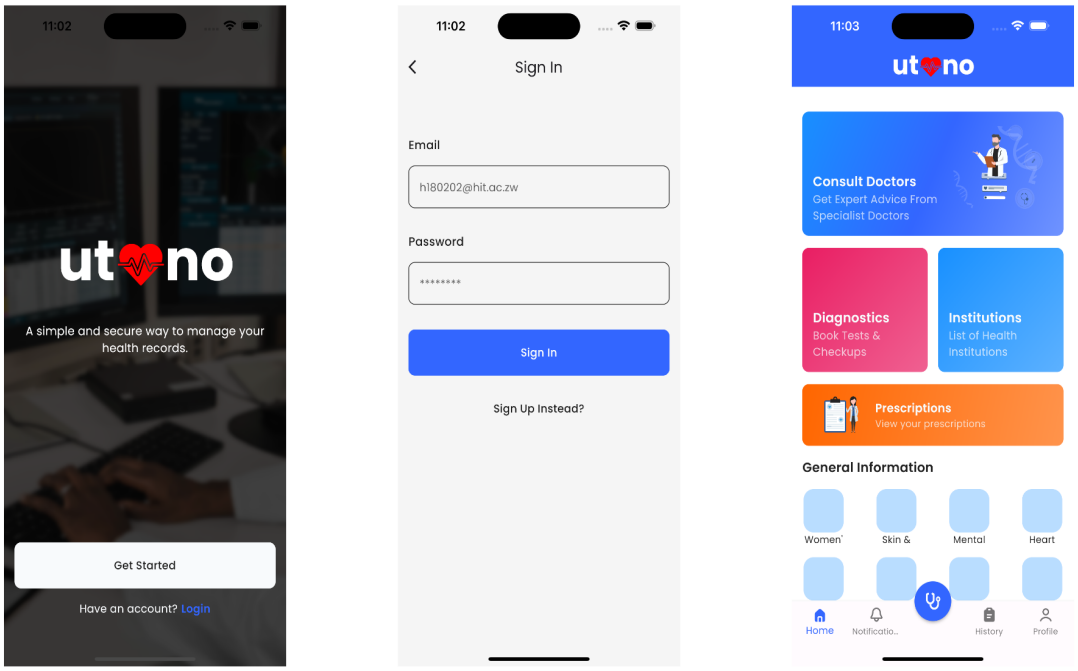

Figure 13: Mobile App Screenshots 1

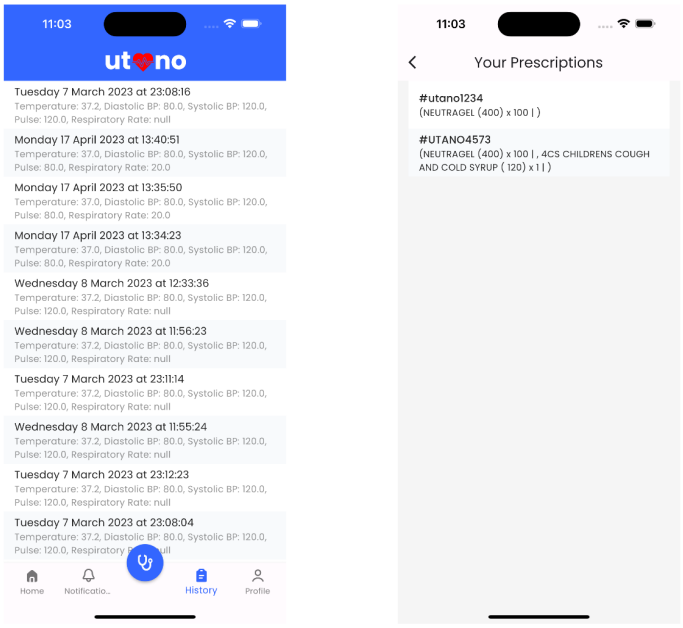

Figure 14: Mobile App Screenshots 2

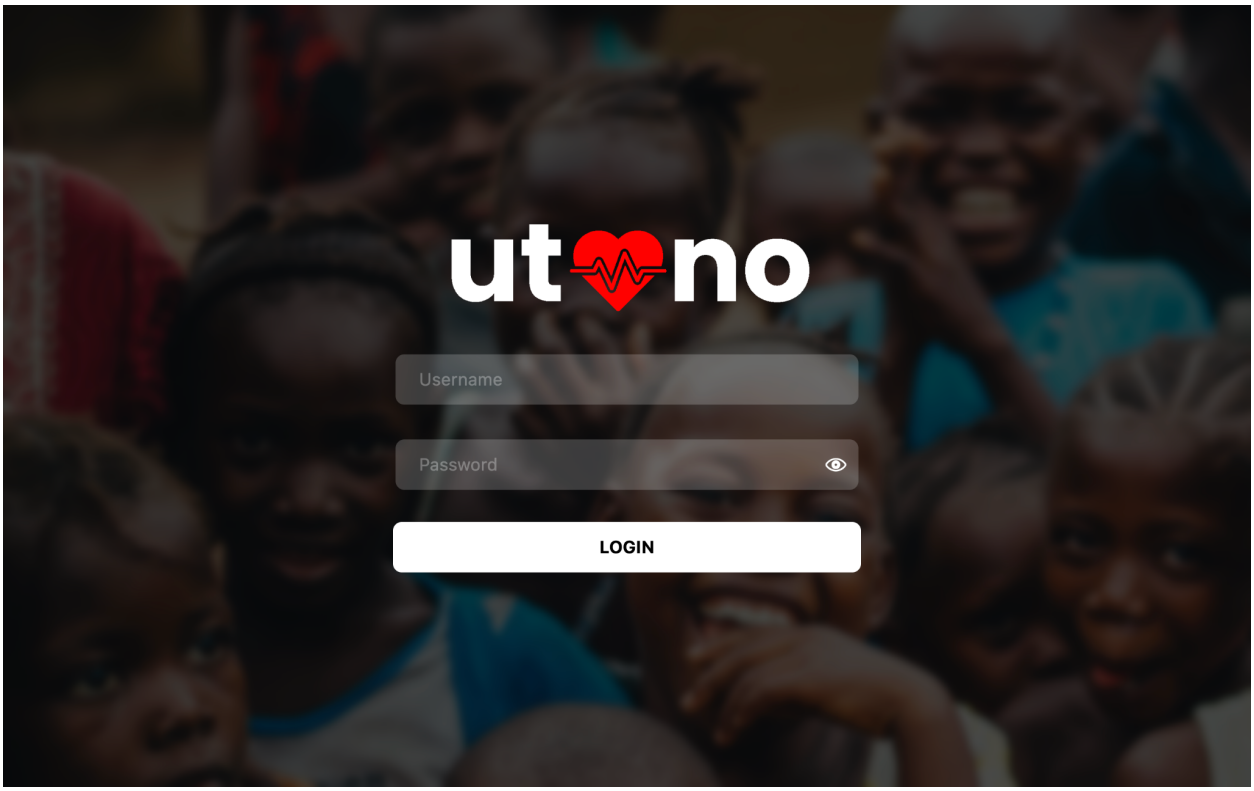

Figure 15: Desktop App Login Screen

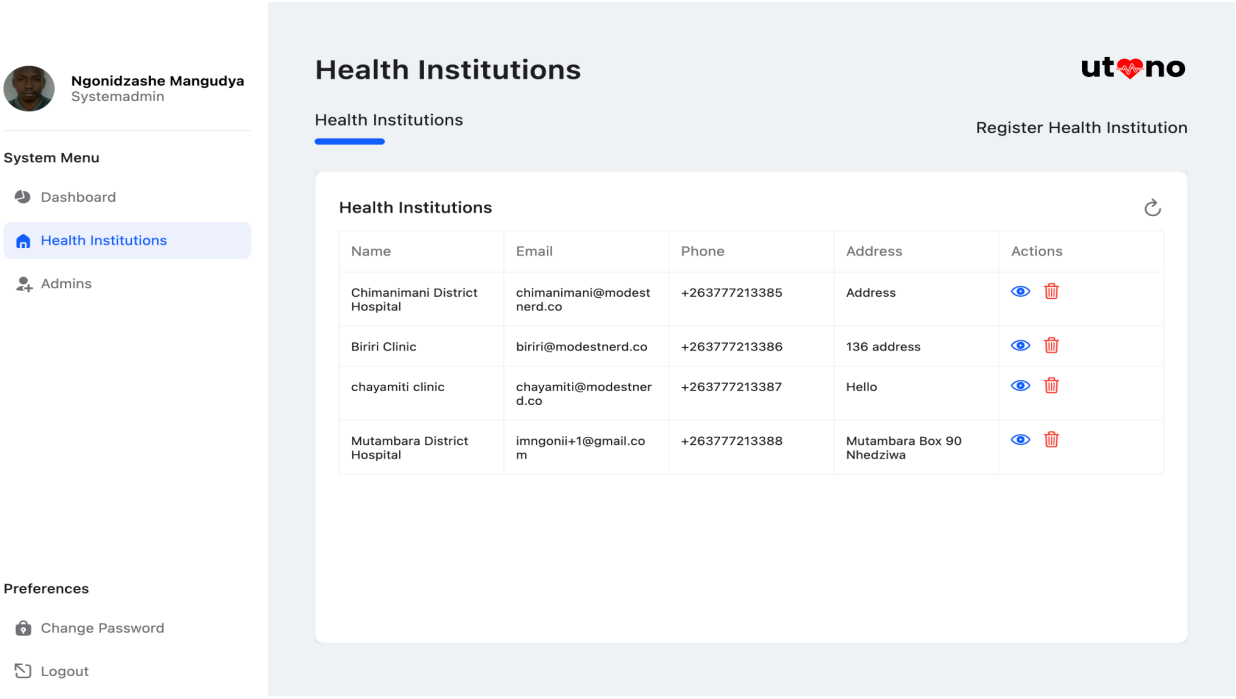

Figure 16: System Admin Management Dashboard

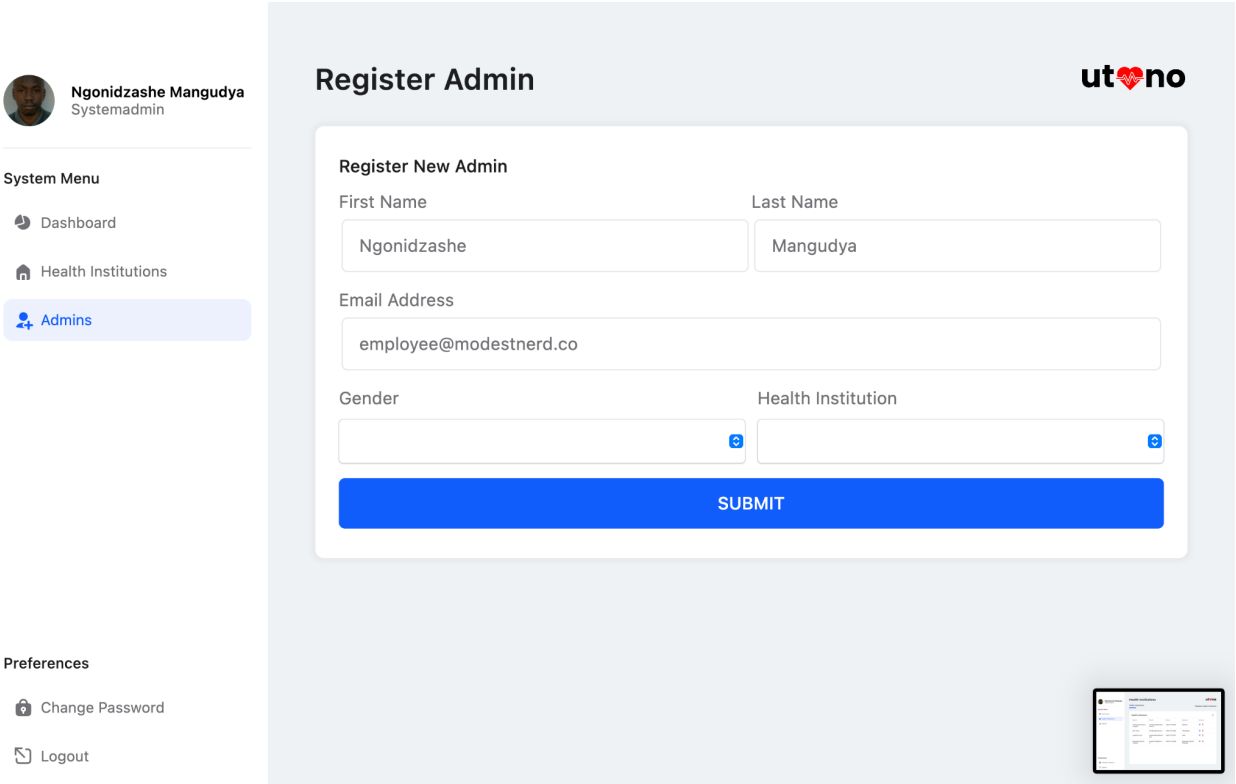

Figure 17: System Admin Register New Admins

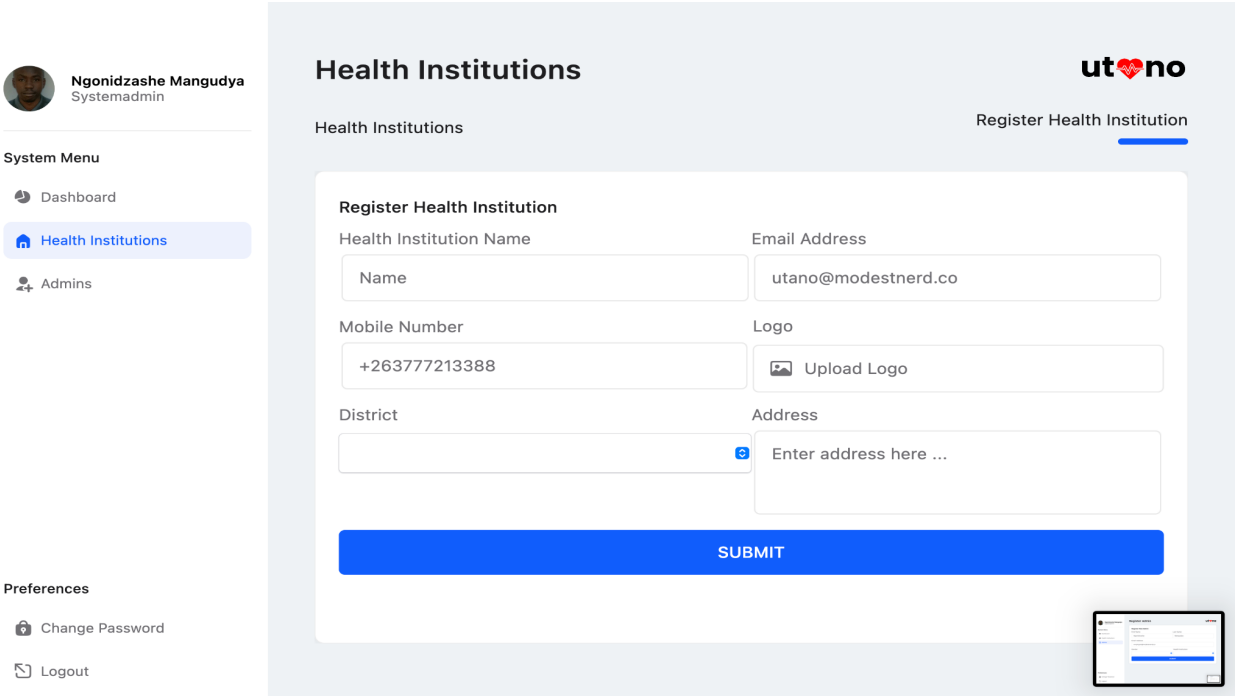

Figure 18: System Admin Register New Health Institutions

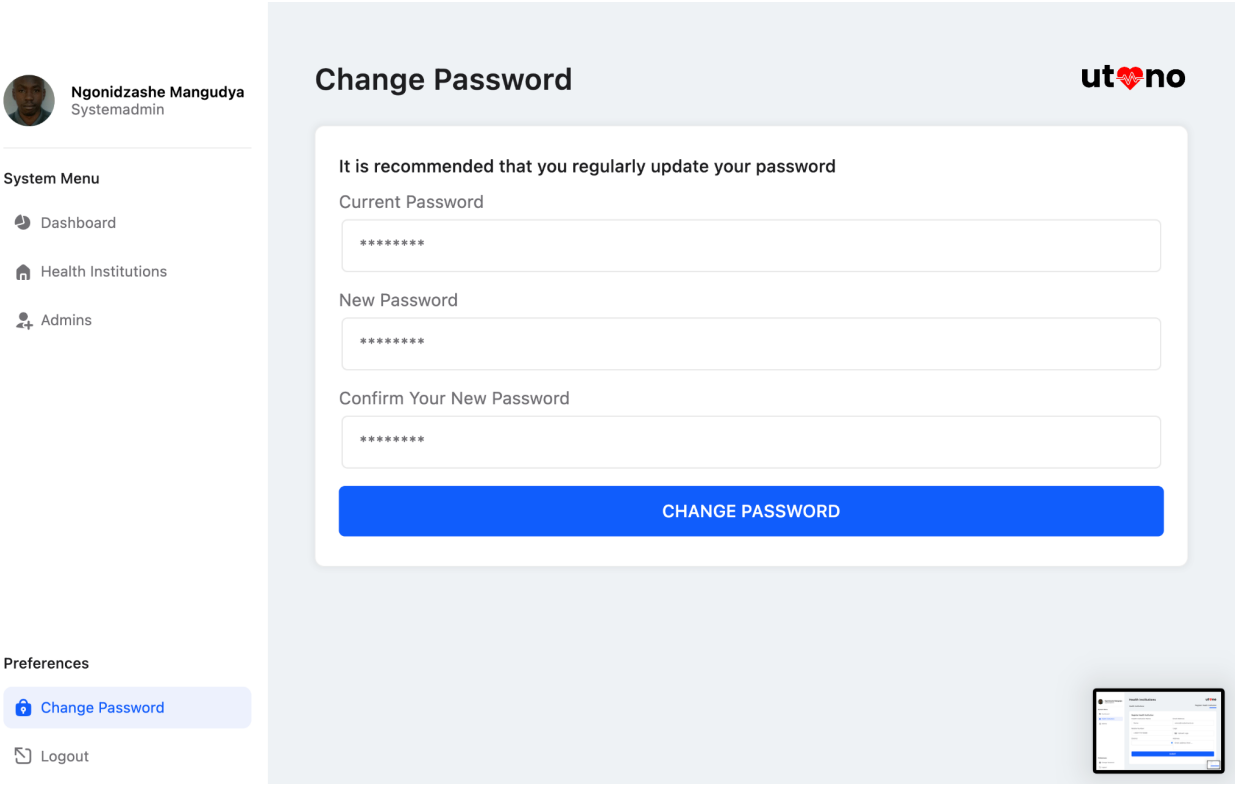

