Republic of Yemen
Ministry of Higher Education and Scientific Research
Taiz University
Al-Saeed Faculty of Engineering and IT
Software Engineering Department

الجمهورية اليمنية
وزارة التعليم العالي والبحث العلمي
جامعة تعز
كلية السعيد للهندسة و تكنولوجيا المعلومات
قسم هندسة البرمجيات

AI-Driven Blockchain Platform for Secure and Efficient Patient Records Management

A graduation project document submitted to the Software Engineering department in partial fulfillment to the requirements for Bachelor Degree in Software Engineering

Submitted By:

Abdulrahman Hamood Mohammed Saeed

Ahmed Abdulhameed Abdulazeez qahtan

Mohammed Abdulazeez Qasem Mohammed

Mohammed Ali Abdo Muthanna

Osama Abdulwahed Abdo Noman Alathwari

Supervised by:
DR. AHMED ALSHAMERI

Yemen-Taiz
2023-2024

ABSTRACT

The current era has witnessed significant developments in networks, internet and Artificial Intelligence greatly impacting various domains, particularly web development. This project aims to build an AI-Driven Blockchain Platform for Secure and efficient Patient Records Management using modern technologies and frameworks. By integrating (AI) and blockchain technology, the platform ensures secure, accurate, and efficient management of patient data, addressing issues faced by traditional paper-based records and fragmented electronic systems. The proposed platform leverages AI for automating data entry, organization, and analysis, while blockchain technology ensures data integrity, transparency, and traceability. It offers functionalities such as secure data storage, interoperable data exchange, advanced analytics, patient empowerment, and real-time decision support. The project utilizes Ethereum for the blockchain infrastructure, Solidity for smart contracts, ReactJS for the front-end, Node.js for the back-end, and integrates with AI for intelligent data analysis. The platform aims to enhance security, privacy, and overall patient care quality by streamlining processes and ensuring efficient data handling.

Keywords: AI, Blockchain, Patient Records, Data Security, Interoperability, Smart Contracts, Ethereum, Solidity, ReactJS, Node.js, Data Analytics, Healthcare IT, Data Management, Decentralized Network, Privacy

الملخص

شهد العصر الحالي تطورات كبيرة في الشبكات والإنترنت والذكاء الاصطناعي، مما أثر بشكل كبير على العديد من المجالات، وخاصة تطوير الويب. يهدف هذا المشروع إلى بناء منصة بلوك تشين مدعومة بالذكاء الاصطناعي لإدارة سجلات المرضى بشكل آمن وفعال باستخدام التقنيات والأطر الحديثة. من خلال دمج الذكاء الاصطناعي وتقنية البلوك تشين، تضمن المنصة إدارة آمنة ودقيقة وفعالة لبيانات المرضى، مما يعالج المشكلات التي تواجهها السجلات الورقية التقليدية والأنظمة الإلكترونية المجزأة مثل وجود مشاكل في تكامل البيانات، وثغرات أمنية، ومشكلات في التوافقية. تستفيد المنصة المقترحة من الذكاء الاصطناعي لتحليل بيانات وسجلات المرضى واستخلاص البيانات وتلخيصها ، مما يحسن دقة وشمولية سجلات المرضى ويساعد مقدمي الرعاية الصحية على تقديم رعاية افضل، بينما تضمن تقنية البلوك تشين سلامة البيانات والشفافية وإمكانية التتبع مما يعالج المخاوف بشأن التلاعب غير المصرح به وانتهاكات البيانات.. توفر المنصة وظائف مثل التخزين الآمن للبيانات، وتبادل البيانات ، والتحليلات المتقدمة، وتمكين المرضى، والدعم الفوري لاتخاذ القرارات. يستخدم المشروع Ethereum للبنية التحتية للبلوك تشين، و Solidity للعقود الذكية، و ReactJS للواجهة الأمامية، و Node.js للواجهة الخلفية، ويتكامل مع الذكاء الاصطناعي لتحليل البيانات الذكي. تهدف المنصة إلى تعزيز الأمان والخصوصية وجودة رعاية المرضى بشكل عام من خلال تبسيط العمليات وضمان التعامل الفعال مع البيانات.

ACKNOWLEDGEMENT

We would like to express our gratitude to Allah Almighty who granted us the opportunity, determination, courage, and patience in difficult times, as well as the strength to complete this work. We also want to thank our family members who provided great support during this work and had faith in us. Without their support, we would not have achieved this success. Finally, we thank all the faculty members in the engineering department who imparted their knowledge to us and assisted us throughout the thesis period. In particular, we extend our gratitude and full appreciation to Dr. Ahmed ALSHAMERI, our project supervisor, for his valuable guidance and assistance that helped us successfully complete our project. We also want to thank the examination committee for their efforts and dedication in evaluating and discussing the project carefully. We wish you all the best in your future academic and professional endeavors.

Table of contents

ABBRAVIATIONS

Acronym	Definition
WWW	**World Wide Web**
ETH	**Ether**
HTML	**Hypertext Markup Language**
JS	**JavaScript**
CSS	**Cascade Style Sheets**
SQL	**Structured Query Language**
VS	**Visual Studio**
API	**Application Programming Interface**
UX	**User Experience.**
IBM	**International Business Machines Corporation**
AI	**Artificial Intelligence**
ML	**Machine Learning**
NLP	**Natural Language Processing**
SPA	**Single Page Application**
MPA	**Multi-Page Application**
NPM	**node package manager**
JIT	**Just in Time**
EVM	**Ethereum Virtual Machine**
DApps	**decentralized applications**
POS	**Proof of Stake**
RDBMS	**robust relational database management system**
MVC	**Model-View-Controller**
RSS	**Really Simple Syndication**
PHP	**Personal Home Page**
DOM	**Document Object Model**

HER	**Electronic Health Record**
IPFS	**InterPlanetary File System**
RPM	**Remote Patient Monitoring**
HIE	**Health Information Exchange**
EMR	**Electronic Medical Record**
MB	**MegaByte**
HTTP	**HyperText Transfer Protocol**
NET	**Network**
EOS	**Enterprise Operating System**
EOSIO	**Enterprise Operating System Input Output**
NEO	**New Economic Order**
MSP	**Managed Service Provider**
IMAP	**Internet Message Access Protocol**
LDAP	**Lightweight Directory Access Protocol**
README	**Read Me file**
CI	**Continuous Integration**
CD	**Continuous Delivery/Deployment**
ACID	**Atomicity, Consistency, Isolation, Durability**
SSL	**Secure Sockets Layer**
CID	**Content Identifier**
AXIOS	**Promise-based HTTP client for JavaScript**
REST	**Representational State Transfer**
XSRF	**Cross-Site Request Forgery**

LIST OF TABLES

LIST OF FIGURES

Chapter 1: Introduction

Chapter 1: Introduction

1.1 Introduction

Healthcare is one of the most data-intensive sector. However, unlike other sectors it has yet not utilized information technology to its full extent. [47]

To ensure quality patient care, the healthcare industry has undergone significant evolution. While many advancements focus on discovering new medicines and developing diagnostic and treatment tools, the rapid shift to digitizing patient information—such as complaints, test reports, diagnoses, and prescriptions—has brought numerous benefits to all stakeholders. Paper-based records had numerous drawbacks, including illegibility, storage and organization issues, information loss in disasters, and limited data-sharing capabilities. Electronic health records (EHRs) have introduced several improvements, such as easier access, interpretation, management, maintenance, sharing, and compilation of patient health records, as well as the generation of research questions. Most notably, historical data can now be used for identifying trends and correlations in diagnosis, anomaly detection, and treatment, significantly enhancing healthcare quality. [1].

The transition from paper-based records to electronic health records is due to some reasons such as: most handwritten prescriptions are illegible, paper-based records are expensive to copy, share, store, and transport, paper-based records are difficult to maintain and store for years, and it is challenging to track access records of paper-based records, which may raise privacy concerns. Traditional healthcare practices have been influenced by the usage of paper-based medical records, and these have evolved into electronic patient records.

"Electronic health records (EHR) often contain highly sensitive healthcare data, which are periodically distributed among healthcare providers, pharmacies and patients for clinical diagnosis and treatment. Furthermore, critical medical information must be regularly updated and shared where proper consent is provided by the patient. Along with this we need strong availability, fast access and the appropriate encryption of these records." [4].

Our project aims to address the longstanding challenges faced by healthcare providers in managing patient records securely, accurately, and efficiently. Traditional methods often involve paper-based systems or disparate electronic databases, leading to issues such as data silos, errors in record-keeping, and concerns regarding data security and privacy. By leveraging artificial intelligence (AI) and blockchain technology, we are pioneering a transformative solution that promises to revolutionize healthcare record management. AI

algorithms will be employed to automate various aspects of record-keeping, including data entry, organization, and analysis. This not only streamlines administrative processes but also enhances the accuracy and comprehensiveness of patient records. Furthermore, blockchain technology will serve as the foundation for a decentralized and immutable ledger of patient health data. Each interaction with the patient record, whether it be a new entry, update, or access request, will be cryptographically secured and recorded on the blockchain. This ensures data integrity, transparency, and traceability, mitigating the risk of unauthorized tampering or data breaches.

To develop this project, we will use Ethereum to build the Blockchain network, Solidity to build smart contracts and back-end, a single-page application library called ReactJS to build the front-end of our platform, Python for implementing AI features, and PostgreSQL and IPFS for managing the distributed database.

In conclusion, the healthcare industry has undergone significant evolution to ensure quality patient care, including a shift towards digitization of patient information from paper-based records. Electronic health records (EHR) offer advantages such as improved access, management, and sharing of patient data, addressing challenges like illegibility and storage issues associated with paper records. However, ensuring security, privacy, and efficient data handling remain critical concerns. Our project integrates AI and blockchain to revolutionize healthcare record management, automating processes and ensuring data integrity and transparency, thus addressing longstanding challenges faced by healthcare providers.

1.2 Problem Statement

The existing patient records management system operates on traditional paper-based methods, involving manual recording and storage of patient information in physical files. Administrative staff are responsible for inputting, organizing, retrieving, and updating patient records, which are stored in filing cabinets within healthcare facilities. Patient information, including personal details, medical history, test results, diagnoses, and treatment plans, is collected and documented on paper forms during registration or appointments. Each patient visit or encounter results in the creation of a new paper record, with subsequent updates made manually. However, this system functions independently of digital technologies and lacks integration with electronic health record (EHR) systems or

other digital platforms, necessitating physical transfer of records between healthcare providers or departments. Performance is hindered by manual data entry and retrieval processes, leading to potential errors, longer wait times, and scalability limitations due to physical storage constraints. Moreover, security concerns arise from the vulnerability of paper records to loss, theft, or damage, with limited control over access and viewing privileges. Maintenance involves regular upkeep of filing systems and support for staff training on record-keeping procedures, but transitioning to a digital system could offer significant improvements in efficiency, accessibility, and security for patient record management.

There are some problems associated with the existing paper-based patient records management system, including:

- **Limited Accessibility and Inefficiency:** With patient records stored in physical files within filing cabinets, healthcare providers often face delays in accessing vital information. In emergency situations, where every second counts, this can have critical consequences for patient care. Moreover, the manual retrieval process consumes valuable time that could otherwise be spent attending to patients, leading to inefficiencies and potentially impacting overall healthcare quality.
- **Data Redundancy and Errors:** The reliance on paper records increases the likelihood of data redundancy and inconsistencies within patient files. Duplicate entries, missing information, or outdated records can all contribute to errors in diagnosis, treatment, and medication management. Such inaccuracies not only compromise patient safety but also pose legal and regulatory risks for healthcare providers.
- **Security and Privacy Concerns:** Paper-based records are inherently vulnerable to loss, theft, or unauthorized access. Unlike digital records, which can be encrypted and protected with robust cybersecurity measures, physical files lack adequate safeguards to ensure patient confidentiality. Breaches in security can lead to breaches in privacy, eroding patient trust and exposing healthcare organizations to legal liabilities.
- **Interoperability Challenges:** The lack of standardized formats and protocols for sharing patient information across different healthcare settings complicates care coordination and continuity. In today's interconnected healthcare landscape, where patients may receive treatment from multiple providers and institutions, seamless data exchange is essential for delivering comprehensive and integrated care. The absence of interoperability hampers communication between healthcare professionals, leading to fragmented care and potential gaps in treatment.

- **Inefficiency and Time-Consuming:** Retrieving patient records from physical files is a time-consuming process, impacting the efficiency of healthcare professionals and contributing to delays in patient care.
- **Financial Implications:** Maintaining paper-based records incurs substantial costs associated with storage, maintenance, and administrative overhead. Healthcare organizations must allocate resources to physical storage facilities, as well as personnel responsible for organizing and managing paper files. Over time, these expenses can accumulate, diverting funds away from frontline healthcare services and technological advancements that could enhance patient care.

In addition to the limitations of the paper-based system, existing E-healthcare Systems also face the following challenges:

- **Data Fragmentation:** Electronic health records (EHRs) are often siloed within individual healthcare organizations, leading to fragmented patient data that is not easily shared or integrated across the healthcare ecosystem.
- **Interoperability Gaps:** Despite efforts to improve interoperability, many EHR systems still struggle to seamlessly exchange data with other healthcare information systems, hindering effective coordination of care.
- **Cyber Security Risks:** Digital health records are vulnerable to cyber threats, such as data breaches and ransomware attacks, which can compromise patient privacy and data integrity.
- **Technological Complexity:** The implementation and maintenance of EHR systems can be complex, requiring significant technical expertise and ongoing training for healthcare staff.
- **User Adoption Challenges:** Healthcare providers may face resistance to adopting new EHR systems, particularly if the user interfaces are not intuitive or the systems are perceived as adding to their administrative workload.
- **Scalability Limitations:** As the volume of patient data grows, many existing EHR systems may face challenges in scaling to accommodate increasing storage and processing requirements.
- **Regulatory Compliance:** Ensuring compliance with evolving healthcare regulations and data protection laws, such as HIPAA and GDPR, can be an ongoing challenge for EHR systems.

These limitations of both paper-based and existing electronic healthcare systems underscore the need for a transformative solution that addresses the critical issues of data security, interoperability, scalability, and user-centric design. The proposed AI-Driven Blockchain Platform aims to tackle these challenges and revolutionize the way patient records are managed in the healthcare industry.

1.3 Proposed System

The proposed AI-Driven Blockchain Platform represents a revolutionary solution aimed at transforming the landscape of patient records management in healthcare. By leveraging the synergies of Artificial Intelligence (AI) and blockchain technology, the platform offers a comprehensive and secure ecosystem for storing, accessing, and analyzing patient data.

Key Components:

The platform consists of several key components, each contributing to its functionality and efficacy:

- **Blockchain Infrastructure:** At its core, the platform utilizes a decentralized blockchain infrastructure to ensure data integrity, security, and immutability. Transactions related to patient records are cryptographically linked and stored across a distributed network of nodes, eliminating the risk of tampering or unauthorized access.
- **Smart Contracts:** Smart contracts, deployed on the blockchain, govern the rules and logic of data access and sharing. These self-executing contracts automate processes such as consent management, data sharing agreements, and access controls, ensuring compliance with privacy regulations and patient preferences.
- **Artificial Intelligence (AI) Engine:** The AI engine embedded within the platform enables advanced data analytics and decision support functionalities. Machine learning algorithms analyze patient records, extracting insights, predicting outcomes, and providing personalized recommendations for healthcare professionals.
- **User Interface (UI) and Applications:** Intuitive user interfaces and applications provide healthcare professionals and patients with seamless access to the platform's features. These interfaces facilitate secure data entry, retrieval, and visualization, enhancing user experience and promoting adoption.

Functionalities and Features:

The AI-Driven Blockchain Platform offers a range of functionalities and features designed to address the diverse needs of healthcare stakeholders:

- **Secure Data Storage:** Patient records, encrypted and securely stored on the blockchain, remain accessible only to authorized users. The decentralized nature of the blockchain ensures resilience against data breaches and ensures data availability even in the event of network disruptions.
- **Interoperable Data Exchange:** The platform facilitates seamless and interoperable exchange of patient data between healthcare providers, laboratories, insurers, and other stakeholders. Smart contracts govern data sharing agreements, ensuring compliance with regulatory requirements and patient consent.
- **Advanced Analytics:** AI-driven analytics empower healthcare professionals with actionable insights derived from patient records. Predictive modelling, risk stratification, and population health management capabilities enable proactive interventions and personalized treatment plans.
- **Patient Empowerment:** Patients have greater control over their health data, with the ability to access, monitor, and contribute to their electronic health records. Transparent consent mechanisms allow patients to manage access permissions and track data usage.
- **Real-Time Decision Support:** AI-powered decision support tools assist healthcare professionals in making informed clinical decisions. Real-time alerts, diagnostic assistance, and treatment recommendations enhance the efficiency and effectiveness of patient care.

Benefits and Advantages

The proposed AI-Driven Blockchain Platform offers several benefits and advantages over traditional patient records management systems and even existing Electronic Health Care Systems (EHCS):

- **Enhanced Security and Privacy:** The decentralized and immutable nature of the blockchain ensures enhanced security and privacy of patient data, mitigating the risks associated with centralized systems.
- **Interoperability and Data Exchange:** The platform promotes interoperability and seamless data exchange between disparate healthcare systems, fostering collaboration and continuity of care.
- **Personalized Healthcare:** AI-driven analytics enable personalized medicine, tailoring treatments and interventions based on individual patient characteristics and medical history.

- **Efficiency and Cost Savings:** Automation of administrative tasks, coupled with advanced analytics, streamlines processes, reduces paperwork, and optimizes resource allocation, leading to improved efficiency and cost savings.

1.4 Project Motivation

The motivation behind this project stems from the prevalent challenges commonly associated with traditional paper-based patient record management systems and the limitations of existing electronic healthcare (e-healthcare) systems. The paper-based approach faces inherent problems, including inefficient data storage, limited accessibility, security vulnerabilities, and interoperability issues. While the shift towards electronic health records (EHRs) and e-healthcare systems was a step forward, these digital solutions still struggle with data fragmentation, interoperability gaps, and cyber security risks.