Figure 19: Change Password

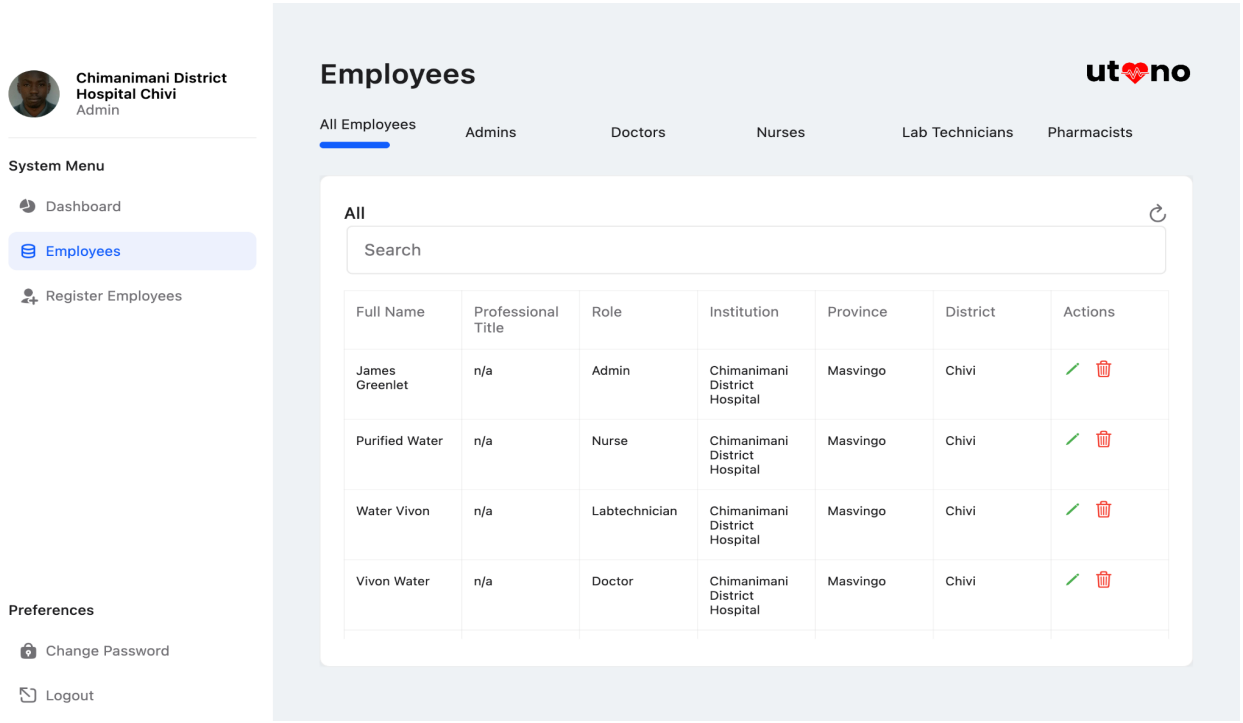

Figure 20: Health Institution Employees Management

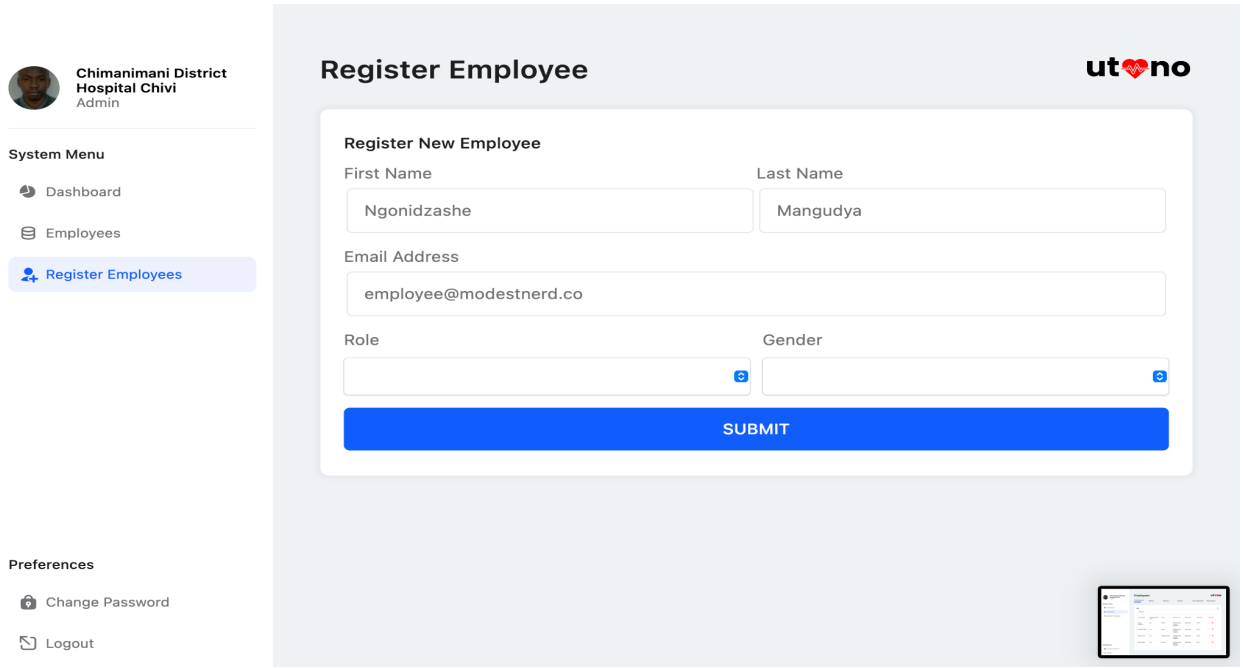

Figure 21: Health Institution Register New Employees

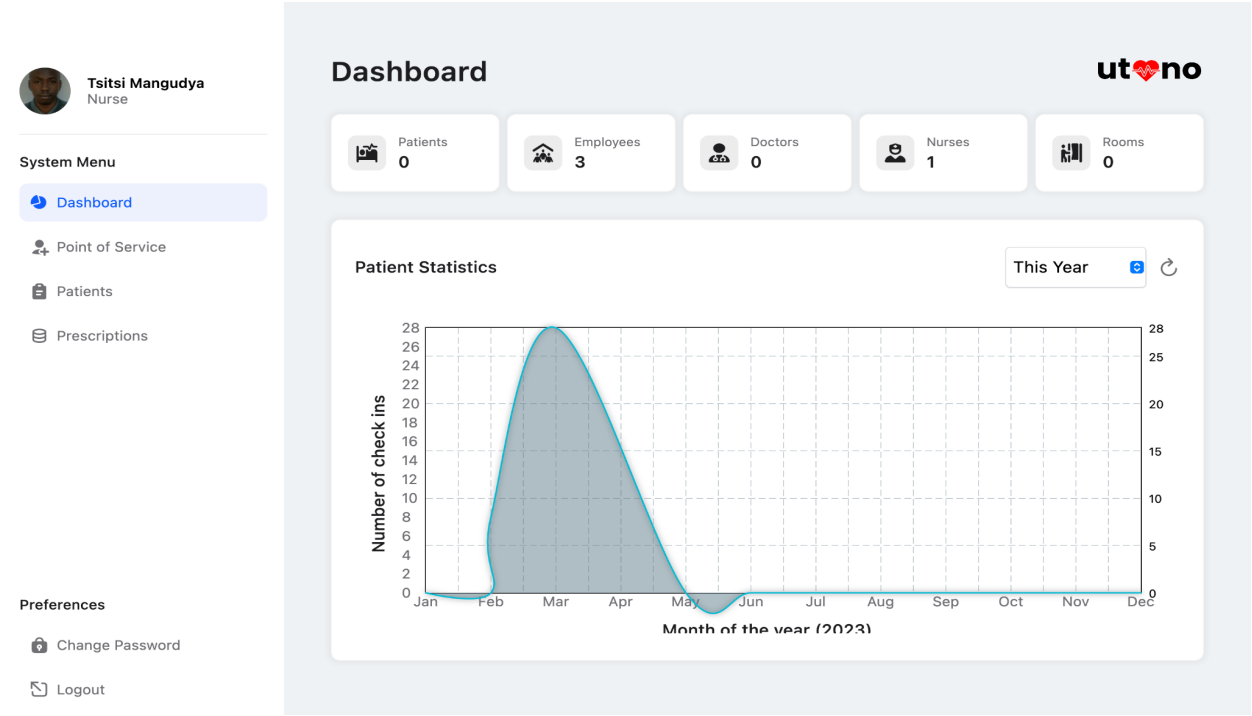

Figure 22: Nurse Dashboard

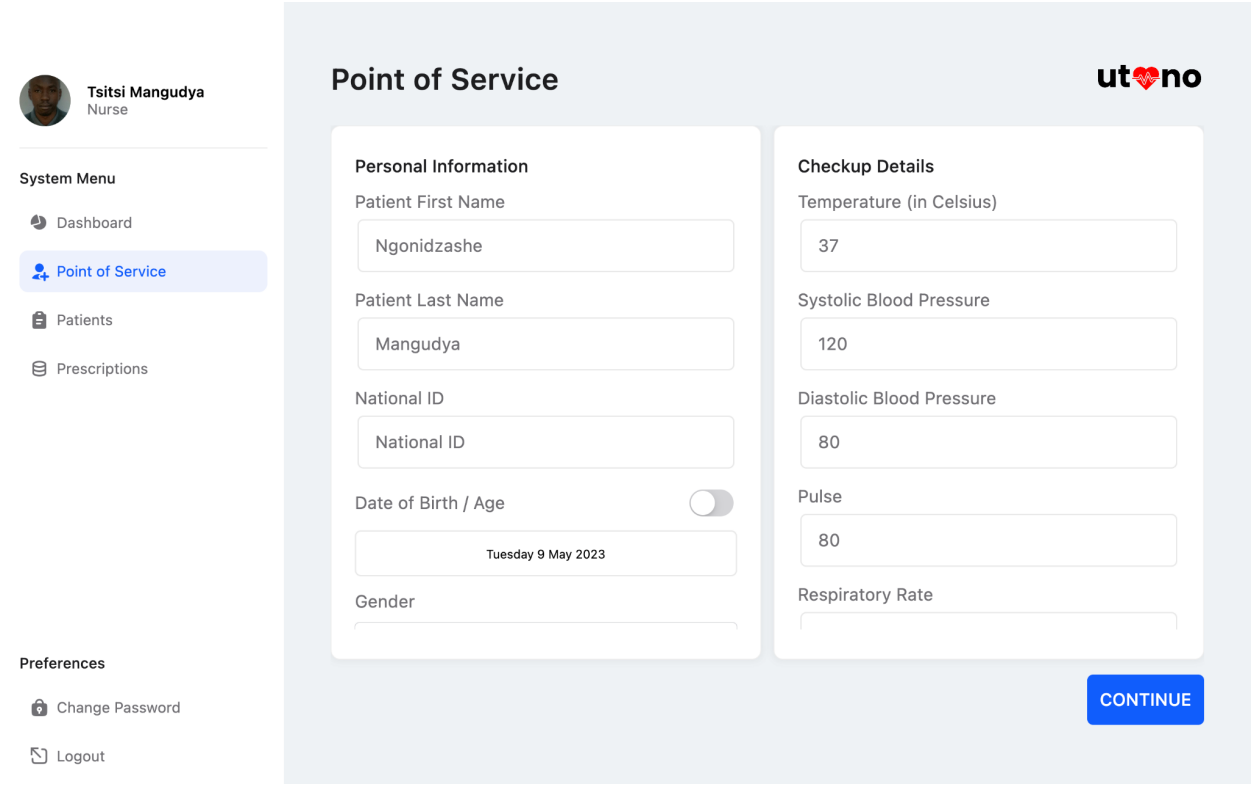

Figure 23: Nurse Point of Service

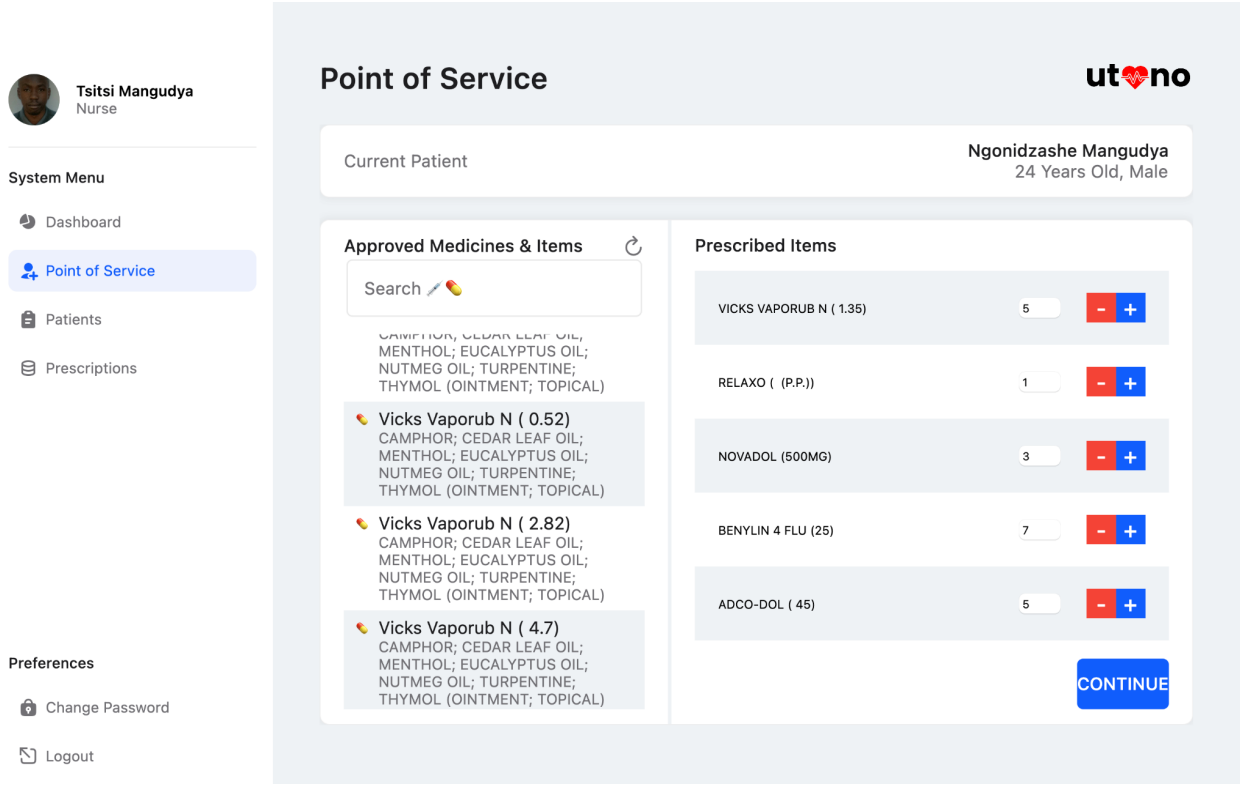

Figure 24: Nurse Point of Service Prescribe Medication

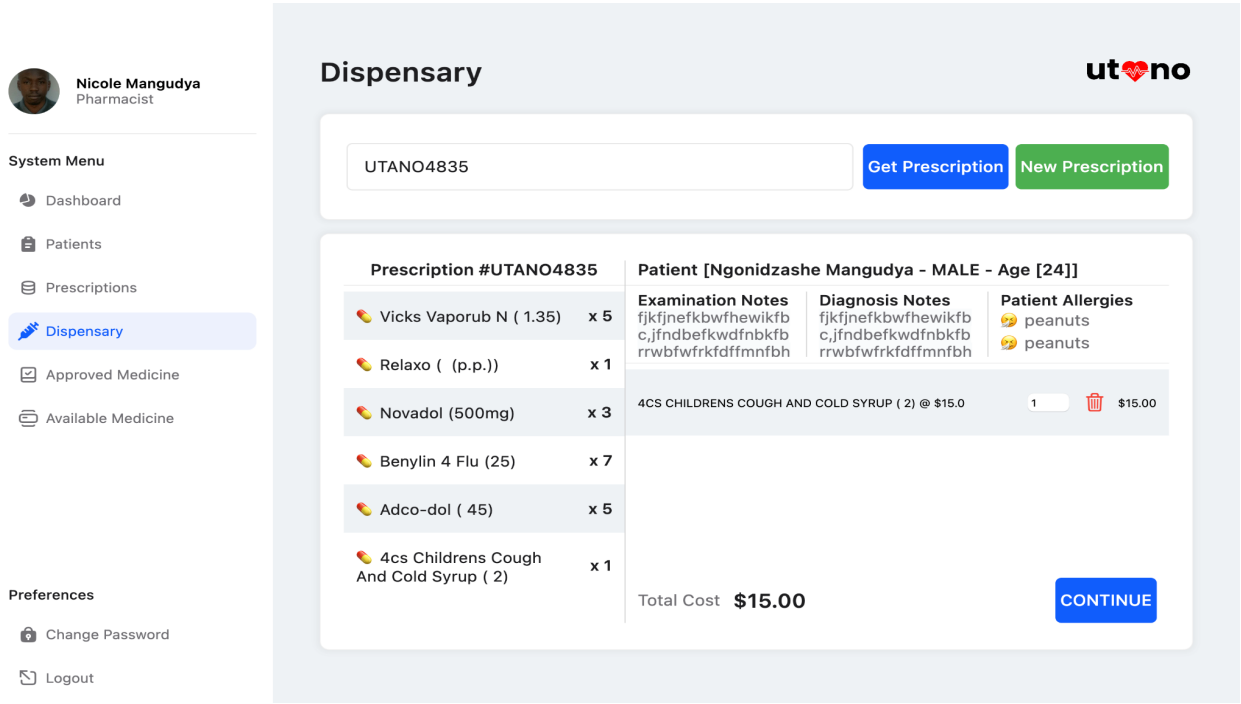

Figure 25: Pharmacist Dispensary