To revolutionize the healthcare industry, this project seeks to leverage the power of artificial intelligence (AI) and blockchain technology. AI algorithms can analyze vast amounts of patient data, extracting valuable insights and providing personalized recommendations to healthcare professionals. The implementation of blockchain technology ensures the integrity, security, and immutability of patient records, mitigating the risks associated with unauthorized access, tampering, and data breaches. By combining the strengths of AI and blockchain, this project aims to create a secure, efficient, and interoperable platform that empowers healthcare providers, patients, and the broader healthcare ecosystem, ultimately leading to better health outcomes and an enhanced patient experience.

1.5 Project Objectives

The overarching goal of the project is to develop an AI-driven blockchain platform that revolutionizes the management of patient records within the healthcare industry. The primary focus of the project is to achieve the following objectives:

1- To develop Secure and Decentralized Platform:

- develop a secure and decentralized platform for the storage and management of patient records, leveraging the capabilities of blockchain technology.

- utilize blockchain to ensure data integrity, transparency, and resistance to unauthorized tampering.

2- To Integrate AI Algorithms:

- integrate advanced AI algorithms into the platform to analyze patient data comprehensively.
- extract valuable insights from the data to enhance diagnostic accuracy and treatment effectiveness.
- provide personalized healthcare recommendations based on AI-driven analysis.

3- To develop a Robust Access Control Mechanism:

- implement a robust access control mechanism to safeguard patient data privacy and confidentiality.
- utilize blockchain-based smart contracts and cryptographic techniques to enforce stringent access controls.

4- To facilitate Seamless Data Sharing:

- establish mechanisms for seamless data sharing between healthcare providers.
- enhance interoperability to promote effective coordination of care among different entities.

5- To develop User Interface Design:

- design an intuitive and user-friendly interface for healthcare professionals, ensuring efficient navigation and utilization.
- create a patient portal with secure authentication mechanisms, empowering individuals to access and manage their health records.

1.6 Project Scope

The project will focus on developing a core platform (web Application) powered with AI for users:

#Patients will be able to:

1. Create a patient account on the platform securely and easily
2. Login to their account from any device across the world
3. Access and manage their own encrypted medical records
4. Sync their records across all devices
5. Add Emergency contacts for Emergency access Protocol

6. View appointments with their doctors
7. View test results
8. Keep track of their medical records and doctor appointments
9. Authorize access to their records

#Doctors will be able to:

1. Create a doctor account on the platform securely and easily
2. Login to their account from any device across the world
3. Review patient records
4. Update patient records
5. Prescribe Medications
6. Schedule appointments for next visits
7. Request Lab tests
8. Request Radiology tests
9. View test results
10. Collaborate with other healthcare providers
11. Summarize Entire Patient history using AI
12. Ability to make accurate diagnosis using AI

#Pharmacies will be able to:

1. Create a Pharmacy account on the platform securely and easily
2. Login to their account from any device across the world
3. Receive Electronic Perceptions from patients via temporary access key
4. Dispense medications
5. Update patient medication records

#Laboratories will be able to:

1. Create a Laboratory account on the platform securely and easily
2. Login to their account from any device across the world
3. Receive and process test orders from doctors or patients
4. Perform tests
5. Upload results to the system
6. Securely share results with doctors and patients

#Radiology Center will be able to:

1 Create a Radiology Section account on the platform securely and easily
2 Login to their account from any device across the world
3 Schedule and perform imaging tests
4 Upload images results to the system
5 Securely share results with doctors and patients
6 Collaborate with doctors to interpret results

#Hospitals will be able to:

1 Create a Hospital account on the platform securely and easily
2 Login to their account.
3 create new Patient account
4 create new Doctor account
5 create new Pharmacy account
6 create new Laboratory account
7 create new Radiology center account
8 Access patients records via access key
9 Add to patient records

1.7 Methodology

This project will employ a comprehensive methodology to develop the AI-Driven Blockchain Platform for Secure and efficient Patient Records Management. The development process will include the following key stages and technologies:

1. **Blockchain Infrastructure**: The project will utilize the Ethereum blockchain platform to establish a decentralized network for secure and transparent data storage. Ethereum's smart contract functionality and distributed ledger technology will form the foundation of the patient records management system.
2. **Smart Contract Development**: The platform will leverage smart contracts to automate and enforce data access control, ensuring compliance with privacy regulations and patient preferences. These smart contracts will be developed using the Solidity programming language, which is specifically designed for creating self-executing contracts on the Ethereum blockchain.
3. **Front-end Development**: The user interface of the platform will be built using ReactJS, a popular JavaScript library for creating interactive and responsive web applications. ReactJS will enable the development of intuitive and user-friendly interfaces for patients, healthcare providers, and other stakeholders to interact with the platform seamlessly.
4. **Back-end Development**: The server-side logic and API infrastructure will be implemented using Node.js, a JavaScript runtime environment that allows for efficient and scalable back-end development. Node.js will handle data processing, integration with external systems, and communication between the front-end and the blockchain network.
5. **Database Management**: The platform will utilize PostgreSQL, a powerful open-source relational database management system, to store and manage off-chain data. PostgreSQL will ensure efficient querying, data consistency, and scalability for handling large volumes of patient records and related information.
6. **Artificial Intelligence Integration**: The project will integrate AI technologies to enable advanced data analytics, predictive modeling, and personalized recommendations. Python, a versatile programming language with extensive AI and machine learning libraries, will be used to develop and deploy AI models within the platform. These models will leverage patient data to provide actionable insights and support decision-making processes.
7. **Testing and Quality Assurance**: Rigorous testing and quality assurance processes will be implemented throughout the development lifecycle. This will involve unit testing, integration testing, and user acceptance testing to ensure the platform's functionality, reliability, and usability meet the required standards.
8. **Deployment and Scalability**: The platform will be deployed on a scalable infrastructure to handle increasing user demand and data volume. Techniques such as containerization and orchestration using tools like Docker and

Kubernetes will be employed to ensure efficient resource utilization and seamless scaling of the system.

1.8 Targeted Customers and Beneficiaries

1. Patients: Users will have access to secure and accurate medical records and will be able to share these records with whom they want on the system.

2. Health Providers: Users will have access to secure and accurate medical records with ability to update them and add new records.

1.9 Project Structure

This report contains six chapters:

Chapter 1: Introduces the project, highlighting the problem statement and the motivation behind it. The main objectives and sub-objectives of the project are outlined, along with the scope of the project and the methodology that will be used to achieve the objectives. It also identifies the targeted customers and beneficiaries of the AI-Driven Blockchain Platform for Secure and efficient Patient Records Management.

Chapter 2: Provides a comprehensive background and literature review on traditional healthcare systems, electronic healthcare records, and the evolution of e-healthcare. The chapter covers various topics, including the history of healthcare systems, key features of traditional healthcare, challenges of traditional healthcare, importance of e-healthcare, comparison between traditional and electronic healthcare records, and the significance of electronic healthcare records. The chapter also discusses web technologies, including WEB2 vs WEB3, centralization vs decentralization, blockchain technology, and its application in healthcare. Additionally, the chapter covers artificial intelligence (AI) and its importance in healthcare and patient records management. It also explores the technical tools and technologies used in the project, such as front-end and back-end frameworks, programming languages, databases, and development tools.

Chapter 3: Focuses on the analysis of the system requirements and the development model used. The chapter begins with an introduction to the analysis process. The development model used is discussed, followed by a detailed description of the system requirements, which include functional and non-functional requirements. The software and hardware requirements are also specified. The system architecture is then described, including the ReactJS, NodeJS, and Blockchain architectures. The chapter concludes with a discussion of system modeling, which includes use-case diagrams, sequence diagrams, and activity diagrams.

Chapter 4: Focuses on the design of the AI-Driven Blockchain Platform for Secure and efficient Patient Records Management. The chapter begins with an introduction, followed by a section on user interface design. This section covers various pages and components of the platform, such as the home page, registration page, user profiles, and more. The chapter also includes sections on API design, entity-relationship diagrams (ERD), and reports.

Chapter 5: Focuses on the implementation and testing of the AI-Driven Blockchain Platform. The chapter begins with an introduction, followed by a section on front-end implementation. This section covers the project folder structure, state management using Recoil, and back-end data connectivity using Axios. The chapter then moves on to back-end implementation, covering the development of APIs and integration with the blockchain. The blockchain implementation section discusses the use of Ethereum, smart contracts, and Solidity. The chapter also includes a section on testing and evaluation, which covers the testing methodologies and procedures used to ensure the system meets the specified requirements and functions as intended.

Chapter 6: Provides a conclusion to the AI-Driven Blockchain Platform for Secure and efficient Patient Records Management project. The chapter begins with an introduction, followed by a section on the achievements and objectives accomplished. The challenges faced during the development process are discussed, along with potential future work and enhancements to the platform. The chapter aims to summarize the project's outcomes, reflect on the development experience, and outline opportunities for further growth and improvement.

Chapter 2: Literature Review and Background

Chapter 2: Literature Review and Background

2.1 Overview

The chapter provides a thorough background and literature review on traditional healthcare systems and the evolution towards electronic healthcare records, it covers a wide range of topics, including history healthcare systems, key features of traditional healthcare and the challenges of traditional healthcare then it moves towards electronic healthcare systems and the significance of it in our life. Additionally, it provides a comparison between traditional and electronic healthcare records system, it also delves into WEB2 and WEB3 definitions, key differences between them and WEB3 limitations then it covers the centralization and decentralization concepts in a form of a comparison. After that it covers Blockchain technology by giving a history of this technology, listing its types, the different platforms to build the network and the importance of Blockchain in E-healthcare records. Moving to Artificial Intelligence and its various subfields, highlighting its potential to revolutionize healthcare through tasks including improved diagnostic accuracy, personalized treatment plans, and early disease detection. It discusses how AI can enhance healthcare efficiency and effectiveness, then explains how AI can be leveraged within the Patients Record Management System. Finally it gives an overview of technical tools that can be used to develop the platform such as front-end programming languages, libraries and frameworks, back-end programming languages and frameworks, the concept of smart contracts used in the development of WEB3 and Blockchain, Python and its importance in AI PostgreSQL as the database, IPFS as the new distributed protocol, Visual studio code as the main IDE for developing the platform, and GitHub as a real-time collaboration platform for sharing a workspace to develop our platform.

Having provided an overview of the chapter, let's dive deeper into the specifics of healthcare systems and the evolution of healthcare records. The following sections will explore traditional healthcare, the shift towards electronic healthcare records, and the importance of these advancements in improving patient care and streamlining healthcare processes.

2.2 Healthcare and Healthcare Records

2.2.1 Traditional Healthcare and Traditional Healthcare Records

"Traditional healthcare refers to the conventional methods of providing medical care and treatment that have been practiced for centuries. It encompasses a wide range of practices,

including diagnosis, treatment, and prevention of illnesses, as well as the promotion of overall well-being. Traditional healthcare typically involves face-to-face interactions between patients and healthcare providers, such as doctors, nurses, and other medical professionals, in settings such as hospitals, clinics, and private practices. "[2]

"Traditional healthcare faces several challenges in modern healthcare systems. One significant challenge is the reliance on manual record-keeping systems, which are often paper-based. This outdated method can result in inefficiencies in storing, retrieving, and organizing patient information, leading to potential errors and delays in healthcare delivery. "[2]

Traditional healthcare records, also known as paper-based medical records, have long been the primary means of documenting patient information in the healthcare industry. "Since time immemorial, hospitals and doctors have been giving handwritten or template-based prescriptions, test/scan reports and any such documents. These paper-based records were primarily used for clinical, research, financial and administrative purposes. "[1]. These records encompass a wealth of data concerning an individual's medical history, treatments, diagnoses, and interactions with healthcare providers. They are comprised of physical documents such as charts, forms, and files meticulously arranged to provide a comprehensive overview of a patient's health journey.

At the heart of traditional healthcare records lies a section dedicated to patient demographics. Here, basic identifying information such as name, date of birth, gender, address, contact details, and insurance information is recorded, serving as the foundational identifier throughout the record. Following this, a detailed medical history is outlined, cataloging past illnesses, surgeries, medications, allergies, and family medical background. This section offers critical context for current health concerns and informs medical decision-making.

Traditional healthcare records are meticulously organized and stored in physical filing systems within healthcare facilities. They are typically sorted alphabetically, by medical record number, or by date of service for ease of retrieval. While these records offer accessibility without relying on technology and are familiar to many healthcare professionals, they present challenges related to storage space, organization, and accessibility for authorized personnel. Nonetheless, traditional healthcare records remain a cornerstone of medical documentation, providing a tangible and comprehensive record of a patient's healthcare journey.

However, traditional healthcare records face several challenges and problems, including massive storage and difficult organization. With the increasing number of patients and the

volume of information being recorded, it may become challenging to quickly find specific information. Additionally, "Paper-based records are difficult to maintain and store for years. Paper, in general, is prone to destruction or decay. Environmental factors like moisture may damage it. Rodents like mice or insects like termites may chew and waste them. As well as human error, like misplacement, deliberate destruction, etc. is another enemy of the long life of papers. "[1].

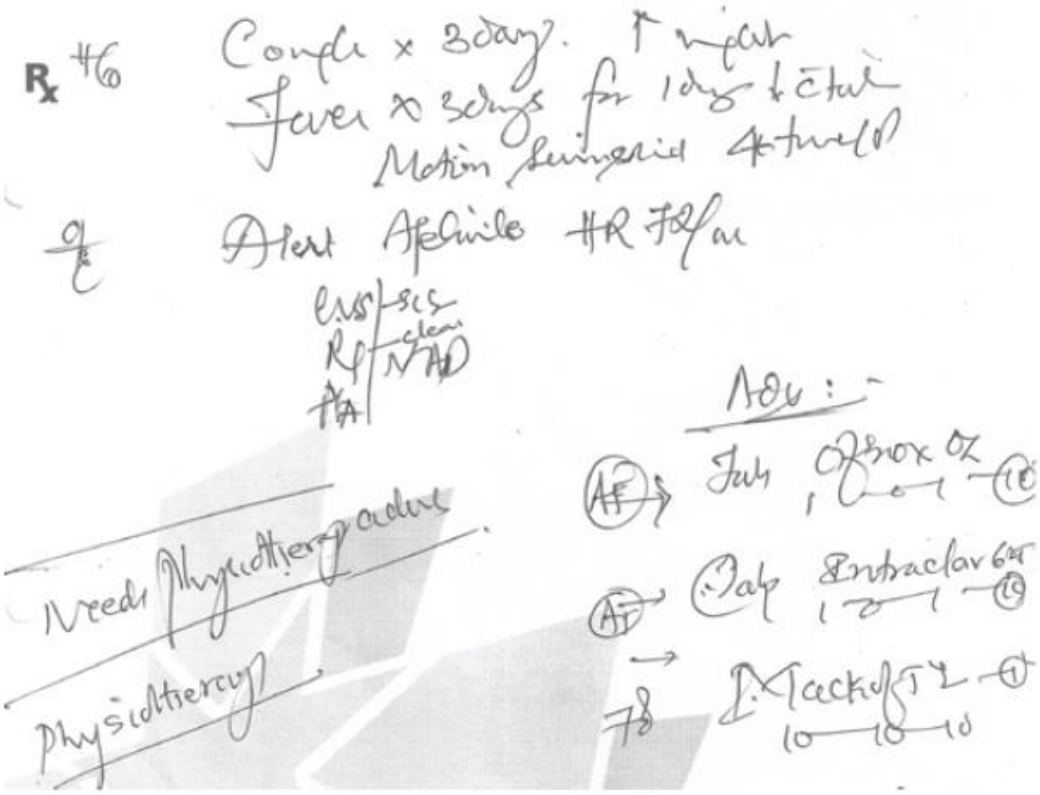


Figure 2-2 A sample handwritten prescription

Paper records are also vulnerable to unauthorized access, as anyone present in the medical facility can potentially access them, posing a threat to patient privacy and the security of their medical information. Furthermore, paper records may encounter challenges in sharing information among different healthcare providers, as they must be manually transferred between departments and different medical facilities.

Moreover, "Paper-based records are expensive to copy, share, store and transport. "[1], and traditional healthcare records can lead to delays in care delivery and medical decision-making, as manually searching for information can be time-consuming, negatively impacting the patient experience and the quality of care received. Additionally, it may be difficult to detect errors or changes in paper records due to the difficulty of making alterations without leaving a clear trace.

In this way, despite the many benefits offered by traditional healthcare records, they face multiple challenges that require innovative solutions to improve their efficiency and ensure the safety of patient information.

While traditional healthcare and paper-based records have been the norm for centuries, the advent of technology has paved the way for a significant shift towards electronic healthcare systems. The next section will explore the emergence of electronic healthcare and the benefits it brings to the healthcare industry.

2.2.2 Electronic Healthcare and Electronic Healthcare Records

“To ensure quality patient care, the health care industry has been evolving a lot. While most of the developments may be in the discovery of various medicines and development of various tools for diagnosis and treatment, a swift transition to the digitization of patient information viz. complaint, test reports, diagnosis and prescriptions, etc. brought about a plethora of benefits to all stakeholders. Paper-based records had many disadvantages specifically their illegibility, storage and organization, information loss in case of disaster, lack of facilities to share data and many more."[1].

"E-health, use of digital technologies and telecommunications, such as computers, the Internet, and mobile devices, to facilitate health improvement and health care services. E-health is often used alongside traditional “off-line” (non-digital) approaches for the delivery of information directed to the patient and the health care consumer."[11].

Electronic Health Records (EHRs) replace traditional paper-based records, facilitating seamless data sharing among healthcare providers. Telemedicine and Telehealth leverage telecommunications technology for remote clinical services and health-related education, expanding healthcare access. Mobile Health (mHealth) integrates mobile devices and applications for health monitoring, while Remote Patient Monitoring (RPM) utilizes technology to track real-time health data, especially for managing chronic conditions. Health Information Exchange (HIE) ensures secure data sharing across healthcare entities, promoting better care coordination.

"An EHR is a digital version of a patient’s paper-based records of health. EHR is a patient-centric system, which makes data available instantaneously and securely to the official users in real-time. It does contain the medical histories of patients, also go beyond the clinical data collection and give a broader insight into a patient’s care. These electronic data items can be managed, transmitted, stored, reproduced, and replicated efficiently. With the incredible evolution of the acceptance of EMR, different clinical data such as diagnostic

history, medications, lab test reports, demographics, vital sign, and so on are getting to be accessible. This sets up the EMR as a fortune gem for data analysis of health data." [1].