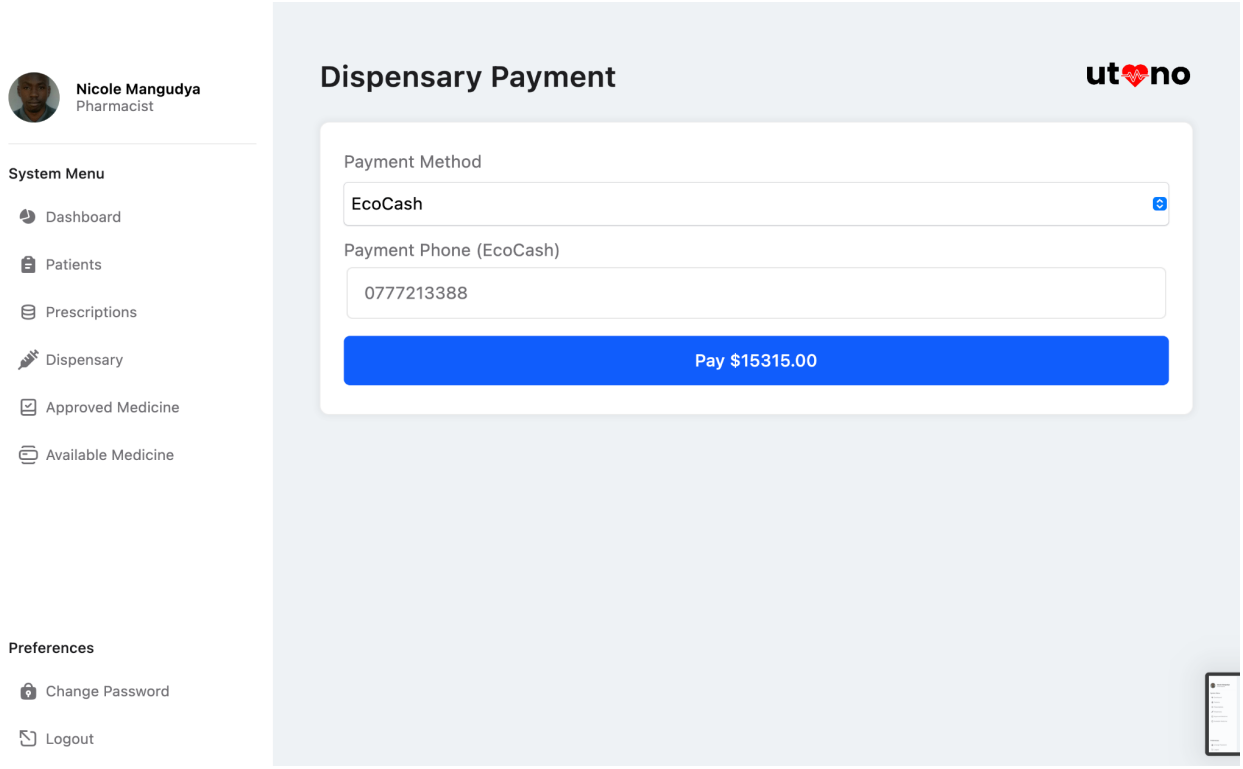

Figure 26: Dispensary Payments (using Ecocash)

Figure 27: Approved Medicines Management

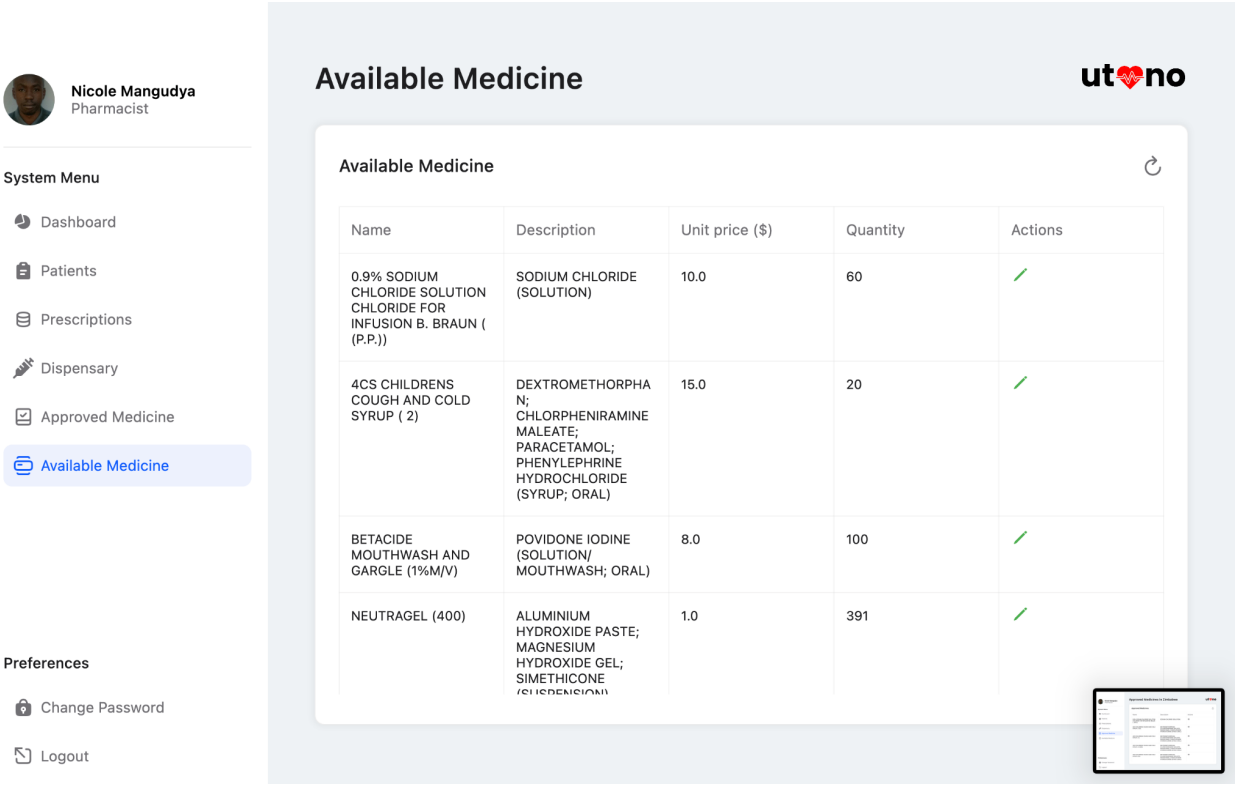

Figure 28: Health Institution Medicines Management

CHAPTER FIVE: IMPLEMENTATION AND TESTING

5.1 Coding Standards

Internal Documentation Standards

Internal documentation standards improve the readability of software source code.

A comment block at the start of a file containing one or more software modules (methods, functions, or subroutines) or a shell script file should contain the following essential details:

- Name of the file
- Name of the project
- Name of the author who created the file.
- Date the file was created.
- Licence / Copyright for that specific file

Indentation

In order to create programs that are easy to read and maintain, proper and consistent indentation is essential. Use of indentation is advised for:

- Put more emphasis on a control statement's body.
- Put more emphasis on a conditional statement's body.
- Incorporate a new scope block.

A minimum of 2 spaces should be used for indentation as to limit line length.

Classes, Subroutines, Functions and Methods

Any logic section with more than 15 lines of code should be divided into a function or method. For example, *decryptFromBase64* and *dioErrorToApplicationError* are examples of names for classes, subroutines, functions, and methods that should indicate the function they provide.

Variable Names

Variable names should be relevant and communicate the purpose for which they are used, such as "*mobileNumber*," and they should also follow the camel case rule.

Version Control

When implementing various features, version control helps prevent conflict, keeps track of code changes, and allows rollback to a previously working state when errors happen.

The version control tools to be used are git and GitHub, whilst adhering to the following order:

- Create an issue (an enhancement to the project, bug fix)
- Create a branch for the issue.
- Add changes to the branch $git\ add\ <filename>$
- Commit the new changes $git\ commit\ -m\ "<commit\ message>"$
- Push the new changes to the remote repository $git\ push\ -u\ <remote><branch\ name>"$

The GitHub repository for the project was a mono repository for all components deployed at *https://github.com/iamngoni/utano*.

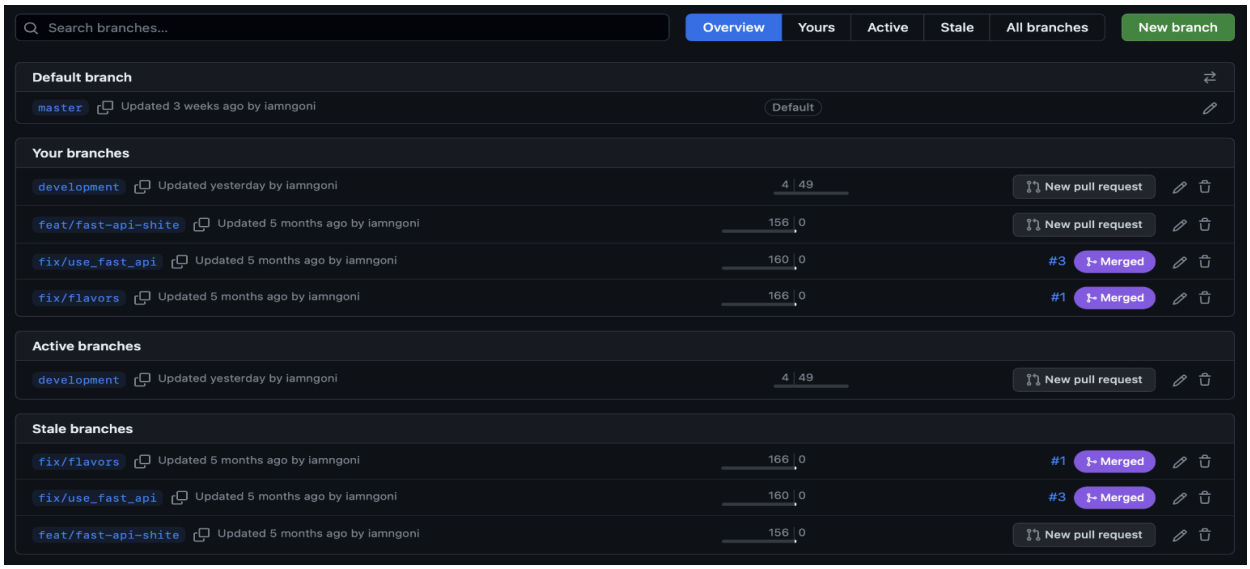

Figure 29: Project Repository Branches

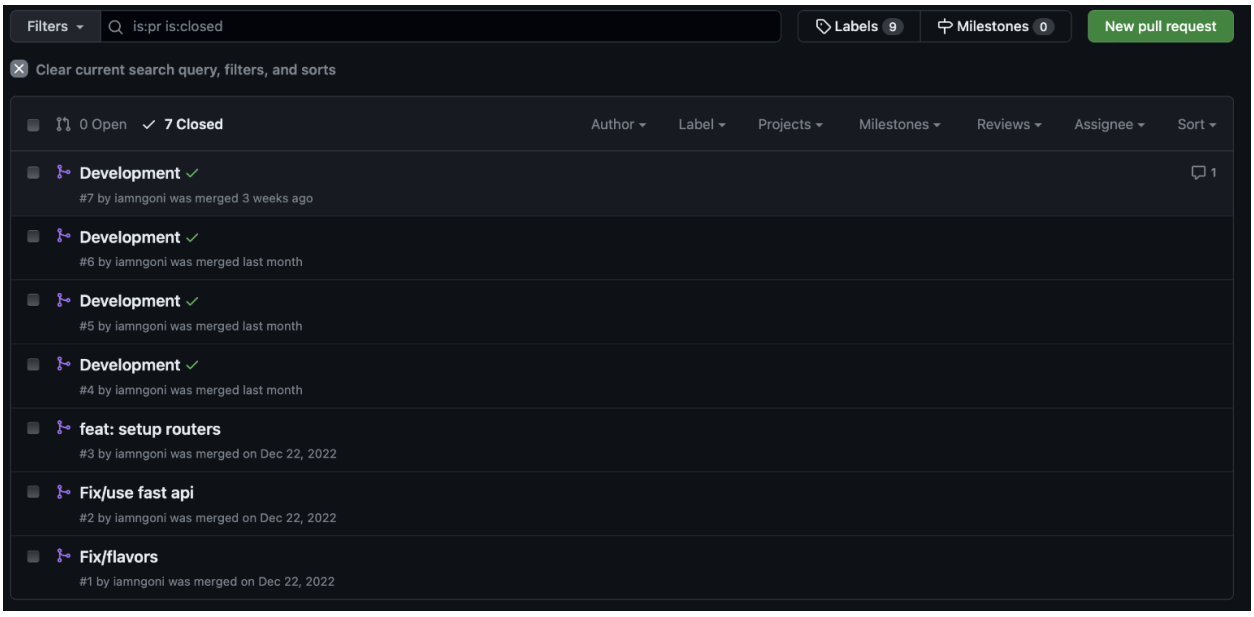

Figure 30: Project Repository Merged Changes

5.2 Implementation Strategy

Components To Be Implemented

Table 2: Components to be implements

Name of Component	**Function**	**Source**
Backend API	Accountable for carrying out business standards pertaining to patient information handling, payments and on-site analytics	In-house Development
Health Institutions Client UI (Desktop)	Responsible for rendering and requesting information (credentials, payment information, drugs, etc)	In-house Development
Patient Client UI (mobile)	Responsible for allowing the patient to interact with their health information	In-house Development
Database	PostgreSQL will store every information from the system and Redis will be used as the intermediary cache.	In-house Development

Development Approach

The development of the Digital Health Records system will follow the Agile style of developing software - specifically the feature-driven development.

Feature-driven software development (FDD) is a method that focuses on developing software features in a planned, organised, and systematic way. In FDD, the software development process is broken down into small, manageable features or modules. Each feature is designed, implemented, and tested separately before being integrated into the larger system. This approach allows for better control over the development process and makes it easier to track progress and identify issues. Overall, FDD is a structured and disciplined approach to software development that prioritises delivering functional software features in a timely and efficient manner.

It was specifically used in the development of the Backend API where each module was a separate feature and was worked on and made available, then a corresponding UI was developed for the new feature.

The method has the following advantages:

- Better control over the development process: FDD provides a clear and structured approach to software development, which helps track progress and identify issues early on.
- Faster development times: By breaking down the development process into smaller features or modules, FDD allows for faster development times as individual features can be completed and integrated into the system in a timely manner.
- Greater flexibility: FDD is a flexible methodology that can be adapted to meet new needs. It allows for changes to be made to the development process as needed, without disrupting the overall project plan.
- Better quality software: By focusing on delivering functional software features in a timely manner, FDD can help to ensure that the final product is of high quality and meets the needs of end-users.