The primary purpose of an EHR is to automate information access, streamlining clinician workflows, and improving the overall efficiency of healthcare delivery. By consolidating patient information into a centralized digital platform, EHRs facilitate seamless communication and coordination among healthcare providers, ensuring that pertinent data is readily accessible when making care-related decisions.

One of the significant benefits of EHR implementation is its role in reducing medical errors. By improving the accuracy and clarity of medical records, EHRs help mitigate potential errors stemming from illegible handwriting or incomplete documentation. This, in turn, enhances patient safety and reduces the risk of adverse events during treatment.

Moreover, "EHRs are used to share data with other health care organizations for instance specialists, pharmacies, medical imaging facilities, laboratories, emergency facilities, and providers for instance clinics. Hence the information is gathered from all clinicians to provide better health care to a patient. Information shared in these records should be protected by law and to the authorized users only" [1].

Big Data Analytics and Artificial Intelligence (AI) contribute to data-driven decision-making, enabling predictive analytics and personalized medicine. E-Prescribing streamlines medication management, and Patient Portals empower individuals to access their health information and communicate with providers securely. Cybersecurity measures remain critical in safeguarding patient data within this evolving digital landscape.

Digital tools give healthcare providers an extensive view of patient health by significantly increasing access to health data and giving patients greater control over their health. The result is increased efficiency and improved medical outcomes. In addition, Innovative IoT applications in healthcare continue to emerge, giving patients better insight into the health data and transmitting the health information to their physicians [10].

There are several digital health technologies, for example, as shown in (Figure 2-1).

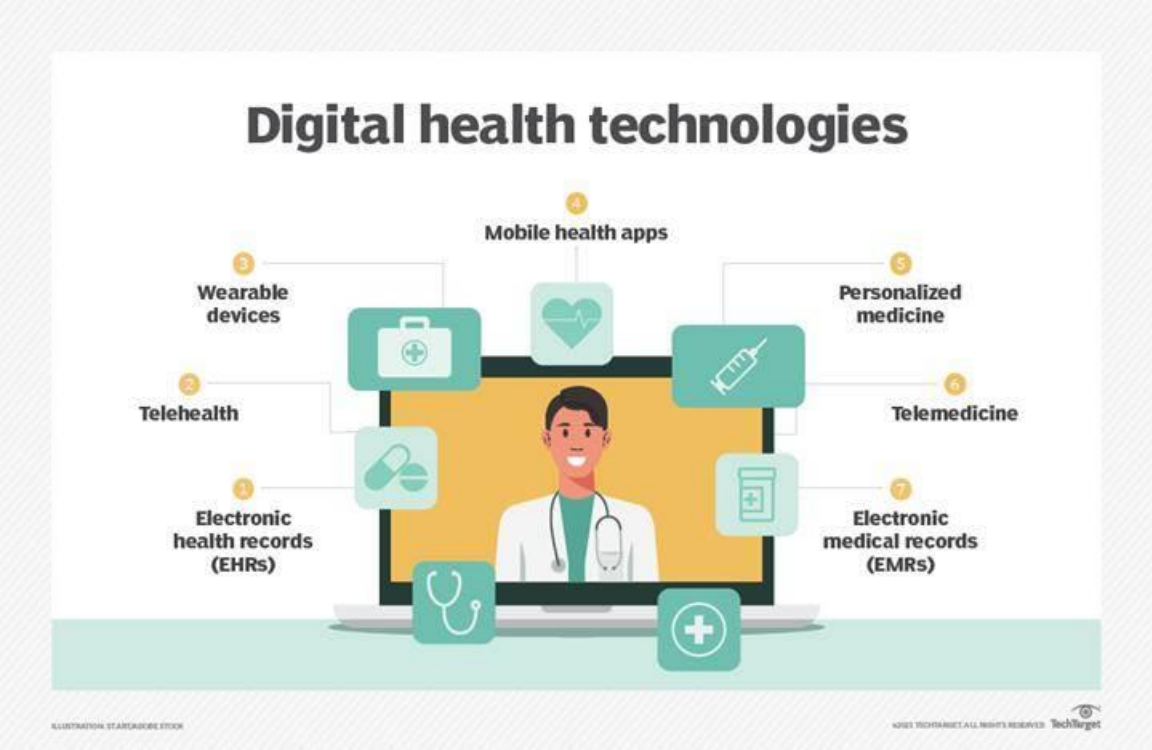


Figure 2-1 Digital health technologies

Furthermore, embracing EHRs empowers patients to actively participate in their healthcare journey. Through secure patient portals and online access to their medical records, individuals can review their health information, track their progress, and communicate with their healthcare providers more effectively. This transparency fosters a stronger connection between patients and clinicians, promoting shared decision-making and personalized care.

Overall, the implementation of EHRs represents a pivotal measure in optimizing the overall healthcare landscape. By providing timely and accessible data, EHRs enable healthcare providers to make informed decisions, improve care coordination, and ultimately enhance the quality and efficiency of patient care delivery.

The adoption of electronic healthcare and electronic health records has revolutionized the way healthcare is delivered and managed. In the following section, we will examine the importance of e-healthcare and how it contributes to improved patient care, increased efficiency, and better overall healthcare outcomes.

2.2.3 Importance of Electronic Healthcare Records

Electronic Health Records (EHRs) play a crucial role in modern healthcare systems, offering numerous benefits for both healthcare providers and patients. Here are some key reasons highlighting the importance of electronic health records:

1. Improved Accessibility and Efficiency: "Digital health records are one of the most reliable ways to get accurate patient information quickly. "[13]. EHRs enable quick and easy access to patient information by authorized healthcare professionals, regardless of their physical location. This accessibility enhances the efficiency of healthcare delivery, allowing for faster decision-making and improved coordination of care.

2. Enhanced Patient Care and Safety: EHRs provide a comprehensive and up-to-date overview of a patient's medical history, medications, allergies, and test results. This information is vital for healthcare providers to make well-informed decisions, leading to better patient care and reduced medical errors.

3. Security and Privacy: "Enabling the security and privacy of patients medical information within multiple providers. "[1].

4. Coordination of Care: Electronic health records facilitate better communication and coordination among different healthcare providers involved in a patient's care. This is particularly important for patients with chronic conditions or those receiving care from multiple specialists.

5. Reduced Duplication of Tests and Procedures: EHRs help eliminate unnecessary duplication of tests and procedures by providing a centralized repository of patient information. This not only saves time and resources but also reduces the potential risks associated with repeated diagnostic tests.

6. Cost Savings: "Benefit organization in lower business-related costs "[1]. Over time, the implementation of EHRs can lead to cost savings for healthcare organizations. Electronic records reduce paperwork, streamline administrative processes, and contribute to more efficient use of resources.

7. Patient Engagement: EHRs empower patients to actively participate in their healthcare. Patients can access their records, view test results, and communicate with healthcare

providers through secure online portals, fostering a more engaged and informed patient population.

8. Data Accuracy and Legibility: "Enhancing patient's safety by reducing diagnostic errors and improving accuracy "[1]. EHRs eliminate the issues related to illegible handwriting seen in traditional paper records. Electronic records are typed and standardized, contributing to improved accuracy and reducing the likelihood of errors in documentation.

9. Compliance with Regulations: Many healthcare systems around the world have implemented regulations and standards that encourage or mandate the use of electronic health records. Meeting these regulatory requirements ensures that healthcare providers maintain high standards of care and data security.

Electronic healthcare records have become a cornerstone of modern healthcare, offering advantages such as improved accessibility, enhanced patient care, and increased efficiency. To better understand the impact of this transition, the following section will compare traditional healthcare records with electronic healthcare records, highlighting the key differences and benefits.

2.2.4 Traditional Healthcare Records VS Electronic Healthcare Records

Traditional healthcare records, often paper-based, have been the primary means of documenting patient information for decades. They typically involve physical files stored in cabinets within healthcare facilities. On the other hand, e-healthcare records, also known as electronic health records (EHRs) or electronic medical records (EMRs), are digital versions of patients' medical histories.

The following table (Table 2-1**)** depicts a few of the vis-a-vis comparisons of EHR and paper-based records [1]:

Table 2-1 Table depicting few of the vis-a-vis comparison of EHR and paper-based records

Features	EHR	Paper-based records
Organization	Complete and systematic	Random
Sharing	Fast, efficient, economical	Cumbersome and costly
Accessibility	24*7, parallel access	Limited, no parallel access
Search	Fast	Slow

Loss or misplacement of records	Rare	Frequent
Maintenance	Easy	Difficult
Illegibility	None	Might be
Analysis	Easy	Difficult
Security	High	Low
Environmental impact	Low	High
Impact of disaster	Low	High
Alerts, reminders	Yes	No
Compliance	Yes	May/may not
Archival	Indefinitely long	Paper shelf life

The following table (Table 2-2) summarizes the key differences between traditional healthcare records and e-healthcare records across various aspects:

Table 2-2 comparison of traditional healthcare records and e-healthcare records

Aspect	***Traditional Healthcare Records***	***E-Healthcare Records***
Accessibility and Portability	Limited to physical location, manual sharing	Accessible remotely, easier sharing between providers
Storage and Space	Require physical space, filing cabinets	Stored electronically, no physical space needed
Data Security	Vulnerable to physical damage, theft	Enhanced security measures, encryption, access controls
Efficiency and Workflow	Manual retrieval, updating can be labor-intensive	Faster retrieval, automated data entry, streamlined documentation

Integration and Interoperability	Lack interoperability, limited exchange between systems	Integration with other systems, seamless data sharing
Cost	Initial setup costs lower, ongoing expenses for storage, printing	Higher initial implementation costs, long-term savings

The comparison between traditional healthcare records and electronic healthcare records demonstrates the significant advancements brought about by technology in the healthcare industry. As we move forward, it's essential to explore the broader context of web technologies and their role in shaping the future of healthcare. The next section will introduce the concept of WEB2 and WEB3, setting the stage for a deeper understanding of the technological landscape.

2.3 WEB

2.3.1 WEB2 VS WEB3

Web2 refers to the version of the internet most of us know today. An internet dominated by companies that provide services in exchange for your personal data. Web3 refers to decentralized apps that run on the blockchain. These are apps that allow anyone to participate without monetising their personal data.

Understanding the differences between WEB2 and WEB3 is crucial for grasping the potential of decentralized technologies in various domains, including healthcare. The following subsections will explore the benefits and limitations of WEB3, as well as the fundamental concepts of centralization and decentralization.

2.3.2WEB3 BENEFITS

Table 2-3 key differences between WEB2 and WEB3

Web2	***Web3***
Twitter can censor any account or tweet	Web3 tweets would be uncensorable because control is decentralized

Payment service may decide to not allow payments for certain types of work	Web3 payment apps require no personal data and can't prevent payments
Servers for gig-economy apps could go down and affect worker income	Web3 servers can't go down, they use a decentralized network of 1000s of computers as their backend

2.3.3 WEB3 LIMITATIONS

Web3 has some limitations right now:

- Scalability – transactions are slower on web3 because they're decentralized. Changes to state, like a payment, need to be processed by a node and propagated throughout the network.
- UX – interacting with web3 applications can require extra steps, software, and education. This can be a hurdle to adoption.
- Accessibility – the lack of integration in modern web browsers makes web3 less accessible to most users.
- Cost – most successful dApps put very small portions of their code on the blockchain as it's expensive.

The limitations of WEB3 highlight the ongoing challenges and areas for improvement in the development and adoption of decentralized technologies. To further understand the implications of WEB3, it's important to explore the fundamental concepts of centralization and decentralization, which will be covered in the next subsection.

2.3.4 CENTRALIZATION VS DECENTRALIZATION

Table 2-4 key differences between Centralized and Decentralized Systems

Centralized Systems	Decentralized Systems
Low network diameter (all participants are connected to a central authority); information propagates quickly, as propagation is handled by a central authority with lots of computational resources.	The furthest participants on the network may potentially be many edges away from each other. Information broadcast from one side of the network may take a long time to reach the other side.
Usually higher performance (higher throughput, fewer total computational resources expended) and easier to implement.	Usually lower performance (lower throughput, more total computational resources expended) and more complex to implement.

In the event of conflicting data, resolution is clear and easy: the ultimate source of truth is the central authority.	A protocol (often complex) is needed for dispute resolution, if peers make conflicting claims about the state of data which participants are meant to be synchronized on.
Single point of failure: malicious actors may be able to take down the network by targeting the central authority.	No single point of failure: network can still function even if a large proportion of participants are attacked/taken out.
Coordination among network participants is much easier, and is handled by a central authority. Central authority can compel network participants to adopt upgrades, protocol updates, etc., with very little friction.	Coordination is often difficult, as no single agent has the final say in network-level decisions, protocol upgrades, etc. In the worst case, network is prone to fracturing when there are disagreements about protocol changes.
Central authority can censor data, potentially cutting off parts of the network from interacting with the rest of the network.	Censorship is much harder, as information has many ways to propagate across the network.
Participation in the network is controlled by the central authority.	Anyone can participate in the network; there are no “gatekeepers.” Ideally, the cost of participation is very low.

The comparison between centralization and decentralization sets the stage for understanding the potential of blockchain technology, which is built upon the principles of decentralization. The following subsections will delve into the specifics of blockchain technology, its platforms, and its applications in healthcare.

2.3.5 Blockchain

Blockchain is a collection of computers connected together in a peer-to-peer (P2P) network. It is a decentralized, distributed ledger system that records transactions across multiple computers in a way that ensures transparency, security, and immutability. It is the combination of existing technologies such as cryptography, shared ledger and distributed network. "Blockchain technology can be defined as a distributed system in which transactional or historical data can be recorded, stored and maintained across a network. It is a non-changeable, public digital ledger similar to a database. Blockchain technology is a horizontal innovation that can be adopted by any industry."[2].

Blockchain 1.0 was announced as a bitcoin emergence in 2008; the chains of blocks which contain several units of information and transaction are which leverages the capabilities of digital ledger in an electronic P2P system. In 2013, Blockchain 2.0 introduced a public blockchain named Ethereum blockchain that facilitates a user to record the assets as smart

contracts. It acts as a platform for developing many decentralized applications. In 2015, Hyperledger was introduced as an open-source blockchain that promotes global industry collaboration by improving the reliability and performance of current systems. Later, the blockchain networks involving private, public and federated (consortium) blockchains have started evolving in the recent years, which improves operational efficiency in blockchain technology applications. A transaction is represented by a user initiating a request to create a block and then the transaction being broadcasted for each node that is connected in the P2P network. Using consensus algorithms, all nodes validate each transaction along with the user details in the blockchain network. Once the transaction is validated, the block that is newly created can be added to the existing blockchain. The basic working procedure of blockchain is depicted in (Figure 2-3):

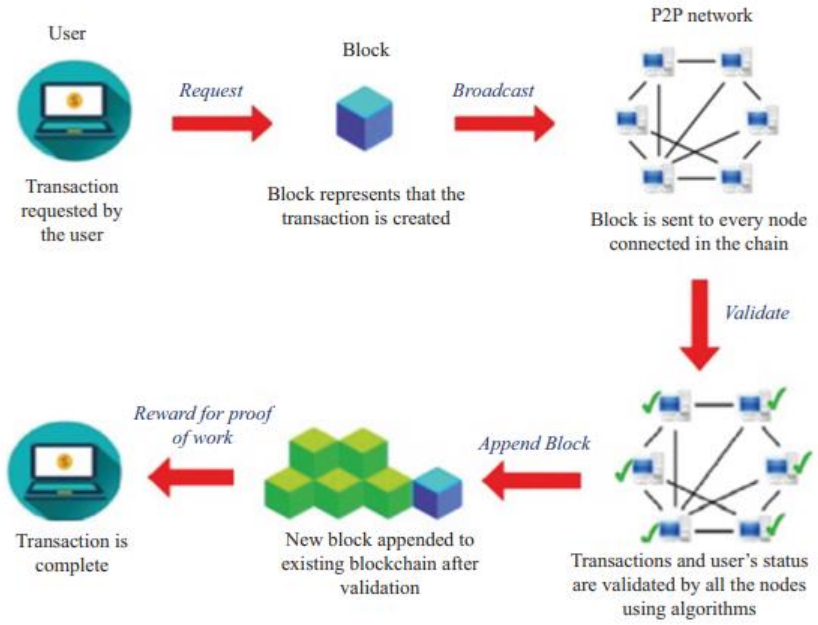


Figure 2-3 The basic working procedure of blockchain

Blockchain technology has certain inherent features that can be utilized for diversified applications. The various characteristics include the following:

- **Decentralized:** Public blockchain allows anyone connected to access, monitor, modify, and update the database without involving a central authority, reducing costs and performance issues. Consistency and integrity are maintained through consensus algorithms. In contrast, consortium and private blockchains vary in centralization, with the former being partially centralized and the latter fully centralized.

- **Transparency:** Verification and tracking of data are easily achieved as all users on the network, i.e., the public, have access to the data. Interactions among nodes are verified by an authorized entity, ensuring automatic facilitation of transparency.

- **Persistence:** "The transactions are distributed over all nodes that are validated and checked by other nodes using a consensus algorithm before it is being added to the block. Hence it is highly difficult to delete or alter any data. The public blockchain is immutable. However, if the majority of the nodes are interested in modifying the consortium and private blockchain, it can be altered."[2].

- **Distributed control:** The data stored in the blockchain is maintained in a distributed manner, which guarantees no single-point failure.

- **Provenience:** The blockchain allows for easy tracking of the origin of each transaction recorded in its ledger. The authenticity of stored data is guaranteed by digital signatures within the blockchain system.

Basic components of blockchain:

Cryptographic Hash Functions: Ensure data integrity by converting data into unique hash values.

- **Asymmetric-Key Cryptography:** Used for secure encryption, employing public and private keys for transaction integrity and privacy.

- **Transactions**: Involve data transfer among nodes, detailing sender/receiver information and transaction specifics.

- **Ledgers:** Utilize distributed databases synchronized across nodes, ensuring security and immutability via cryptographic techniques.

- **Blocks:** Immutable records storing digital transaction data, forming the blockchain structure.

Challenges and opportunities of blockchain technology :

- **Security and Privacy of Data:** Blockchain's decentralization raises security and privacy challenges for sensitive data. Robust access controls are needed to prevent

unauthorized access due to the open nature of blockchain networks." Medical data holds sensitive information in which safety and privacy should be ensured. Blockchain uses decentralized system in which the data is shared among different services and nodes; hence, there is a potential chance of data leakage."[2].