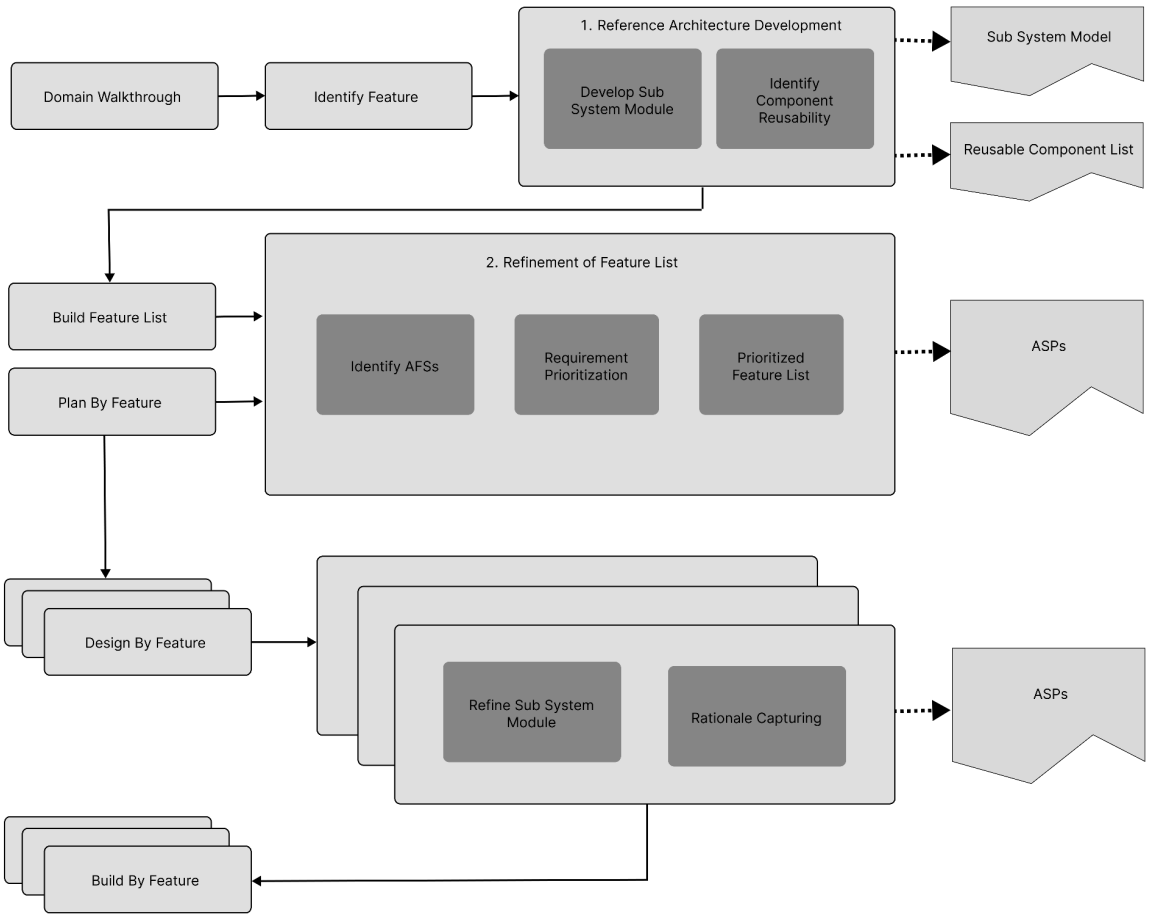

Figure 31: Feature Driven Development Methodology

5.3 Code Sample

Code Snippet: Patient Self Registration Endpoint

```
class PatientPreregistrationView(APIView):
    authentication_classes = ()
    serializer_class = PreregistrationPayloadSerializer
```

```
@transaction.atomic()
def post(self, request):
    try:
        payload = self.serializer_class(data=request.data)
        if payload.is_valid():
            logger.info("creating account for patient")
            user = User(
                email=payload.validated_data.get("email"),
                username=create_username(
                    first_name=payload.validated_data.get("first_name"),
                    last_name=payload.validated_data.get("last_name"),
                ),
                first_name=payload.validated_data.get("first_name"),
                last_name=payload.validated_data.get("last_name"),
                role=UserRoles.PATIENT,
                gender=payload.validated_data.get("gender"),
            )
            user.set_password(payload.validated_data.get("password"))
            user.save()
            logger.success("user account created successfully")

            # create patient profile
            logger.info("creating patient profile")
            medical_record_number = generate_medical_record_number()
            patient = Patient(
                user=user,
                medical_record_number=medical_record_number,
                mobile_number=payload.validated_data.get("mobile_number"),
                date_of_birth=payload.validated_data.get("date_of_birth"),
                marital_status=payload.validated_data.get("marital_status"),
```

```
            national_id_number=payload.validated_data.get("national_id_number"),
            address=payload.validated_data.get("address"),
            employment_status=payload.validated_data.get("employment_status"),
        )
        patient.save()

        # welcome user and send verification code
        send_welcome_note_to_patient.delay(user)
        send_verification_code_to_user.delay(user)

        jwt_payload = generate_jwt_payload(user)

        access_token = jwt.encode(
            payload=jwt_payload["access"],
            key=config("JWT_SECRET"),
            algorithm="HS256",
        )
        refresh_token = jwt.encode(
            payload=jwt_payload["refresh"],
            key=config("JWT_SECRET"),
            algorithm="HS256",
        )

        return ApiResponse(
            data={
                "access_token": access_token,
                "refresh_token": refresh_token,
                "user": UserModelSerializer(user).data,
            },
        )
    else:
```

```
            logger.error(payload.errors)
            return ApiResponse(
                num_status=400, bool_status=False, issues=payload.errors
            )
        except Exception as exc:
            logger.error(exc)
            return ApiResponse(num_status=500, bool_status=False)
```

Code Snippet: Health Institution Registration Notification

```
@job("health_institution")
def notify_health_institution_on_registration(
    health_institution: HealthInstitution, password: str
):
    html_content = render_to_string(
        "notification.html",
        {
            "message": f"Good day, {health_institution.name}.\n\nYou've been registered as a healthcare provider on"
            f" Utano EHR and an admin account with the following details has been registered for you:\n\n"
            f"Email: {health_institution.email}\nPassword: {password}",
            "email": health_institution.email,
        },
    )
    send_email_alt.delay(
        email=health_institution.email,
        html_content=html_content,
        email_subject="Utano EHR Registration",
    )
```

```

**Code Snippet: Point of Service Information Collection**

```
class PatientCheckInView(APIView):
    permission_classes = (IsAuthenticated, IsEmployee)
    serializer_class = PatientCheckInPayloadSerializer

    @transaction.atomic()
    def post(self, request):
        try:
            payload = self.serializer_class(data=request.data)
            if payload.is_valid():
                logger.info("saving check in information")
                check_in = PatientCheckIn(
                    health_institution=request.user.employee.registered_at,
                    **payload.validated_data,
)
                check_in.save()
                logger.success("check in information saved")

                # look for previous check ins and set them as dismissed
                previous_check_ins = PatientCheckIn.objects.filter(
                    mobile_number=check_in.mobile_number
).exclude(id=check_in.id)

                for check_in in previous_check_ins:
                    check_in.status = CheckInStatus.DISMISSED
                    check_in.save()

                if check_in.patient is None:
```
```

```
logger.info(
    "information not related to any patient... creating a new patient profile"
)
# create patient and user
username = create_username(
    first_name=check_in.first_name, last_name=check_in.last_name
)
logger.success("username generated")
password = generate_random_password()
logger.success("password generated")

email = f"{check_in.get_mobile_number}@utano.modestnerd.co"
logger.success("custom email generated")

user = User(
    first_name=check_in.first_name,
    last_name=check_in.last_name,
    username=username,
    gender=check_in.gender,
    email=email,
    role=UserRoles.PATIENT,
)
user.set_password(password)
user.save()
logger.success("user account created")

logger.info("generating new mrn")
mrn = generate_medical_record_number()
logger.info("creating patient profile")
patient = Patient(
    user=user,
```

```
                medical_record_number=mrn,
                date_of_birth=check_in.date_of_birth,
                address=check_in.address,
                mobile_number=check_in.mobile_number,
                registered_at=request.user.employee.registered_at,
                height=check_in.height,
                weight=check_in.weight,
            )
            patient.save()
            logger.success("patient profile successfully created")

            # welcome user and send verification code
            send_welcome_note_to_patient.delay(user)
            send_verification_code_to_user.delay(user)
        else:
            logger.info("information relates to an existing patient")

        return ApiResponse(
            data={"check_in": PatientCheckInModelSerializer(check_in).data}
        )
    else:
        logger.error(payload.errors)
        return ApiResponse(
            num_status=400, bool_status=False, issues=payload.errors
        )
except Exception as exc:
    logger.error(exc)
    return ApiResponse(num_status=500, bool_status=False)
```

Code Snippet: Payloads AES-ECB Encryption and Decryption

```
def encrypt(raw):
    logger.info("Encrypting data")
    raw = pad(raw.encode(), 16)
    cipher = AES.new(key.encode("utf-8"), AES.MODE_ECB)
    return base64.b64encode(cipher.encrypt(raw)).decode("utf-8")


def decrypt(enc):
    logger.info("Decrypting data")
    enc = base64.b64decode(enc)
    cipher = AES.new(key.encode("utf-8"), AES.MODE_ECB)
    return unpad(cipher.decrypt(enc), 16)
```

Code Snippet: Payment Processing for Dispensary Items with Paynow

```
def process_dispensary_payment(self, dispense: Dispense) -> bool:
        """
        This method creates a payment on paynow
        :return:
        """

        if self.payment.payment_method not in [
            PaymentMethod.ECOCASH,
            PaymentMethod.ONEMONEY,
        ]:
            logger.info("Payment method not supported for paynow")
            return True

        payment = paynow.create_payment(
```

```
    reference=self.payment.id,
    auth_email=self.payment.patient_email
    if self.payment.patient_email
    else "h180202m@hit.ac.zw",
)
logger.info("Payment created on paynow")

payment.add(
    f"Dispensary Payment: {dispense.id}",
    dispense.total_rtgs_cost,
)
logger.info("Payment item added to paynow")

response = paynow.send_mobile(
    payment, self.payment.patient_phone, self.payment.payment_method.lower()
)

if response.success:
    logger.info("Payment created successfully")
    logger.info(vars(response))
    self.payment.provider_response = str(vars(response))
    self.payment.payment_option_info_1 = response.poll_url
    self.payment.payment_option_info_2 = response.instruction
    self.payment.payment_link = response.poll_url
    self.payment.save()
    return True
else:
    logger.error("Payment creation failed")
    logger.error(response)
    # print all attribute of response
    logger.error(vars(response))
```

```
        self.payment.provider_response = str(response)
        self.payment.save()
        return False
```

5.4 Software Test Plan

The goal of this exercise is to identify flaws and faults in the Digital Health Records system (i.e., the Desktop Application, the Backend API, and Mobile Application). These flaws must be fixed in order to enhance usability, performance, robustness, and UX.

Features Tested

- Authentication (through email or phone number)
- Creation of accounts of all types
- Registration of Health Institutions
- Sending of emails
- Sending of text messages
- Record patient information on check in
- Payment collection for services

Features Not Tested

There are implemented features and items that won't be included in the present testing process. The following features are only a few of them:

- Dispatchment of drugs to patients
- Admission at health institution

5.5 Testing Strategy

Integration testing and manual testing are the main testing techniques. The ability to log into different system components is an example of a test case that has a similar feature technique. To make sure that each scenario is handled appropriately, each test case must test the functionality using both valid and invalid data.
Integration testing will be done on outcomes and functionality that may be accessed through HTTP or REST calls.

For features that need outcomes to be visually validated, such as collecting payments, manual testing will be done.

Item Pass/Fail Criteria

Implemented features/items are deemed to have passed if they accomplish the user's goals with minimal difficulty. The user's capacity to operate the feature with little to no training serves as the criterion for difficulty in this context.

Major flaws in features will cause them to fail the testing process.

5.6 Test Case Specification

Table 3: Test Case Specification

Test Case	Objective	Inputs	Expected Output	Actual Output	Comment
1.1	Authentication (through email or phone number)	username=h180202m@hit.ac.zw password=123456@Me	Success and redirect to dashboard for specific	Success and redirect to dashboard for specific role	Passed

			role		
1.2	Authentication (through email or phone number)	username=0777111222 password=123456@Me	Error: Invalid username or password error	Error: Invalid username or password error	Passed
2.1	Creation of accounts of all types (different roles)	first_name=Ngonidzashe last_name=Mangudya role=DRIVER email=h180202m@hit.ac.zw gender=MALE	Error and employee is not registered in the system	Error is not registered with error being: Specified role not recognized	Passed
2.2	Creation of accounts of all types (different roles)	first_name=Ngonidzashe last_name=Mangudya role=NURSE email=h180202m@hit.ac.zw gender=MALE	Success and employee is registered under that health institution	Success and employee is registered under that health institution	Passed
3.1	Database Connectivity	Start the server (online)	Health check with status 200	Health check with status 200	Passed

3.2	Database Connectivity	Start the server (offline)	Health check with status 500	Health check with status 500	Passed
4.1	Patient Check In	Provide all the required details	Success and redirect to prescriptions page	Success and redirect to prescriptions page	Passed
4.2	Patient Check In	Provide details missing temperature readings	Error with visual cues to signify the missing required fields	Error with visual cues to signify the missing required fields	Passed
5.1	Collect payments through Ecocash	Provide paying mobile phone number	Success i.e. the supplied phone number is used instead of the existing one in records	Success i.e. the supplied phone number is used instead of the existing one in records	Passed
5.2	Collect payments through Ecocash	Paying mobile phone number is not provided	Success i.e. the system manages to make use of the phone number already	Success i.e. the system manages to make use of the phone number already saved	Passed

			saved against the patient's health record	against the patient's health record	

5.7 Test Reports

The results of the tests are described in this section. Here, incidents, flaws, and necessary modifications will be formally presented.

Incidents

No issues were discovered during testing, and all described tests were passed.

Defects

No defects were discovered during testing, and all described tests were passed.

5.8 Deployment

To deploy the Digital Health Records system the following commands may be used for each platform:

Desktop Client

Run ***cd dhra*** to navigate into directory.

Run ***flutter pub get*** to install dependencies.

Run ***flutter build macos*** to get the executable application for MacOS or run ***flutter run -d macos*** to run the system as is.

Mobile Application

Run cd utano_patient

Run flutter pub get to install dependencies.

Run ***flutter build appbundle*** to get the build files deployable on the Google Play Store

Run ***flutter build ios*** to get the build files deployable on the App Store

Backend API

Run ***cd dhra_api***

Run ***source env/bin/activate*** to enable a virtual environment for that specific project.

Run ***pip install -r requirements.txt*** to install the dependencies required by the project.

Run ***bash start_app.sh*** to start the main server.

Run ***bash start_rq.sh*** to start the Redis queue handler.

CHAPTER SIX: CONCLUSION AND RECOMMENDATIONS

6.1 Results

These were the results at the time the Digital Health Records system was implemented and tested:

- Authenticating into the system via the API took an average of 250 milliseconds and through the UI platforms an average of 2 seconds.
- Patients check in and prescription modelling was a flawless process split into simple and clear steps.
- Email and SMS notifications were sent for most of the activities within the system.
- The system could handle having multiple users and user roles with different privileges.
- The system was able to produce detailed statistics.
- The system was able to capture patient information and make it accessible securely to 3rd party stakeholders.

Discussion of Results

- A positive user experience was made possible via quick registration (of patients and health institutions), simple sign-in, quick response times for various operations, and properly labelled controls.
- Users would always be informed in time thanks to immediate notice via system, SMS, and email notifications.
- The system was able to produce detailed statistical reports as required by the relevant authorities.
- The system was able to collect payments through all acceptable channels, which provides huge value to customers and makes the process frictionless.

6.2 Summary

The Digital Health Records system satisfies the project's requirements for success by achieving the goals, features, and functionality listed while being extremely efficient and offering a wonderful user experience. Using FDD made it possible to produce a product in the lowest amount of time with the fewest defects possible because debugging was quite simple.

Obtaining the specifications for what the system must accomplish was one of the obstacles encountered throughout software development.

6.3 Recommendations

The system for Digital Health Records can be improved in several ways. The system must be improved in several key areas, including the handling of insurance claims. This might entail adding directly into the system capability for processing insurance claims, which would speed up the way healthcare providers submit claims and receive payments. Automation of the insurance claims submission procedure lowers the possibility of errors and cuts down on the time and resources needed to manage claims.

The system's connection with telehealth platforms is another area that needs work. As the need for telehealth services increases, integrating the system with telehealth platforms can assist patients in receiving care remotely and enhance access to care in remote or underserved areas. By providing telehealth services via the system, patients may be assisted in avoiding the need for in-person appointments, cut down on wait times, and boost their satisfaction.

6.4 Future Work

Secure messaging will be incorporated into system upgrades in the future to improve patient satisfaction and patient-provider communication. This can enhance patient participation and happiness, which can ultimately result in better health results.

Another area that may be incorporated in future improvements is predictive analysis, which can be used to identify patients who are at high risk for specific health disorders or complications, allowing healthcare providers to intervene early and prevent or lessen these risks.

Bibliography

1. Taylor, David. (2017). *A Literature Review of Electronic Health Records in Chiropractic Practice: Common Challenges and Solutions,* Journal of Chiropractic Humanities. 24. 10.1016/j.echu.2016.12.001.)
2. Carbone, Sarah & Peckham, Allie & Bhatia, Dominika & Allin, Sara & Marchildon, Gregory. (2020). *Adopting EMRs and Information Technology in Primary Care*. NAO Rapid Review No. 18.
3. Adeyinka, Tirenioluwa. (2021). *Electronic Health Record Implementation in Developing Countries: A Systematic Review*.
4. Attafuah, Priscilla & Abor, Patience & Abuosi, Aaron & Nketiah-Amponsah, Edward & Tenza, Immaculate. (2022). *Satisfied or not satisfied? Electronic health records system implementation in Ghana: Health leaders' perspectiv*e. BMC Medical Informatics and Decision Making. 22. 10.1186/s12911-022-01998-0.
5. Mitreska, Izabela & Marina, Ninoslav & Capeska Bogatinoska, Dijana. (2021). *Electronic Health Records System for Efficient Healthcare Services.* 10.1007/978-3-030-73909-6_38.
6. Tubaishat, Ahmad. (2021). *The Adoption of Electronic Health Records in Primary Healthcare Settings*. CIN: Computers, Informatics, Nursing. Publish Ahead of Print. 10.1097/CIN.0000000000000755.
7. Cowie, Martin & Blomster, Juuso & Curtis, Lesley & Duclaux, Sylvie & Ford, Ian & Fritz, Fleur & Goldman, Samantha & Janmohamed, Salim & Kreuzer, Jörg & Leenay, Mark & Michel, Alexander & Ong, Seleen & Pell, Jill & Southworth, Mary & Stough, Wendy & Thoenes, Martin & Zannad, Faiez & Zalewski, Andrew. (2017). *Electronic health records to facilitate clinical research. Clinical research in cardiology*, Official journal of the German Cardiac Society. 106. 10.1007/s00392-016-1025-6.)