- **Scalability:** "Blockchain technology should be capable of handling a large number of users and medical devices such as sensors, smart devices or Internet of Things (IoT) that are more prevalent in the health industry. In bitcoin, the rate of growth of the chain for every 10 min is 1 MB per block along with copies of data stored in the nodes". [7].

  "Currently, blockchain networks are not as scalable as, for example, current financial networks. This is a known area of concern and a very ripe area for research." [3].

- **Interoperability:** "The storage of medical data is done mainly in a centralized database server that leads to data fragmentation, reduced data quantity and quality for medical research, slow access and lack of system interoperability."[8]. A single patient's medical record may be present at different locations in various systems. So, sharing and interoperability of data among various communicating providers and services is a major issue in blockchain. The transfer and sharing of data among various sources help health industry in providing improved services to the patients. Blockchain systems should be designed to be interoperable among different medical systems.

- **Accuracy:** The conventional health systems face the problem of data inconsistencies as the patient data may not be shared and updated by all parties involved in the system leading to inaccuracies and fragmented data. Rather in blockchain, the data can be shared and verified by all parties connected in the chain and updated immediately.

Type of Blockchain:

- **Public Blockchain:** Public blockchains are transparent and open, so any participant can avail the blocks at any time. For instance, bitcoin, a cryptocurrency and P2P payment system introduced by Satoshi Nakamoto, is based on public blockchain." In public blockchain, every member in the network can access the block and can make transactions, and every participant can involve in the process of creating the

consensus. In this case, there is neither an intermediatory register nor a trusted third party."[2]. The structure of public blockchain is shown in (Figure 2-4).

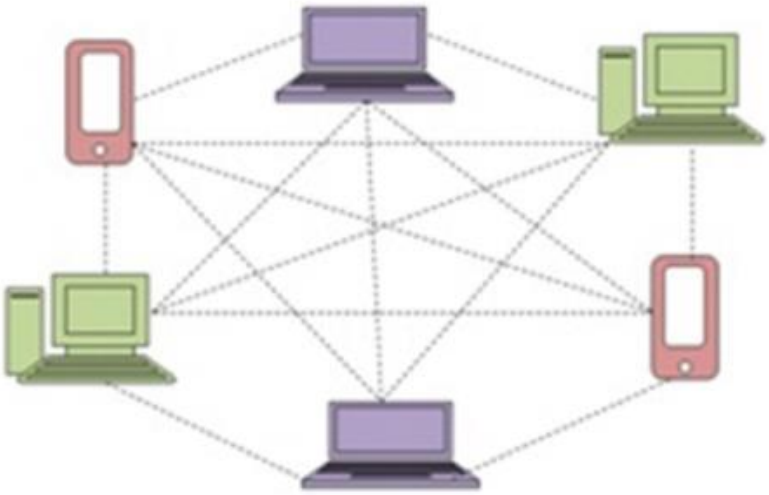


Figure 2-4 The structure of public blockchain

- **Private blockchain:** operates on access control, limiting participation to specific users and entities. This ensures reliability for transactions among third parties. Transactions are private, accessible only to involved entities. Examples include Linux Foundation and Hyperledger Fabric, commonly used in database management and auditing. Its primary applications are internal, maintaining data privacy and security, adhering to government regulations." The main advantages of private blockchain are data redundancies, easier data-handling, transaction cost and extra automated compliance functionalities."[2]. The structure of private blockchain is shown in (Figure 2-5)

Figure 2-5 structure of private blockchain

- **Consortium blockchain:** "Consortium blockchain is an amalgamation of public and private blockchains. Private blockchain is applicable for enterprise solutions to preserve business data. The consortium blockchain is considered as a semi-private blockchain with a restricted user group but available across various organizations. In other words, this type of blockchain can be utilized if organizations are ready to share the blockchain, but restrict data access to them, and retain it secure from public access."[2] Thus, it possesses the features of both public and private blockchains. Consortium blockchain is a cross-discipline and cross-company solution provider with the support of many blockchain platforms. A blockchain consortium of concurring companies can leverage information to advance workflows, accountability and transparency. The structure of Consortium blockchain is shown in (Figure 2-6).

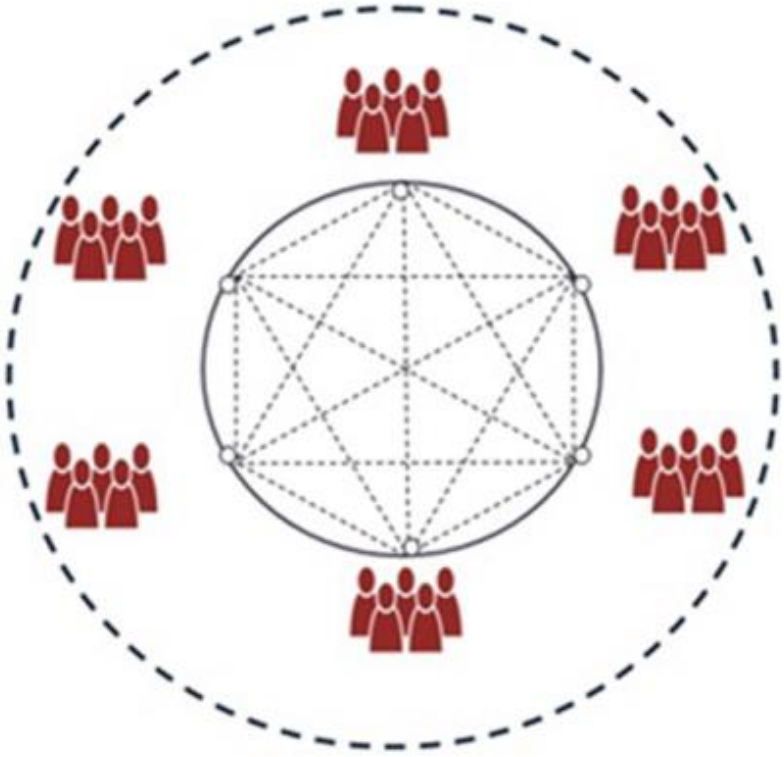


Figure 2-6 The structure of Consortium blockchain

Table 2-5 comparison between types of Blockchain

Feature	*Public Blockchain*	*Private Blockchain*	*Consortium blockchain*
Accessibility	Open to anyone who wants to participate and view transactions	Restricted access, typically limited to approved participants	Restricted access, available to approved participants from multiple organizations
Participation Control	Decentralized, anyone can participate in transaction validation	Centralized, access controlled by designated entities	Semi-decentralized, control shared among participating organizations
Transparency	High transparency, all transactions are visible to all participants	Limited transparency, transactions may only be visible to authorized parties	Moderate transparency, transactions visible to approved participants across organizations
Data Privacy	Low data privacy, all transaction data is publicly accessible	High data privacy, transaction data is only visible to authorized parties	Moderate data privacy, data access restricted to approved participants
Security	Relies on consensus mechanisms to secure the network	Relies on access control and encryption for security	Relies on access control, encryption, and collaboration for security
Speed and Scalability	Slower transaction speeds and lower scalability due to	Faster transaction speeds and higher scalability due	Moderate transaction speeds and scalability depending on consortium size

	open participation	to controlled participation	
Use Cases	Ideal for cryptocurrencies and public applications where transparency is key	Suitable for enterprise solutions requiring data privacy and control	Suitable for collaborative projects across multiple organizations requiring shared data access and control

2.3.6 Blockchain Platforms

2.3.6.1 Ethereum Platform

"Ethereum is a blockchain with a computer embedded in it. It is the foundation for building apps and organizations in a decentralized, permissionless, censorship-resistant way." [16]

Ethereum is a public network that requires users to make payments in the form of Ether (ETH) to access its computational resources.

"Ether (ETH) is the native cryptocurrency of Ethereum. The purpose of ETH is to allow for a market for computation. Such a market provides an economic incentive for participants to verify and execute transaction requests and provide computational resources to the network." [17]

Ether can be exchanged between users, used for trading, and utilized by developers on the Ethereum blockchain. Ethereum offers advantages such as decentralization, rapid deployment, permissioned network options, network size, private transaction capabilities, scalability, performance, transaction finality, tokenization of assets, and interoperability. However, there are also disadvantages, including the complexity of learning Solidity, scaling issues, and the risks associated with investing in Ethereum due to volatility and fluctuating fees. Ethereum uses programming languages like Solidity, LLL, Serpent, Vyper, Bamboo, and others. It can be both public and private, with applications built on the Ethereum Mainnet or private blockchains based on Ethereum technology. Data mining in

Ethereum involves creating and adding blocks of transactions to the blockchain through the Proof-of-Work consensus mechanism, securing the network.

2.3.6.2 IBM Blockchain Platform

"The IBM Blockchain Platform for IBM Cloud is the next generation of IBM Blockchain Platform offerings, which gives you total control over deployments, certificates, and private keys. It includes the new IBM Blockchain Platform console, a user interface that can simplify and accelerate the process of deploying components into a Kubernetes cluster on IBM Cloud managed and controlled by you. For more information about deploying a Kubernetes cluster on IBM Cloud." [18].

IBM Blockchain is a robust platform developed by IBM to help businesses build and manage blockchain networks for various applications. It offers several advantages such as security, transparency, efficiency, traceability, and scalability. However, there are also challenges associated with its implementation, including complexity, regulatory uncertainty, integration issues, and governance concerns.

"Smart contracts on IBM Blockchain can be written in languages like Solidity, JavaScript, Go, and Java, catering to different development preferences and requirements." [19]

The platform supports both public and private network configurations, with transaction costs varying depending on factors like network congestion and configuration.

While IBM Blockchain provides tools and frameworks for developing decentralized applications (dApps), the need for data mining within these applications depends on their specific functionalities and requirements. Overall, IBM Blockchain offers a robust foundation for building secure and scalable blockchain solutions, but businesses should carefully consider their needs and challenges before adopting the platform.

2.3.6.3 Hyperledger Platform

"Hyperledger Fabric is an enterprise-grade permissioned blockchain platform that offers advantages such as a permissioned network structure, modular architecture, high performance, privacy features, and robust identity management capabilities." [20].

"Fabric is the first distributed ledger platform to support smart contracts authored in general-purpose programming languages such as Java, Go, and Node.js, rather than constrained domain-specific languages (DSL). This means that most enterprises already have the skill set needed to develop smart contracts, and no additional training to learn a new language or DSL is needed." [20].

It's typically deployed as a private network, though certain data or services can be made public if needed. Transaction costs can vary based on network configuration, and dApps

built on Fabric do not require data mining for consensus. Overall, Hyperledger Fabric provides a flexible and customizable solution for organizations looking to build secure and scalable blockchain applications.

2.3.6.4 Hydrachain Platform

"In a joint venture of the Ethereum project and Brainbot technologies, Hydrachain is an open-source blockchain platform to support and create private/permission Blockchain networks." [21].

It serves as a flexible framework for building decentralized applications (dApps) and private blockchains. While specific details about HydraChain's features, advantages, and disadvantages may vary, it likely inherits some of the benefits associated with Ethereum, such as decentralization and support for smart contracts written in Solidity. Developers can deploy applications on both public and private networks, with transaction costs varying based on factors like network congestion and gas fees. Whether dApps built on HydraChain require data mining depends on their specific functionalities. Overall, HydraChain provides developers with a platform to create decentralized solutions while benefiting from the expertise of the Ethereum team.

2.3.6.5 R3 Corda Platform

"R3 Corda is an enterprise level blockchain technology platform, was founded in 2014 by David E Rutter. It offers to deploy interoperable blockchain networks that interact in rigid privacy and requires two types of consensus, validity consensus and uniqueness consensus, for a transaction to get indexed inside a block." [2].

It is an open-source blockchain platform designed specifically for the financial services industry. It was developed by R3, a company focused on creating interactive solutions for businesses using blockchain technology. Corda aims to address the specific needs and challenges faced by financial institutions by providing a secure and efficient platform for conducting transactions. Unlike traditional blockchain networks, Corda's architecture is designed to ensure privacy and confidentiality of transaction data, making it suitable for sensitive financial agreements. The platform utilizes a unique consensus mechanism called "Pluggable Consensus" that allows participants to select the consensus algorithm that best fits their needs. This flexibility enables Corda to be adaptable to various regulatory requirements and business preferences. Corda also offers "smart contract" functionality, known as "CorDapps," which are applications running on the network that can automate and enforce the terms of agreements without the need for intermediaries. This feature streamlines processes and increases efficiency in complex financial transactions. Furthermore, Corda emphasizes interoperability and connectivity, allowing different

businesses or consortia to easily connect and transact with each other. This makes it easier for financial institutions to collaborate and share information securely within the platform. Overall, R3 Corda provides a robust, secure, and scalable blockchain solution tailored for the unique requirements of the financial services industry. Its focus on privacy, flexibility, and interoperability makes it an attractive choice for organizations seeking to leverage blockchain technology in their operations.

2.3.6.6 Multichain Platform

"Multichain blockchain platform facilitates business houses and organizations to build and deploy private blockchain applications with speed and is peculiar for financial transactions only. While in other blockchain platforms, the design parameters are fixed but in Multichain blockchain platform, a single file params.dat holds the critical parameters which can be tweaked to suit the organizational requirements, and parameters like block incentive and transaction fees are null by default." [2].

It is also a versatile blockchain platform that allows organizations to create and deploy their own customized blockchain networks. Its advantages include customization, privacy, scalability, interoperability, and cost-effectiveness. However, there are also concerns such as centralization, complexity, and limited decentralization. Smart contracts on Multichain can be written in various programming languages, and it supports the creation of both public and private blockchain networks. Transaction costs may vary, and data mining for dApps built on Multichain depends on the specific requirements of the application. Overall, Multichain offers a flexible solution for implementing blockchain technology tailored to the needs of businesses and organizations.

2.3.6.7 BigchainDB Platform

"It has some database characteristics and some blockchain properties, including decentralization, immutability and native support for assets. At a high level, one can communicate with a BigchainDB network (set of nodes) using the BigchainDB HTTP API, or a wrapper for that API, such as the BigchainDB Python Driver. Each BigchainDB node runs BigchainDB Server and various other software." [22].

BigchainDB is an open-source, decentralized database system that combines the benefits of distributed databases and traditional blockchains. It provides decentralization, immutability, and scalability. While advantageous for various applications, it comes with complexities in implementation, potential storage costs, and a learning curve. BigchainDB supports multiple programming languages, and can be operated publicly or privately. Overall, it offers a flexible and versatile platform with considerations for both advantages and challenges.

2.3.6.8 Openchain Platform

"Openchain is an open-source distributed ledger technology designed for secure and scalable management of digital assets, is based on partitioned consensus wherein different sets of the data take part in different consensus protocols" [23].

It is a private blockchain with advantages like scalability, customization, and interoperability. The platform primarily uses programming languages such as C#, ASP.NET Core, JavaScript/TypeScript, and SQL.

Openchain is free and open source, the need for mining in Openchain depends on the organization's use case and goals, with potential applications in transaction analysis, smart contract monitoring, audit, security analysis, and business intelligence.

2.3.6.9 Quorum Blockchain Platform

"Quorum is a private/permissioned blockchain based on the official Go implementation of the Ethereum protocol2. Quorum uses a 'raft-based' consensus algorithm (a consensus model for faster blocktimes, transaction finality and on-demand block creation), and achieves Data Privacy through the introduction of a new "private" transaction type. One of the design goals of Quorum is to reuse as much existing technology as possible, minimizing the changes required to go-ethereum in order to reduce the effort required to keep in sync with future versions of the public Ethereum code base, much of the logic responsible for the additional privacy functionality resides in a layer that sits atop the standard Ethereum protocol layer. "[24].

Compared to public blockchains. Smart contracts on Quorum are typically written in Solidity, and it can be deployed as either a private or consortium network. Transaction fees can vary, and dApps built on Quorum may not require traditional data mining but may still involve data analysis for various purposes. Overall, Quorum presents a viable option for enterprise blockchain solutions, balancing benefits and challenges based on specific use cases and requirements.

2.3.6.10 EOS Blockchain Platform

"The EOSIO blockchain platform is the next-generation, open-source platform with industry-leading transaction speed and a flexible utility. As a blockchain platform, EOSIO is designed for enterprise-grade use cases and built for both public and private blockchain deployments. EOSIO is customizable to suit a wide range of business needs across industries with role-based permissions system and secure application transactions processing." [25].

EOS is a public blockchain platform known for its scalability, flexible governance model, and free transaction model for users. It allows for the development and deployment of decentralized applications (dApps) using smart contracts written in EOSIO C++ and other supported languages. While it offers advantages such as high throughput and easy upgrades, it also faces criticism for potential centralization concerns and complexity in development. Overall, EOS provides a platform for developers to build and deploy dApps with varying resource requirements, leveraging its unique features and governance structure.

2.3.6.11 Other Blockchain Platforms

There are other Blockchain platforms like:

1. Bitcoin
2. Binance Smart Chain
3. Cardano
4. Solana
5. Polkadot
6. Avalanche
7. Tezos
8. NEO

2.3.7 Blockchain in Healthcare

Blockchain technology's features such as decentralization, data immutability, security, privacy, availability, and transparency make it ideal for the healthcare sector. It promotes patient-centered interoperability over conventional institution-driven methods [38].

Blockchain offers a solution to secure patient records, ensuring confidentiality and preventing unauthorized access. It addresses issues like patient uniqueness and data sharing among healthcare providers. Blockchain facilitates secure data sharing through data sharding and encryption, allowing patients to control access to their medical records [39].

"Blockchain provides a secure, distributed environment for storing and sharing research data globally. This accelerates the research process and maintains data integrity through proof-of-existence for clinical trial documents. Any data modifications must be agreed upon by the majority of blockchain nodes, ensuring transparency and authenticity " [41].

"Blockchain can enhance insurance claim processes by utilizing its robust, distributed nature. Smart contracts between insurers and insurance companies automate and secure

claim settlements, reducing the need for intermediaries and minimizing fraud. The global healthcare market's blockchain spending is expected to reach $5.61 billion by 2025 "[40].

Drug counterfeiting is a significant issue, especially in developing countries. Blockchain enables traceability and security in drug supply chains, allowing drugs to be tracked from manufacturers to consumers. This ensures drug authenticity and helps prevent counterfeit drugs from entering the market [42].

Therefore, the Electronic Health Record has contributed to:

- "**Assure data integrity across multiple parties**

As organizations seek to confirm the health status of their customers and employees, blockchain networks help assure data integrity by storing an immutable, single version of the truth. Network participants can collaborate with confidence as they exchange information while controlling data access"[42].

- "**Achieve full traceability**

As pharmaceutical products move through the supply chain, they are recorded on a blockchain. That audit trail means an item can be traced back to its origin, or out to the receiving pharmacy or retailer. This helps reduce counterfeiting and manufacturers can locate a recalled product in seconds so they can respond rapidly"[42].

- "**Gain new operational efficiencies**

From resolving disputes to triggering next steps in supply chain transactions, to moving medical images through review, smart contracts can automate processes for increased speed and efficiency. Smart contracts automatically take action when established conditions are met"[42].

The potential of blockchain technology in healthcare is vast, ranging from secure data management to improved interoperability and patient empowerment. As we delve further into the world of decentralized technologies, it's crucial to understand the concept of wallets and their role in facilitating secure transactions and data access. The following subsections will explore wallets, their importance, types, structure, and operations.

2.3.8 Wallets

A wallet contains a set of user identities. An application run by a user selects one of these identities when it connects to a channel. Access rights to channel resources, such as the ledger, are determined using this identity in combination with an MSP.

2.3.9 Importance of Wallet

"When an application connects to a network channel such as PaperNet, it selects a user identity to do so, for example ID1. The channel MSPs associate ID1 with a role within a particular organization, and this role will ultimately determine the application's rights over channel resources. For example, ID1 might identify a user as a member of the MagnetoCorp organization who can read and write to the ledger, whereas ID2 might identify an administrator in MagnetoCorp who can add a new organization to a consortium." [30].

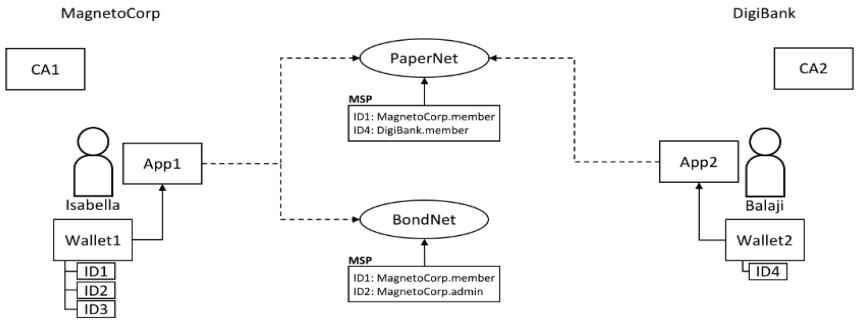


Figure 2-7 Two users, Isabella and Balaji have wallets containing different identities they can use to connect to different network channels, PaperNet and BondNet.

Consider the example of two users; Isabella from MagnetoCorp and Balaji from DigiBank. Isabella is going to use App 1 to invoke a smart contract in PaperNet and a different smart contract in BondNet. Similarly, Balaji is going to use App 2 to invoke smart contracts, but only in PaperNet. (It's very easy for applications to access multiple networks and multiple smart contracts within them.)

2.3.10 Types of Wallets

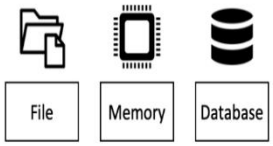


Figure 2-8 Wallet Types

- **File system**: This is the most common place to store wallets; file systems are pervasive, easy to understand, and can be network mounted. They are a good default choice for wallets.
- **In-memory**: A wallet in application storage. Use this type of wallet when your application is running in a constrained environment without access to a file system; typically, a web browser. It's worth remembering that this type of wallet is volatile; identities will be lost after the application ends normally or crashes.
- **CouchDB**: A wallet stored in CouchDB. This is the rarest form of wallet storage, but for those users who want to use the database back-up and restore mechanisms, CouchDB wallets can provide a useful option to simplify disaster recovery.

2.3.11 Structure of Wallet

single wallet can hold multiple identities, each issued by a particular Certificate Authority. Each identity has a standard structure comprising a descriptive label, an X.509 certificate containing a public key, a private key, and some Fabric-specific metadata. Different wallet types map this structure appropriately to their storage mechanism.

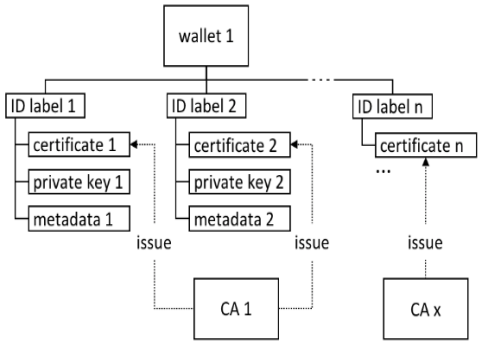


Figure 2-9 A Fabric wallet can hold multiple identities with certificates issued by a different Certificate Authority. Identities comprise certificate, private key and Fabric metadata.

2.3.12 Wallet Operations

The different wallet types all implement a common Wallet interface which provides a standard set of APIs to manage identities. It means that applications can be made independent of the underlying wallet storage mechanism; for example, File system and HSM wallets are handled in a very similar way.

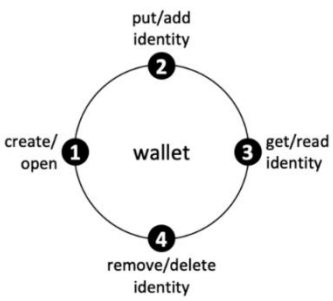


Figure 2-10 Wallets follow a lifecycle: they can be created or opened, and identities can be read, added and deleted.

An application can use a wallet according to a simple lifecycle. Wallets can be opened or created, and subsequently identities can be added, updated, read and deleted.

Wallet operations form the foundation for seamless and secure interactions within the blockchain ecosystem. As we continue our exploration of cutting-edge technologies, the next section will introduce artificial intelligence (AI) and its transformative potential in various domains, particularly healthcare.

2.4 Artificial Intelligence (AI)

2.4.1 Overview

AI, a use of man-made brainpower, centers around creating computer programs that can get to information and learn to take decisions alone. It gives systems the capacity to learn without being unequivocally programmed. Its essential objective is to build strong algorithms which can get input information and utilize statistical examination for predicting and error-free outputs." [2].

The roots of AI can be traced back to the mid-20th century when pioneers like Alan Turing and John McCarthy laid the groundwork for the field. Turing's concept of a "universal machine" and McCarthy's coining of the term "artificial intelligence" set the stage for decades of exploration and innovation. Early AI systems focused on symbolic reasoning

and rule-based approaches, culminating in expert systems that could emulate human expertise in specific domains.

Over time, AI has evolved significantly, driven by advancements in computing power, data availability, and algorithmic sophistication. Key milestones include the development of neural networks in the 1950s, the resurgence of deep learning in the 21st century, and breakthroughs in areas such as natural language processing (NLP), computer vision, and reinforcement learning. These advancements have propelled AI from a theoretical concept to practical applications with real-world impact.

At its core, AI revolves around several fundamental concepts:

1. **Machine Learning (ML):** "Machine learning is a method of data analysis that automates analytical model building. Here the systems can learn from data, identify patterns and make decisions with minimal human interference. ML is an application of artificial intelligence that provides systems the ability to automatically learn and improve without being clearly programmed. It focuses on the development of computer programs that can access data and use them in learning. The process of learning begins with observations, like direct experience, or instructions, in order to find patterns in data and make better decisions for the future" [2].

   ML algorithms enable systems to learn from data and improve performance over time without being explicitly programmed. Supervised learning, unsupervised learning, and reinforcement learning are common paradigms within ML.

2. **Deep Learning:** "Deep learning is a branch of ML, uses non-linear algorithms to enhance the performance and artificial neural networks with multiple layers to extract hierarchical representations of data." [2].

   It has driven significant breakthroughs in tasks such as image recognition, speech recognition, and natural language understanding.

3. **Natural Language Processing (NLP):** NLP focuses on enabling computers to understand, interpret, and generate human language. Applications range from chatbots and virtual assistants to language translation and sentiment analysis.

4. **Computer Vision:** Computer vision involves teaching computers to interpret and understand the visual world, enabling tasks such as object recognition, image classification, and autonomous driving.

5. **Reinforcement Learning:** Reinforcement learning is a branch of ML concerned with training agents to make sequential decisions in an environment to maximize cumulative rewards. It has applications in robotics, game playing, and autonomous systems.

AI has permeated various sectors, transforming industries and driving innovation in areas such as:

- Healthcare: AI is revolutionizing medical diagnosis, drug discovery, personalized treatment plans, and patient care management.
- Finance: AI algorithms are used for fraud detection, algorithmic trading, risk assessment, and customer service automation in the financial sector.
- Transportation: Autonomous vehicles powered by AI are poised to reshape the transportation landscape, improving safety, efficiency, and accessibility.
- Manufacturing: AI-enabled automation, predictive maintenance, and quality control are enhancing productivity and optimizing manufacturing processes.
- Retail: AI-driven recommendation systems, demand forecasting, and inventory management are enhancing the customer experience and optimizing operations in retail.

The overview of artificial intelligence provides a glimpse into the vast possibilities and advancements it offers. Building upon this foundation, the next subsection will explore the importance of AI specifically in the healthcare industry, highlighting its potential to revolutionize patient care and streamline healthcare processes.

2.4.2 Importance of AI in Healthcare

"The sophistication and growth of healthcare data imply that AI is becoming rapidly implemented in the area. Payers and care providers, and life sciences companies are already employing various types of AI. The key classes of utilizations include determination and treatment proposals, quiet commitment and adherence, and regulatory activities. Throughout the healthcare sector, the effect of AI is changing treatment delivery through natural language processing and ML. As is the case for many sectors, these innovations are projected to continue to develop at a steady rate over the next few years. As AI discovers its way into everything from our cell phones to the flexibly chain, applications in medicinal

services, its potential in healthcare services may involve activities that vary from easy to complex—everything from answering the phone to analyzing medical history and patterns and the monitoring of public health, developing medicinal medications and tools, interpreting radiology scans, creating clinical diagnosis and treatment decisions and even talking to patients." [5].

One of the key areas where AI is making a significant impact is In personalized Patient Care AI algorithms analyze patient data, including medical records, genomic information, and lifestyle factors, to generate personalized insights and treatment recommendations. This personalized approach enhances patient engagement, improves treatment adherence, and ultimately leads to better health outcomes [2].

Moreover, AI is revolutionizing diagnostic Accuracy and Efficiency AI-powered diagnostic tools can analyze medical images, pathology slides, and clinical data with remarkable accuracy and speed. Machine learning algorithms trained on vast datasets enable early detection of diseases, reducing diagnostic errors, and facilitating timely interventions [6].

In addition to diagnostics, AI is transforming the way healthcare is delivered through remote Monitoring and Telemedicine AI enables remote monitoring of patient vital signs, medication adherence, and disease progression, allowing healthcare providers to deliver virtual care effectively. Telemedicine platforms equipped with AI-driven chatbots and virtual assistants offer round-the-clock support, triage services, and medical advice to patients, particularly in underserved or remote areas [5].

Furthermore, AI is driving operational efficiency and optimization healthcare Operations Optimization: AI optimizes healthcare operations by streamlining administrative tasks, resource allocation, and workflow management. Predictive analytics models forecast patient demand, optimize bed utilization, and schedule appointments efficiently, reducing waiting times and improving healthcare service delivery[2].

AI is also acceleratingdrug Discovery and Development AI accelerates drug discovery and development processes by analyzing vast datasets, simulating biological processes, and identifying potential drug candidates. Machine learning algorithms predict drug-target interactions, optimize drug formulations, and expedite clinical trials, leading to the discovery of novel therapies for various diseases[5].

Another crucial application of AI in healthcare is Clinical Decision Support Systems AI-powered clinical decision support systems provide healthcare practitioners with evidence-

based recommendations, treatment guidelines, and real-time alerts. These systems analyze patient data, medical literature, and clinical guidelines to assist clinicians in making informed decisions, reducing medical errors, and enhancing patient safety [5].

Moreover, AI plays a vital role in healthcare Fraud Detection and Prevention AI algorithms detect anomalies, patterns, and inconsistencies in healthcare claims data to identify potential cases of fraud, waste, and abuse. Machine learning models analyze billing patterns, patient histories, and provider behavior to flag suspicious activities, mitigate financial losses, and protect the integrity of healthcare systems [6].

In continuous Learning and Improvement AI systems learn and adapt over time, continuously improving their performance and capabilities. Through iterative learning processes, feedback loops, and data-driven insights, AI fosters a culture of continuous improvement in e-healthcare, driving innovation, and advancing medical knowledge.

The importance of AI in healthcare cannot be overstated, as it has the potential to transform various aspects of the industry, from diagnosis and treatment to patient management and research. To further underscore the significance of AI, the next subsection will delve into its specific applications and benefits within the context of patient record management systems.

2.4.3 Importance of AI in Patients Record Management System

"The integration of Artificial Intelligence (AI) in a Blockchain-based platform for patient records management represents a groundbreaking approach that addresses critical challenges in the healthcare industry. This convergence of technologies offers unique advantages, revolutionizing how patient data is managed, secured, and utilized." [6].

First and foremost, AI algorithms can analyse vast amounts of patient data efficiently, extracting valuable insights to improve healthcare outcomes. By leveraging machine learning techniques, the platform can identify patterns, trends, and anomalies within patient records, aiding in diagnosis, treatment planning, and disease prevention [2].

Moreover, Blockchain technology ensures the integrity and immutability of patient records, protecting them from unauthorized access and tampering. AI-powered encryption mechanisms further enhance data security by identifying potential threats and implementing robust encryption protocols, safeguarding sensitive patient information against breaches and cyberattacks [6].

In interoperability and Accessibility AI algorithms facilitate interoperability between disparate healthcare systems, enabling seamless data exchange and collaboration among healthcare providers. Through natural language processing (NLP) and data standardization techniques, the platform can reconcile inconsistencies in patient records, ensuring data accuracy and accessibility across different healthcare environments [5].

Furthermore, AI-driven insights derived from patient records empower healthcare providers to deliver personalized and precision medicine approaches tailored to individual patient needs. By 48nalysing historical data, genetic information, and clinical variables, the platform can recommend optimal treatment plans, predict disease progression, and identify personalized interventions for better patient outcomes [6].

In the realm of Clinical Decision Support, AI algorithms embedded within the platform offer real-time clinical decision support to healthcare practitioners, assisting them in making evidence-based decisions at the point of care. From drug interactions and adverse event predictions to diagnostic assistance and treatment recommendations, AI augments clinician expertise, improving diagnostic accuracy and patient safety [2].

Moreover, AI-enabled predictive analytics optimize resource allocation within healthcare systems, reducing operational costs and improving efficiency. By forecasting patient demand, predicting readmissions, and identifying high-risk populations, the platform enables proactive resource allocation, ensuring that healthcare resources are allocated where they are needed most effectively [2].

Lastly, AI algorithms learn and adapt over time, continuously improving the performance and capabilities of the platform. Through feedback loops and iterative learning processes, the system evolves to address emerging healthcare challenges, incorporate new medical knowledge, and enhance decision-making accuracy, fostering a culture of continuous improvement and innovation in patient care.

The integration of AI in patient record management systems opens up new possibilities for enhancing the efficiency, accuracy, and security of healthcare data. As we move forward, it's essential to explore the various techniques and tools used in application development, which will be covered in the next section.

2.5 Techniques and tools used in application development

2.5.1 Internet Technology

The Internet is defined as an electronic communications network that connects the computer network, public network connecting millions of computers around the world. It consists of millions of governments, academic, commercial and small government networks. The Internet is at the same time a global broadcasting capability, an information dissemination mechanism and a medium for cooperation and interaction between individuals and their computers without regard to their geographical location, Today, the Internet has a great deal of data and services, and perhaps the most commonly used today is web pages and phone applications.

2.5.2 Web Programing Languages

Web development languages are programming languages used to create websites and web applications. There are several web development languages, each with its own syntax, features, and purpose. Some of these languages are for front-end programming, while others for backend programming. As illustrated in the following sections.

2.5.3 Single Page compared to Multi-Page Application

SPA (Single Page Application) and MPA (Multi-Page Application) are two different approaches to building web applications.

An SPA is a web application that loads a single HTML page and dynamically updates the content of that page as the user interacts with the application. The content is usually loaded using JavaScript frameworks such as React, Angular, or Vue.

SPAs provide a smooth user experience because they do not require the whole page to reload every time the user performs an action, and they can provide real-time updates without requiring a refresh.

On the other hand, an MPA is a web application that consists of multiple HTML pages, where each page represents a different functionality or feature of the application.

When the user interacts with the application, the server loads a new page and sends it to the client. MPAs are the traditional way of building web applications, and they provide a simple and straightforward approach to web development.

Here are some of the key differences between SPA and MPA

Table 2-6 key differences between SPA and MPA

Characteristic	*SPA*	*MPA*
Speed and Performance	SPA is usually faster than an MPA as most resources like HTML + CSS + Scripts are only loaded once throughout the lifecycle of applications	MPA is usually slower than SPA as Every change request renders a new page from the server in the browser.
Development Time	Developing, testing, and launching a single-page web app takes a lot less time as there is no need to write code and design an interface for multiple pages	Building a multi-page web application takes longer than building a single page app. This is because each page in your web app will need separate code and a separate design. Depending on the number and complexity of features, the time might also affect the cost
Navigation	SPA does not directly support back and forth navigation and sharing links of a specific location to a site, for this developers need to use an API.	The multi-page web application supports traditional navigation, each page of an MPA has its own URL that users can copy and paste. The backward and forward buttons also work easily.
Scalability	To make a SPA scalable developer might need to write big chunks of code.	MPAs are infinitely scalable.

2.5.4 Front-End Programing Languages

Front-end programming languages are used by developers to create the user interface of websites and web applications. "The component of an app or website that users interact with is called Front End or Client Side of the application. It includes everything that users come across directly including images, buttons, text colors, graphs, tables, etc." [9].

Some of the most popular front-end languages include HTML, CSS, and JavaScript. HTML and CSS are not programming languages, but a markup and a style sheet language. JavaScript is a programming language and is fundamental for website development.

2.5.4.1 Hypertext Markup Language HTML

"HyperText Markup Language, or HTML, is the standard markup language for describing the structure of documents displayed on the web. HTML consists of a series of elements and attributes which are used to mark up all the components of a document to structure it in a meaningful way."[26].

HTML used for creation hypertext documents that are platform independent. HTML gives a web page its structure. It is used to organize, format, and display a web page's content (like text, images, videos). HTML also makes it easier to navigate through the internet through hyperlinks. It allows users to control text and visual elements layout and display.

2.5.4.2 Cascade Style Sheeting CSS

"Cascading Style Sheets provide the look and feel, or presentation layer of the page."[26]. CSS is used to define the presentation of HTML documents. With CSS, we can assign the page layout, font properties, colors, sizes, borders, backgrounds, and positioning elements on the page. It is a markup language like HTML, and the two are frequently used together. CSS enables the display to be adjusted for different types of devices, such as large and small screens or printers. CSS is independent from HTML and does a few things that HTML can’t, which makes it useful for scaling content across different platforms as laptops and mobiles.

2.5.4.3 JavaScript

"JavaScript (JS) is a lightweight interpreted (or just-in-time compiled) programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles." [27].

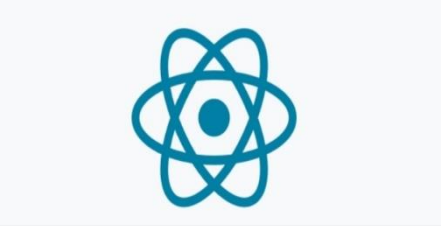
It is mainly used in web browsers to create more interactive pages. The main peculiarity of this language is that it's supported by most web browsers. Furthermore, JavaScript is a multi-paradigm language with numerous applications, but its ability to handle object-oriented styles makes it a wonderful companion for Java when it comes to websites. It can handle nested and anonymous functions, as well as classes, with well-structured syntax. So, one can create interactive websites that work on almost all web browsers using this coding tool. It also makes websites launch super-fast and provides the end users with enhanced user experience (UX). The web browser can naturally understand the language, like how a native English speaker can naturally understand English.

2.5.5 Frontend Frameworks and Libraries

2.5.5.1 ReactJS

ReactJS is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is developed by Meta. It can build web applications quickly and efficiently with significantly less code than you would with vanilla JavaScript. In React, you develop your applications by creating reusable components that you can think of as independent blocks. These components are individual pieces of a final interface, which, when assembled to form the application's entire user interface.

"ReactJS is a JavaScript library for developing dynamic User Interfaces. It's developed and maintained by Facebook. React is an efficient, declarative and versatile JavaScript library for building user interfaces. Complex UIs can be composed from small, isolated and reusable pieces of code called “components”. ReactJS uses JSX to simplify writing HTML. JSX is a pre-processor that adds XML syntax to JavaScript " [9]. As one of the top front-end frameworks React is an open-source tool powered by Facebook. It has been a developer’s delight owing to its salient features and user-friendliness. It is ideal for those who assume heavy traffic and need a robust platform for effective management. Its virtual DOM functionality and component-driven architecture offer great results.



Key features of React:

• Reusability with high performance

• Great community support

• SEO and user-friendliness

• Good interaction between JavaScript and HTML

• Friendly syntax with ease of coding

2.5.6 Back-End Frameworks and Languages

"Web creation at the Backend consists of countless activities. For example, protecting APIs (Application Programming Interfaces) against external attacks, authenticating users, enabling seamless interaction with databases, and handling user requests to collect and present the required information, etc. The backend frameworks enable all of these activities to developers simple and trouble-free." [9].

2.5.6.1 NodeJS

"Node.js is an open-source and cross-platform JavaScript runtime environment. It is a popular tool for almost any kind of project. Node.js runs the V8 JavaScript engine, the core of Google Chrome, outside of the browser. This allows Node.js to be very performant." [15].

"Node.js is an asynchronous event-driven JavaScript runtime designed to build scalable network applications. It supports the handling of many connections concurrently. Call back is triggered for each connection and it sleeps if there is no work to be done. This behavior unlike too many other common concurrency models,

in which operating system threads are used. Moreover, the users of Node.js need not worry about deadlocks, since there are no locks. Since almost no function performs an I/O in Node.js, the process never blocks. These properties make Node.js scalable. JavaScript language is used to write applications in Node.js and run with Node.js runtime environment. The package ecosystem of Node.js called npm (node package manager) has a large number of open-source libraries. Node.js uses Google Chrome's super-fast highly optimized V8 execution engine in JIT (Just in Time) compilation fashion to execute JS code by transforming them into machine language and optimizes through complicated methods such as code in lining, copy emission, etc." [9].

2.5.7 GitHub

"GitHub is a web-based interface allowing real-time collaboration. It encourages teams to work together in developing code, building web pages and updating content" [36]**,** It offers developers so many features like:

1. Distributed Version Control

GitHub offers a distributed version control system (DVCS) powered by Git, which enables us to efficiently manage changes to our codebase. With Git, every developer has a local copy of the entire project history, allowing for seamless branching, merging, and tracking of changes.

2. Collaboration

GitHub provides a robust platform for collaboration among team members and external contributors. Its features such as pull requests, issues, and project boards facilitate

communication, code review, and task management, thereby enhancing team productivity and fostering a collaborative environment.

3. Community and Ecosystem

GitHub boasts a vast community of developers and open-source projects. By hosting our project on GitHub, we tap into this vibrant ecosystem, making it easier for others to discover, contribute to, and provide feedback on our project. Additionally, GitHub's integration with various third-party services and tools further enriches our development workflow.

4. Documentation and Insights

GitHub offers comprehensive documentation features, including wikis and README files, enabling us to provide essential project information and guidelines for contributors. Moreover, GitHub's built-in analytics and insights help us track project activity, identify trends, and make data-driven decisions to drive the project forward.

5. Continuous Integration and Deployment (CI/CD)

Integrating GitHub with CI/CD pipelines allows us to automate the testing, building, and deployment processes, ensuring the reliability and stability of our software. GitHub Actions, in particular, provides powerful workflow automation capabilities, seamlessly integrated within our repository.

2.5.8 PostgreSQL

PostgreSQL is an open-source relational database management system (RDBMS) known for its robustness, extensibility, and adherence to SQL standards. Originally developed at the University of California, Berkeley, PostgreSQL has evolved into a mature and feature-rich database solution widely used in various industries and applications.

"PostgreSQL has earned a strong reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the dedication of the open source community behind the software to consistently deliver performant and innovative solutions. PostgreSQL runs on all major operating systems, has been ACID-compliant since 2001, and has powerful add-ons such as the popular PostGIS geospatial database extender. It is no surprise that PostgreSQL has become the open source relational database of choice for many people and organisations." [34].

Which offers so many advantages in our Blockchain based project Like:

1. **Relational Data Management**: PostgreSQL is a robust relational database management system (RDBMS) that allows you to efficiently store and manage structured data. While Hyperledger Fabric provides distributed ledger capabilities for immutable and transparent transaction recording, PostgreSQL complements it by offering a reliable storage solution for relational data associated with our blockchain transactions.
2. **Querying Flexibility**: PostgreSQL's powerful querying capabilities enable you to perform complex data retrieval operations. This is particularly useful when we need to analyze and extract insights from the data stored on our blockchain network.
3. **Scalability and Performance**: PostgreSQL is known for its scalability and performance optimizations. As our blockchain network grows and the volume of data increases, PostgreSQL can efficiently handle the load while maintaining high performance. With proper indexing and tuning, we can ensure that our database remains responsive even under heavy transactional workloads.
4. **Data Integrity and Security**: PostgreSQL offers robust features for ensuring data integrity and security. We can implement constraints, triggers, and data validation rules to enforce business logic and prevent invalid or unauthorized transactions. Additionally, PostgreSQL supports advanced security mechanisms such as role-based access control (RBAC), SSL encryption, and data encryption at rest, which help protect sensitive information stored in your database.
5. **Integration and Ecosystem**: PostgreSQL has a vibrant ecosystem with a wide range of tools, libraries, and extensions available. This allows us to easily integrate PostgreSQL with other components of our technology stack and leverage additional functionalities as needed. Whether it is geospatial analysis, full-text search, or compatibility with popular programming languages and frameworks, PostgreSQL offers numerous extensions and integrations to enhance the development experience.

6. **Community Support and Documentation**: PostgreSQL boasts an active community of developers and users who contribute to its ongoing development and provide support through forums, mailing lists, and online resources. This wealth of community-driven knowledge ensures that we can find solutions to any challenges we encounter while working with PostgreSQL in our Hyperledger Fabric project.

2.5.9 IPFS

"IPFS is a peer-to-peer version controlled filesystem that synthesizes learnings from many previous successful systems. IPFS combines a distributed Hash table, an incentivized block exchange, and a self-certifying namespace. IPFS is a peer-to-peer hypermedia protocol to make the web faster, safer, and more open.

The InterPlanetary File System (IPFS) which is a peer-to-peer distributed file system, aims to replace HTTP and build a better web for all of us. IPFS seeks to connect all computing devices with the same system of files. IPFS thinks that HTTP fails to take advantage of dozens of brilliant file distribution techniques invented in the last fifteen years" [28,29]

While Bitcoin blockchain has a standard block size of approximately 1 MB, and these contain around 1,500–1,900 transaction details being coordinated from one address to other. On the other hand, if we try to impose the same kind of architecture in healthcare, huge data sets envisaged in the healthcare domain would be a challenge to reckon. Huge data sets as being discussed in the domain of healthcare of the size in zettabytes would not be apt to be indexed in blocks. Even if we increase the block size to contain the healthcare data, it would have a snowball effect into the efficiency and transaction confirmation delays. Thus, there is a need such that the blocks contain the regular hashes of the transaction and data sets are stored in a decentralized manner. This decentralized storage can be coordinated with the aid of IPFS protocol. This integration offers numerous benefits like:

1. **Decentralized File Storage:** IPFS provides a decentralized file storage solution, allowing us to store and retrieve large files in a distributed manner. Unlike traditional centralized storage systems, IPFS distributes files across a network of nodes, ensuring redundancy and fault tolerance.
2. **Immutable File System:** IPFS employs content-addressed storage, where files are identified by their content rather than their location. This ensures immutability, as any change to a file results in a new content identifier (CID). This property aligns well with the principles of blockchain, where immutability is crucial for maintaining the integrity of data.
3. **Reduced Storage Costs:** By utilizing IPFS for file storage, we can potentially reduce storage costs compared to traditional centralized storage solutions. Since IPFS leverages peer-to-peer networking, users contribute storage and bandwidth resources, distributing the cost burden across the network.
4. **Enhanced Data Privacy:** IPFS allows us to encrypt files before storing them on the network, enhancing data privacy and security. This feature is particularly important for sensitive or confidential information stored on the blockchain.
5. **Efficient Content Distribution:** IPFS facilitates efficient content distribution by leveraging a distributed network of nodes. Files are cached locally, reducing latency and improving performance for users accessing the content.
6. **Immutable Off-Chain Data:** We utilize IPFS to store off-chain data, such as large documents or media files associated with transactions on the Hyperledger Fabric blockchain. This approach ensures that transaction data remains immutable on the blockchain while off-chain data can be efficiently stored and accessed through IPFS.
7. **Secure Document Management:** IPFS integration enhances document management capabilities within our application. Documents can be securely stored on IPFS, with their content addresses stored on the blockchain, providing a secure and tamper-resistant reference to the documents.
8. **Scalability and Performance:** By offloading file storage to IPFS, we improve the scalability and performance of our Hyperledger Fabric network. This separation of concerns allows the blockchain to focus on transaction processing, while IPFS handles large file storage and retrieval operations.

9. **"Regulatory Compliance:** IPFS integration helps address regulatory compliance requirements by providing a secure and auditable solution for storing and managing documents associated with blockchain transactions. Compliance-related documents can be securely stored on IPFS, with their integrity guaranteed by blockchain immutability.

10. **Performance:** IPFS provides faster access to data by enabling it to be replicated to and retrieved from multiple locations, and allowing users to access data from the nearest location using content addressing instead of location-based addressing. In other words, because data can be addressed based on its contents, a node on the network can fetch that data from any other node in the network that has the data; thus, performance issues like latency are reduced.

11. **Link rot:** IPFS eliminates the problem of link rot by allowing data to be addressed by its content, rather than by its location. So, in other words, content in IPFS is still reachable regardless of its location, and does not depend on specific servers being available**.**

12. **Data sovereignty:** IPFS protects data sovereignty by enabling users to store and access data directly on a decentralized network of nodes, rather than centralized, third-party servers. This eliminates the need for intermediaries to control and manage data, giving users full control and ownership over their data.

13. **Local-first software:** IPFS benefits local-first software by providing a performant, decentralized, peer-to-peer data addressing, routing, and transfer protocol that prioritizes data storage and processing on individual devices. With IPFS, data can be stored, verified and processed locally, and then synchronized and shared with other IPFS nodes when a network connection is available.

14. **Vendor lock-in:** IPFS prevents vendor lock-in, as users have sovereignty over their data and infrastructure. This is enabled by content-addressing, which decouples the data from a single location or infrastructure provider. Unlike traditional cloud vendors, IPFS enables you to change data storage locations without changing things like APIs and data management. In addition, because IPFS is open-source, community-maintained and modular, users are not obligated to use a particular subsystem (described in How IPFS works). Instead, users can customize IPFS for their preferred technologies, needs and values." [35].

2.5.10 Visual Studio Code

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET).

2.5.11 Smart Contracts

Smart contracts are self-executing contracts with the terms of the agreement directly written into code. In the context of Project, smart contracts facilitate automated and secure execution of predefined business logic on the blockchain. This ensures transparency, immutability, and trust in the execution of transactions. In the development of Our Project , the decision to use Java as the programming language for writing smart contracts on the Hyperledger Fabric.

2.6 Conclusion

To develop our AI-Driven Blockchain Platform for patient records management system we decided to use Ethereum, React, Single page Application, Node.js, Python, PostgreSQL and IPFS. This combination of technologies allowed to create a secure, fast, and responsive platform that offers a great user experience.

In the past healthcare was based on paper records which face several challenges and problems, including massive storage and difficult organization. With the increasing number of patients and the volume of information being recorded, it may become challenging to quickly find specific information. Another solution was Automating this task which was good at the beginning , made the work easier, but when data got bigger and bigger this became a critical issue, because of the centralized model used in this system which made them less secure and less functional compared to our proposed system

With the advent of modern technologies like Blockchain, ReactJS, Node.js, and Single Page Application (SPA) architecture, developers have been able to so many platforms that are fast, secure, and responsive. One of the key benefits of using ReactJS is its ability to increase the speed and responsiveness of your platform. ReactJS is a JavaScript library for

building user interfaces, and it is well-suited for creating dynamic and responsive interfaces for web applications. It uses a virtual DOM (Document Object Model) to optimize rendering performance, which means that updates to the user interface are made more quickly and efficiently. This can lead to faster load times, smoother transitions, and a more responsive user experience overall. Another technology used, Ethereum, played a crucial role in ensuring the security of our platform, Combined with Solodity as a tool for writing and building smart contracts makes a powerful combination in creating WEB3 applications. Additionally, Python is used to implement AI Features in our platform with addition to PostgreSQL and IPFS for managing and sharing the distributed database

Chapter 3: Analysis

Chapter 3: Analysis

3.1 Introduction

The Analysis section delves into a comprehensive examination of the requirements, constraints, and objectives of the software project, laying a crucial foundation for designing a system that effectively meets user needs while addressing technical and operational limitations. This section begins with an overview of the Development Model Used, which outlines the framework and methodologies guiding the project's lifecycle. We then move into a detailed analysis of System Requirements, segmented into Functional Requirements that describe specific behaviors and functions of the system, and Non-Functional Requirements that define the system's performance and quality attributes.

Following this, we explore the Software Requirements, identifying the necessary software tools, libraries, and platforms required for development and deployment. We also examine the Hardware Requirements to ensure the system operates efficiently within the specified hardware environment. The section proceeds with an in-depth look at the System Architecture, detailing the structural design across three key layers: ReactJS Architecture, NodeJS Architecture, and Blockchain Architecture, each addressing different components and their interactions within the system.

Further, System Modeling is covered comprehensively through the development of Use Case Diagrams to illustrate user interactions, Sequence Diagrams to depict the flow of operations, and Activity Diagrams to represent the dynamic aspects of the system. Through meticulous analysis using these structured subsections, this section aims to provide a detailed understanding of the project's goals and strategies, ensuring alignment with objectives and facilitating the delivery of a robust and effective solution.

3.2 The Development model used

The development model used for this software project is Agile, a flexible and iterative approach that emphasizes collaboration, customer feedback, and rapid delivery. Agile methodology is particularly suited for dynamic and complex projects where requirements may evolve over time. It breaks down the project into small, manageable increments called sprints, typically lasting two to four weeks. Each sprint involves cross-functional teams working collaboratively to deliver functional components of the software, which are then reviewed and tested. This iterative cycle allows for continuous improvement and adaptation based on stakeholder feedback, ensuring that the final product aligns closely with user needs and expectations. Agile's emphasis on direct communication, regular updates, and

incremental progress helps in mitigating risks early and ensures that the development process remains aligned with the project's goals. By promoting a responsive and adaptive workflow, Agile facilitates the efficient handling of changes and challenges, ultimately leading to a more resilient and user-centric software solution.

3.3 Requirements Gathering

The requirements gathering process for this project involved a thorough investigation of the existing challenges faced by healthcare providers in managing patient records. This was accomplished through a combination of techniques, including:

1. **Stakeholder Interviews**: The project team conducted extensive interviews with healthcare professionals, including patients, doctors, hospitals, pharmacies, laboratories and radiology centers, to understand their pain points, requirements, and expectations for the patient records management system.
2. **Industry Research**: The team conducted a comprehensive analysis of industry best practices, regulatory guidelines, and emerging trends in healthcare IT to ensure the project aligns with industry standards and addresses the evolving needs of the healthcare sector.
3. **Process Mapping**: The team mapped the existing patient records management workflows within healthcare organizations to identify areas for improvement and opportunities for automation and optimization.
4. **Competitive Analysis**: The team reviewed and benchmarked similar patient records management solutions, both paper-based and electronic, to understand the strengths, weaknesses, and key differentiators of the proposed system.

3.4 System Requirements

3.4.1 Functional Requirements

The following is the desired functionality.

- **For the Patients**
  - Patient will able to Register a new account.
  - Patient will able to Login their account.
  - Patient will able to Access and management of medical records.
  - Patient will able to Add Emergency contacts.
  - Patient will able to View test results.
  - Patient will able to Authorize access to data.

- Patient will able to Summarize their entire records
- Patient will able to Add old paper records.
- Patient will able to Logout from their a account.

❖ **For the Doctor**

- Doctor will able to Register a new account.
- Doctor will able to Login to their account.
- Doctor will able to Review patient records.
- Doctor will able to Update patient records.
- Doctor will able to Prescribe Medications.
- Doctor will able to Schedule appointments.
- Doctor will able to Request tests.
- Doctor will able to View test results.
- Doctor will able to Summarize Entire Patient history using AI.
- Doctor will able to make accurate diagnosis using AI.
- Doctor will able to Logout from their account.
-

❖ **For the Pharmacies**

- Pharmacies will able to Login to their account .
- Pharmacies will able to Create a Pharmacy account on the platform securely and easily.
- Pharmacies will able to Receive Electronic Perceptions.
- Pharmacies will able to Dispense medications.
- Pharmacies will able to Update patient medication records.
- Pharmacies will able to Logout from their account.

❖ **Hospitals**

- Hospital will able to Register a new account.
- Hospital will able to Login to their account.
- Hospital will able to create new Patient account
- Hospital will able to create new Doctor account
- Hospital will able to create new Pharmacy account
- Hospital will able to create new Laboratory account
- Hospital will able to create new Radiology center account
- Hospital will able to Logout from their account.

❖ **For the Radiology**

- Radiology will able to Register a new account.
- Radiology will able to Login to their account.
- Radiology will able to Receive and process test orders from doctors or patient
- Radiology will able to Upload images and results to the system.
- Radiology will able to Securely share results with doctors and patients.
- Radiology will able to Logout from their account.

❖ For the Laboratory

- Laboratory will able to Register a new account.
- Laboratory will able to Login to their account.
- Laboratory will able to Receive and process test orders from doctors or patient
- Laboratory will able to upload test results to the system.
- Laboratory will able to Securely share results with doctors and patients.
- Laboratory will able to Logout from their a account.

3.4.2 Non-functional Requirements

Non-functional Requirements It specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other nonfunctional standards that are critical to the success of the software system.

- **Secure:** The system must be able to provide security against any external injections by using a layered security system. Implementation of user login functionalities also ensures the system is secure from unauthorized persons.

- **Privacy:** The system should protect sensitive information and ensure that data is only accessible to authorized parties, using cryptographic techniques to secure data.

- **centralization:** The system should maintain a high degree of decentralization to ensure no single point of control or failure, adhering to core blockchain principles.

- **Availability:** The system should remain operational in any day and any place.

- **Compliance:** The system must comply with relevant legal, regulatory, and industry standards to ensure legality and industry acceptance.

- **Accuracy:** There is a need to optimize the system to ensure more accurate results and calculations.

- **Latency:** The system should minimize the delay in processing transactions and providing feedback to users to ensure efficient operation.

- **Usability:** The system should provide a User-friendly user interface and tooltips to enhance itself and be effectively responsive.

- **Portability:** compatible with a variety of device (web)

- **Performance of the system:** Response time is very good for given piece of work. The system will support multi user environment.

- **Reliability of the system:** The system will be highly reliable and it generates all the updates information in correct order. Data validation and verification is done at every stage of activity system recovery will also be speed.

- **Scalability:** The system must handle increasing amounts of transactions and data efficiently without performance degradation.

- **Latency:** The system should minimize the delay in processing transactions and providing feedback to users to ensure efficient operation.

- **Resilience:** The system should quickly recover from failures, ensuring minimal downtime and data loss, and handle unexpected spikes in activity or malicious attacks.

- **Maintainability:** The system should be easy to maintain and update, with clear documentation and a modular architecture to facilitate quick bug fixes and enhancements.

3.5 Software Requirements

The software requirements for this project encompass a range of tools and technologies essential for developing, deploying, and maintaining a robust and efficient system. These requirements ensure that the system operates smoothly and meets the functional and non-functional needs of the users.

3.5.1 Node.js Web Server

Node.js is a crucial component for the backend of the application, providing a scalable and efficient runtime environment for executing JavaScript code server-side. It enables the development of high-performance web servers capable of handling numerous simultaneous connections with minimal resource consumption.

3.5.2 PostgreSQL Database

PostgreSQL is the chosen relational database management system (RDBMS) for this project. It is known for its reliability, robust feature set, and performance. PostgreSQL will manage and store the application's data, offering strong ACID (Atomicity, Consistency, Isolation, Durability) compliance, complex querying capabilities, and support for various data types.

3.5.3 HTTPS Protocol

Security is a paramount concern for the application, and the HTTPS protocol ensures that data transmitted between the client and server is encrypted. This prevents unauthorized access and protects sensitive user information from being intercepted during transmission.

3.5.4 Google Chrome

Google Chrome is the recommended web browser for testing and running the application. Its widespread adoption, robust developer tools, and consistent performance make it an ideal choice for both development and end-user interaction.

3.5.5 ReactJS

React.js is the primary JavaScript library used for building the user interface of the application. Its component-based architecture facilitates the creation of reusable UI components, enhancing the application's modularity and maintainability. React's efficient rendering and state management capabilities contribute to a seamless and responsive user experience.

3.5.6 Tailwind CSS

Tailwind CSS is a utility-first CSS framework that streamlines the process of styling the application's front end. By providing a set of predefined classes, Tailwind CSS allows for rapid and consistent design implementation, reducing the need for custom CSS and ensuring a cohesive look and feel across the application.

3.5.7 Solidity

Solidity is the programming language used for writing smart contracts on the Ethereum blockchain. Given the project's need for blockchain integration, Solidity enables the development of secure, decentralized applications (dApps) with self-executing contracts that operate on the Ethereum network.

3.5.8 Ethereum

Ethereum is a decentralized, open-source blockchain platform that enables the creation of smart contracts and decentralized applications (dApps). It serves as the underlying blockchain infrastructure for this project, providing a secure and transparent environment for executing transactions and storing data.

3.5.9 IPFS (InterPlanetary File System)

IPFS is a peer-to-peer hypermedia protocol designed to make the web faster, safer, and more open. It is used in this project to store and share large files, such as medical records or images, in a decentralized manner. IPFS ensures efficient content distribution and enables data to be accessed even if the original source becomes unavailable.

3.5.10 Ganache

Ganache is a personal blockchain for Ethereum development that can be used to deploy contracts, develop applications, and run tests. It provides a local development environment for testing and debugging smart contracts before deploying them to the main Ethereum network. Ganache simplifies the development process and allows for faster iteration and testing cycles.

3.6 Hardware Requirements

- **Development Workstation**
  - Processor: Intel Core i5 or higher, or AMD equivalent
  - RAM: 16 GB or more
  - Storage: 512 GB SSD or larger
  - Graphics Card: Dedicated GPU with at least 4 GB VRAM (for AI model training and testing)
  - Operating System: Windows, macOS, or Linux
- **User Devices**
  - Any device that can connect to internet
  - A modern browser that can connect to internet
  - Internet connection
  - Metamask wallet account

3.7 System Architecture

In this project, there are three different architectures:

• React JS architecture

• Nodejs architecture

• Blockchain Architecture

3.7.1 React Architecture

Below is a conceptual diagram of the React.js architecture for our project:

• Folder structure.

• Recoil toolkit and context: The store of the data (how to handle data).

• Backend data connectivity (AXIOS for REST API).

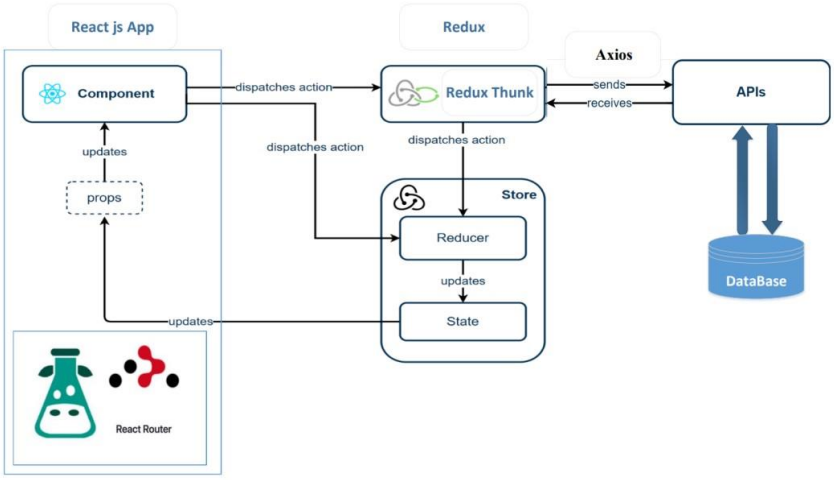


Figure 3.1 React Structure

Screenshot that illustrates the components of React and how does these components communicate

3.7.1.1 Folder Structure:

The following table outlines the folder structure of the React.js application, describing the purpose of each directory and its contents:

Table 3.1 React Folder Structure

Folder / File	Description
`src/`	The root directory for the source code.
`src/components/`	Contains reusable UI components.
`src/store/`	Recoil has a centralized store to store all the states of the applications that are easily accessible by all the app components. Each component does not have to pass the props around within the component tree to access those states, as displayed below. The store folder houses your state management files, such as Recoil, which are used to make certain functions and variables available throughout your application
`src/hooks/`	Custom hooks to encapsulate reusable logic.
`src/pages/`	Contains components representing different pages or views.
`src/styles/`	Contains global styles, theme configurations, and Tailwind CSS configurations.
`src/assets/`	Contains static assets such as images, fonts, and icons.

`src/index.js`	The entry point for the React application.
`src/App.js`	The root component that includes routing and global providers.
`public/`	Contains public assets and the HTML template.
`package.json`	Lists project dependencies and scripts.

3.7.1.2 Recoil:

Recoil is a state management library for React that provides a flexible and efficient way to manage the application's state. It allows for fine-grained control over state and enables easy sharing of state across components. The main concepts in Recoil include atoms and selectors:

- **Atoms**: These are units of state that can be read from and written to. Components can subscribe to atoms to re-render when the atom's state changes.
- **Selectors**: These are pure functions that derive state from atoms or other selectors. They provide a way to compute derived state based on the current state of atoms.

In our application, Recoil is used to manage global state such as user authentication status, theme settings, and other shared data across components.

3.7.1.3 Backend Data Connectivity (AXIOS for REST API):

AXIOS is a popular HTTP client library used for making requests to the backend APIs. It is used in our React application to handle communication with the backend server, enabling data retrieval and submission through RESTful APIs.

The simple data fetching module for REST API connectivity using all the required methods like GET, POST, PUT, PATCH, DELETE. Ways of fetching data from an API:

• Fetch API

• AXIOS

In both APIs the result is same but there are some differences.

Why do we use Axios?

• Fetch API does not directly convert the data to json format. We have to tell it.

• But Axios converts data to json format for better readability.

• Axios has built-in XSRF (Cross-Site Request Forgery) protection, while Fetch does not.

• Axios has the ability to intercept HTTP requests but Fetch, by default, does not.

• Axios allows canceling requests and request timeout but fetch does not.

By utilizing Recoil for state management and AXIOS for backend connectivity, our React application maintains a clean and efficient architecture, ensuring smooth interaction between the frontend and backend components.

3.7.2 NodeJS Architecture

Node.js is a powerful, open-source, server-side runtime environment that executes JavaScript code outside of a browser. It is built on the V8 JavaScript engine and provides an event-driven, non-blocking I/O model, making it efficient and scalable for building fast, data-intensive applications.

Express.js, a minimal and flexible Node.js web application framework, provides a robust set of features for web and mobile applications. It is designed to build single-page, multi-page, and hybrid web applications, facilitating the management of various web application functionalities such as routing, middleware, and views.

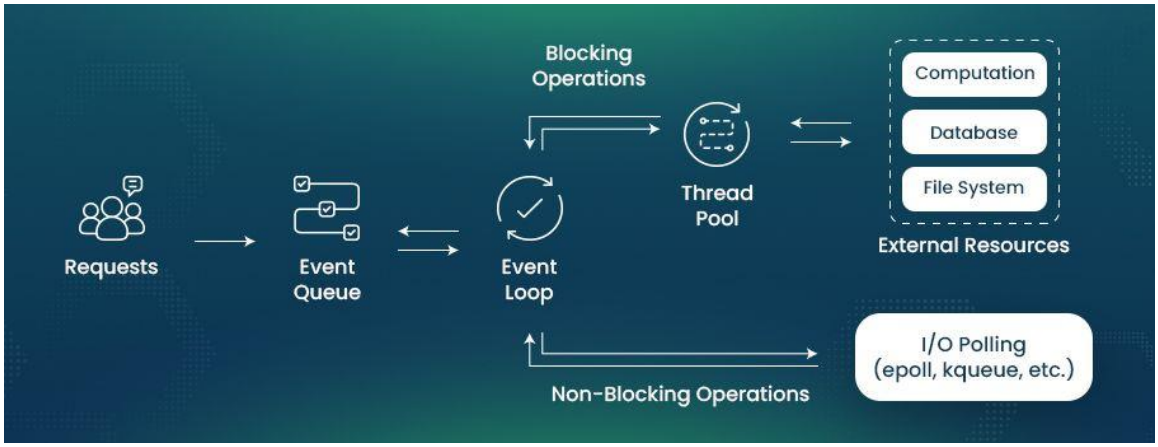


Figure 3.2 Nodejs architecture

Screenshot that illustrates the flow of work of NodeJS

3.7.3 Blockchain Architecture:

In this project, we employ Ethereum as the blockchain platform, Solidity for smart contract development, and MetaMask for interacting with the blockchain. This architecture leverages the strengths of Ethereum's decentralized network, Solidity's robust smart contract capabilities, and MetaMask's seamless integration for end-user interactions.

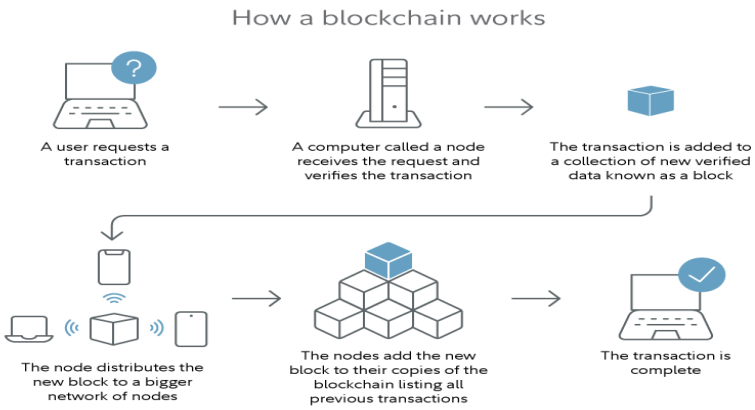


Figure 3.3 Blockchain architecture

1. **Ethereum Blockchain**
    - Ethereum is a decentralized, open-source blockchain platform that enables the creation of smart contracts and decentralized applications (dApps). It uses a consensus algorithm called Proof of Stake (PoS) to validate transactions and secure the network.
    - Ethereum's main features include:
        - **Smart Contracts**: Self-executing contracts with the terms directly written into code.
        - **EVM (Ethereum Virtual Machine)**: A Turing-complete virtual machine that executes smart contracts.
        - **Gas**: A unit that measures the amount of computational effort required to execute operations.
2. **Solidity**
    - Solidity is a high-level programming language designed for writing smart contracts on the Ethereum blockchain. It is statically typed and supports complex data structures and operations.
    - Key aspects of Solidity:
        - **Contract**: The basic building block of Ethereum applications, containing data (state) and functions (code).

        - **Modifiers**: Functions that can change the behavior of other functions.
        - **Events**: Mechanisms for logging that allow smart contracts to communicate with the frontend.
3. **MetaMask**
    - MetaMask is a browser extension and mobile app that functions as a cryptocurrency wallet and a gateway to Ethereum-based dApps. It allows users to manage their Ethereum private keys and interact with smart contracts and dApps directly from the browser.
    - MetaMask features:
        - **Account Management**: Creation and management of multiple Ethereum accounts.
        - **Transaction Signing**: Securely signing transactions and interactions with smart contracts.
        - **Integration**: Seamless integration with dApps through web3.js.

3.7.4 Artificial Intelligence providers In healthcare

Table 3.2 Artificial Intelligence Providers Comparison

Name	Ada Health	Buoy Health	IBM Watson Health	Gemini
advantages	Offers personalized insights, assesses risk levels, and provides recommendations based on symptoms, medical history, and risk factors.	Analyzes symptoms and medical history to provide personalized recommendations using a vast medical information database.	Facilitates remote health monitoring, offers personalized cancer treatment recommendations, and matches patients to clinical trials based on their health profiles and medical history.	The model's multimodal capability enhances healthcare accuracy and efficiency by generating radiology reports, interpreting medical images, and answering complex clinical questions in various scenarios like diagnostics and patient monitoring.
Disadvantages	There is a risk of misdiagnosis with incomplete or inaccurate information, and it may lack information on rare conditions.	It may not address all health issues or rare conditions, and the virtual interface can't fully replace face-to-face expertise and empathy.	Potential for biased recommendations.	limits accessibility, especially for smaller healthcare providers or institutions with limited technological infrastructure
Plan	paid	paid	paid	free

We chose Gemini as our AI provider because of the advantages it gives us and because it’s API is free

3.8 System Modeling

3.7.1 Use-case Diagrams

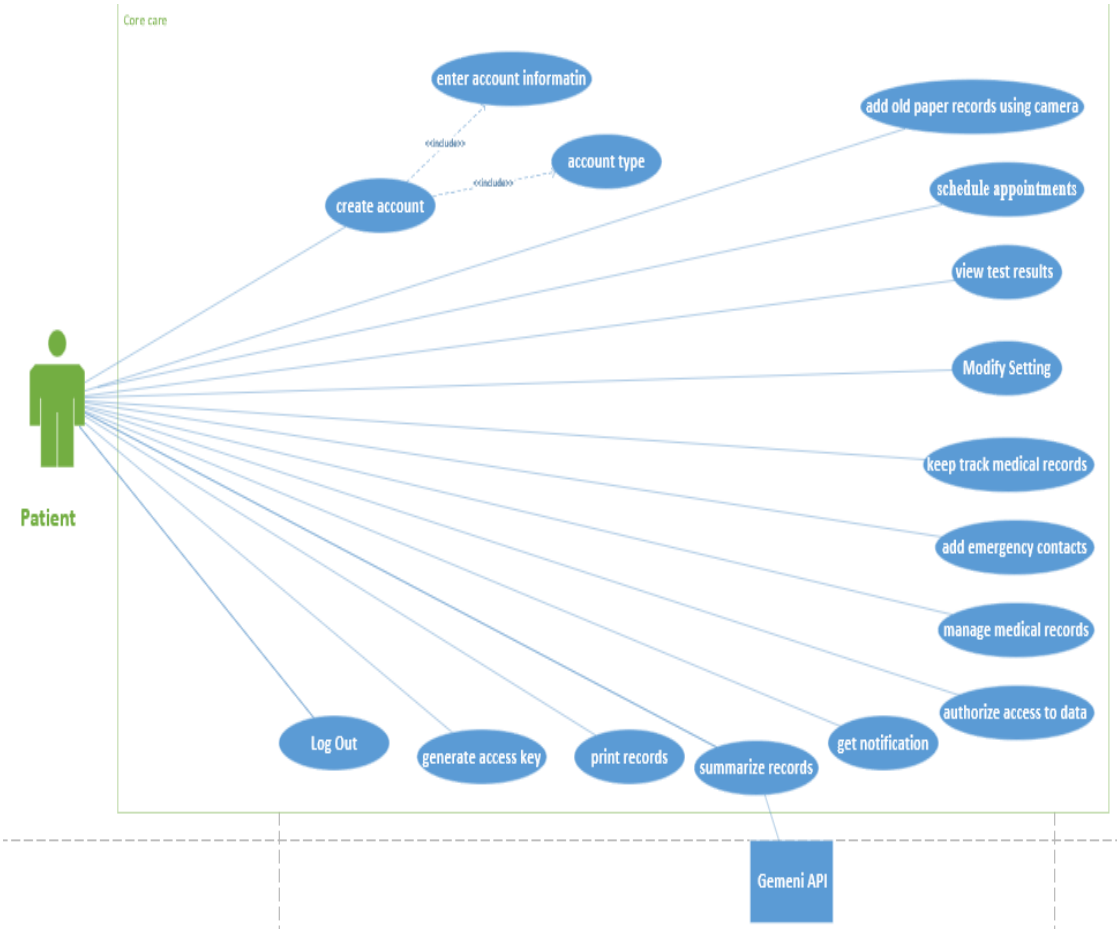


Figure 3.4 Patient Use Case Diagram

Screenshot of patient usecase diagram that demonstrates what a patient can do on our platform

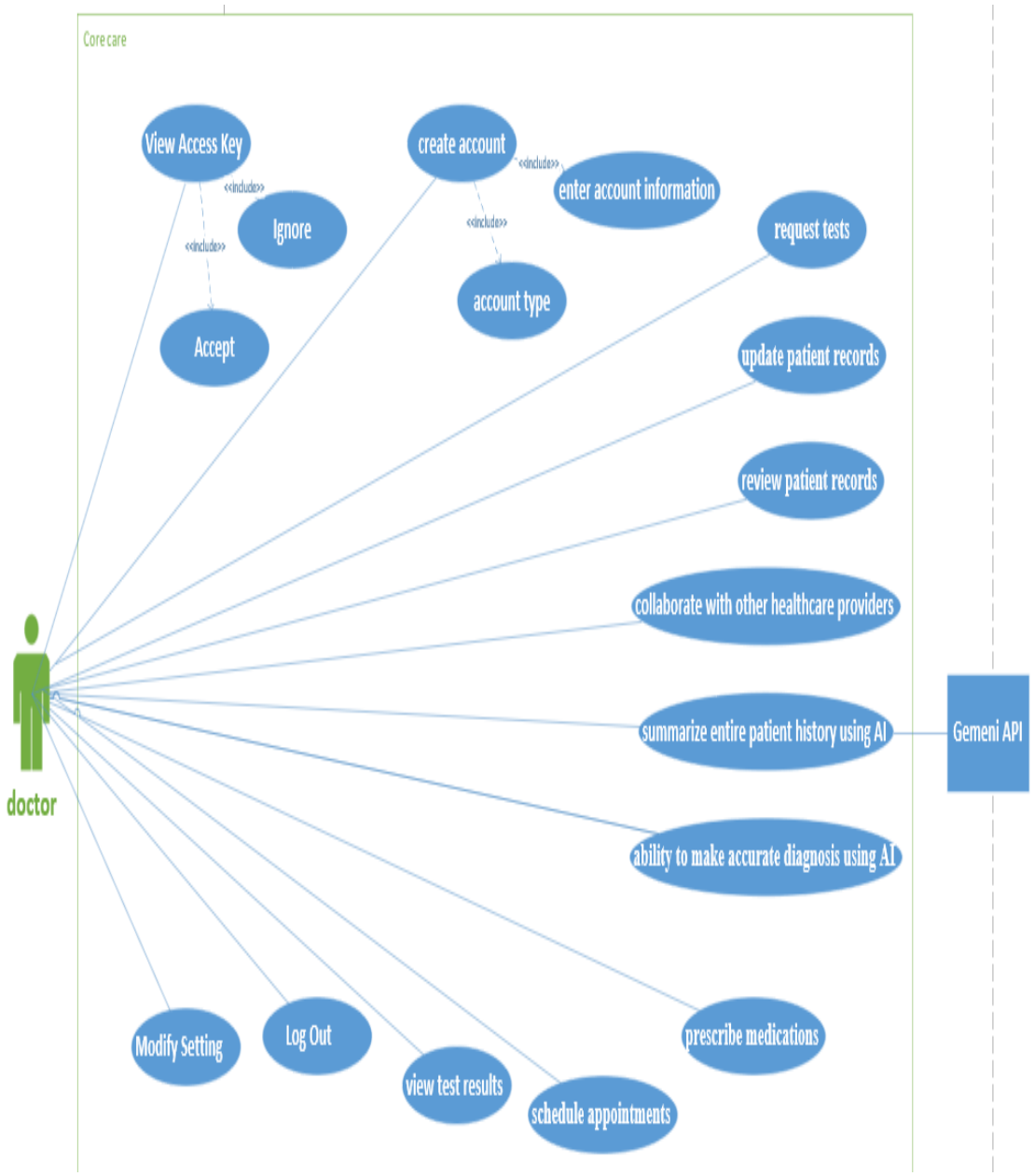


Figure 3.5 Doctor Use Case Diagram

Screenshot of doctor usecase diagram that demonstrates what a doctor can do on our platform

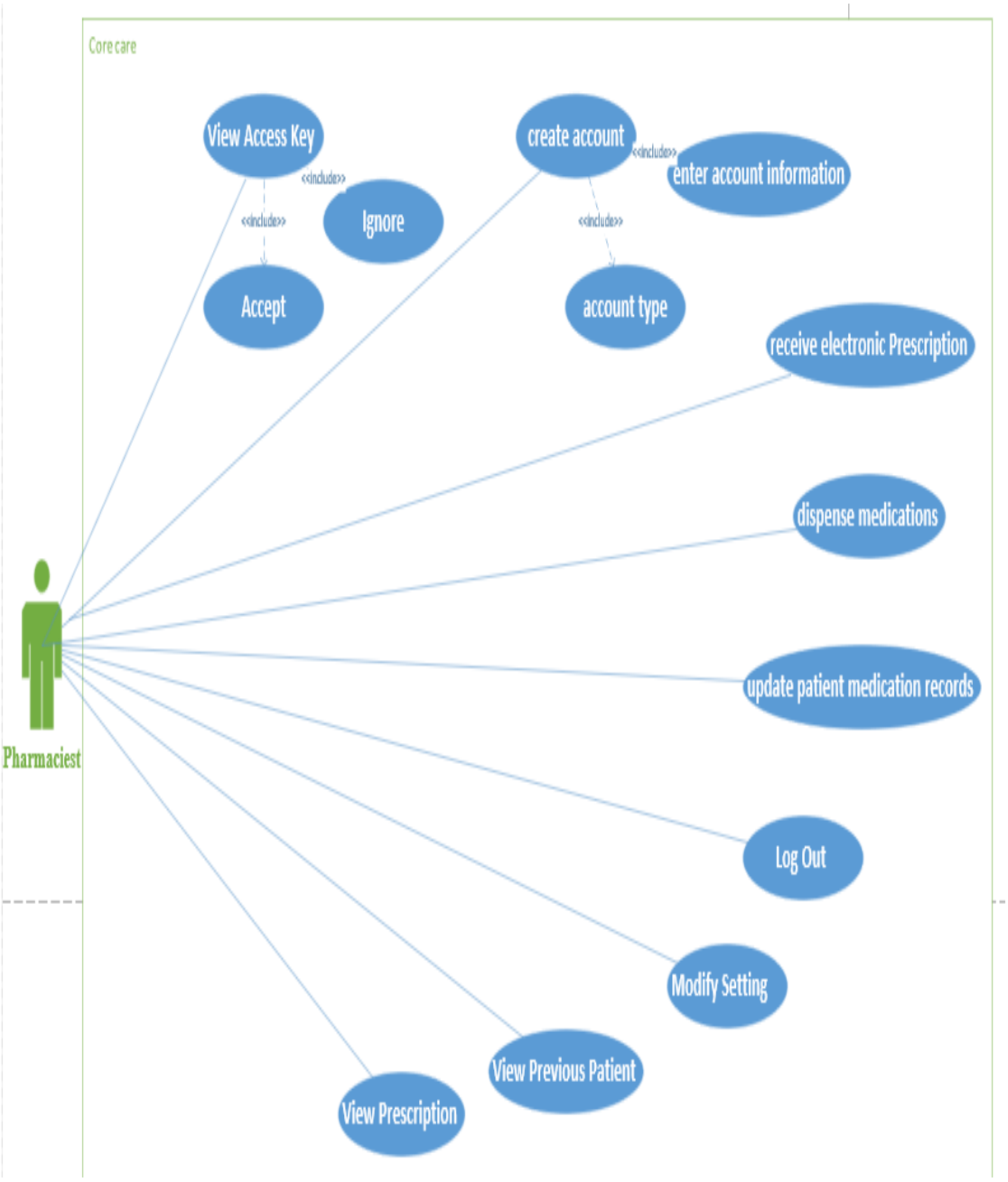


Figure 3.6 Pharmaciest Use Case Diagram

Screenshot of pharmaciest usecase diagram that demonstrates what a pharmaciest can do on our platform

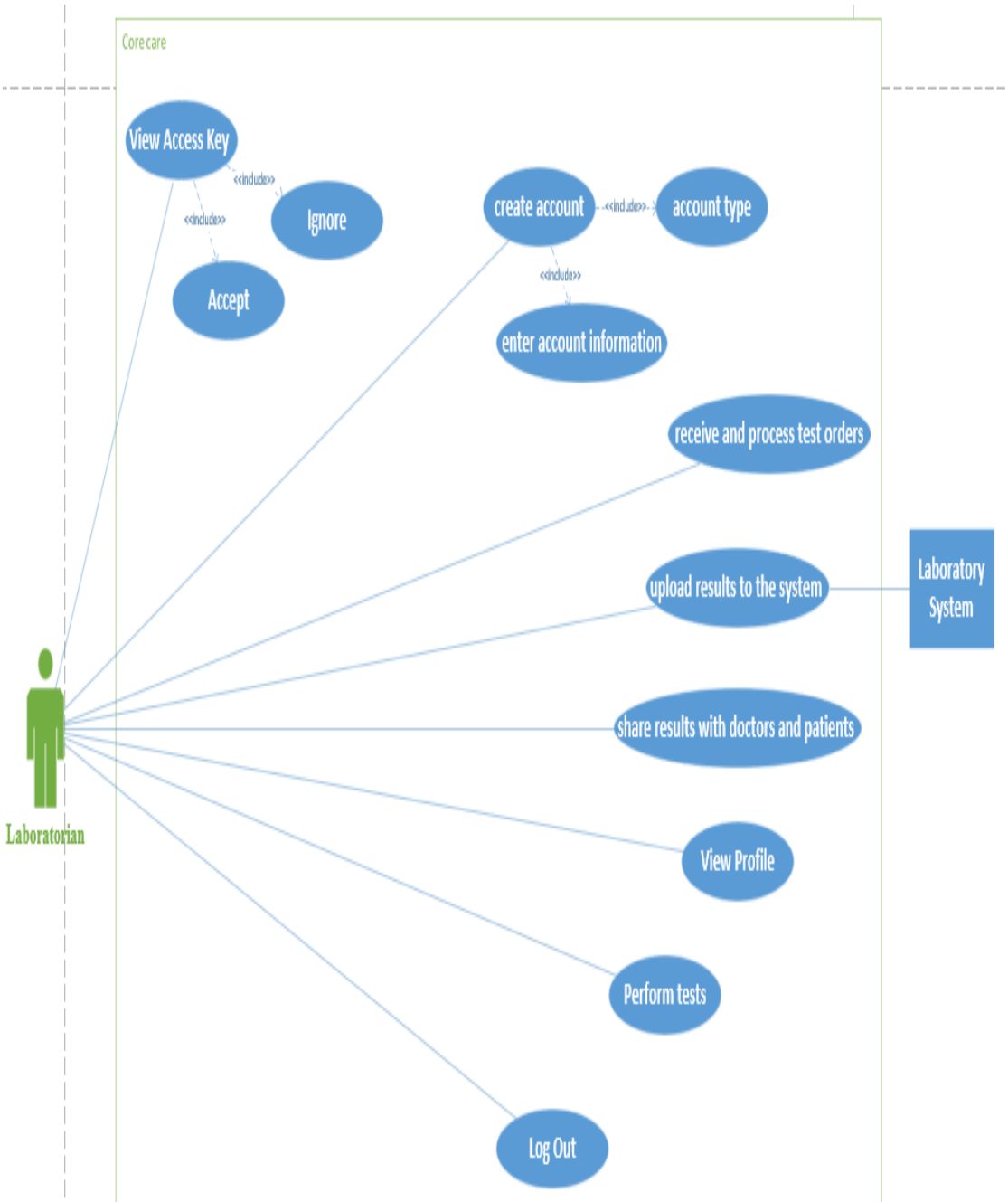


Figure 3.7 Laboratorian Use Case Diagram

Screenshot of laboratorian usecase diagram that demonstrates what a laboratorian can do on our platform

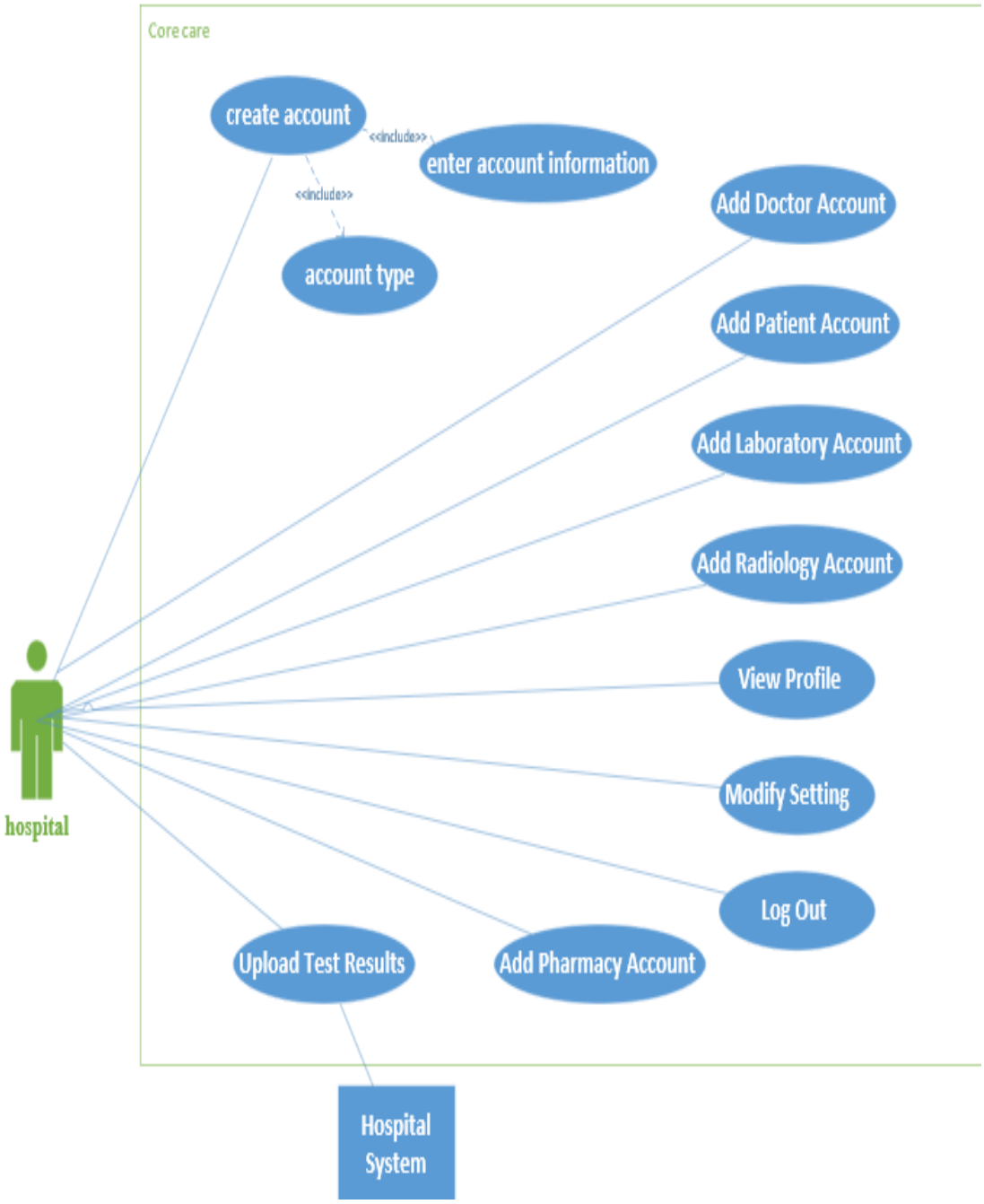


Figure 3.8 Hospital Use Case Diagram

Screenshot of Hospital usecase diagram that demonstrates what a Hospital can do on our platform

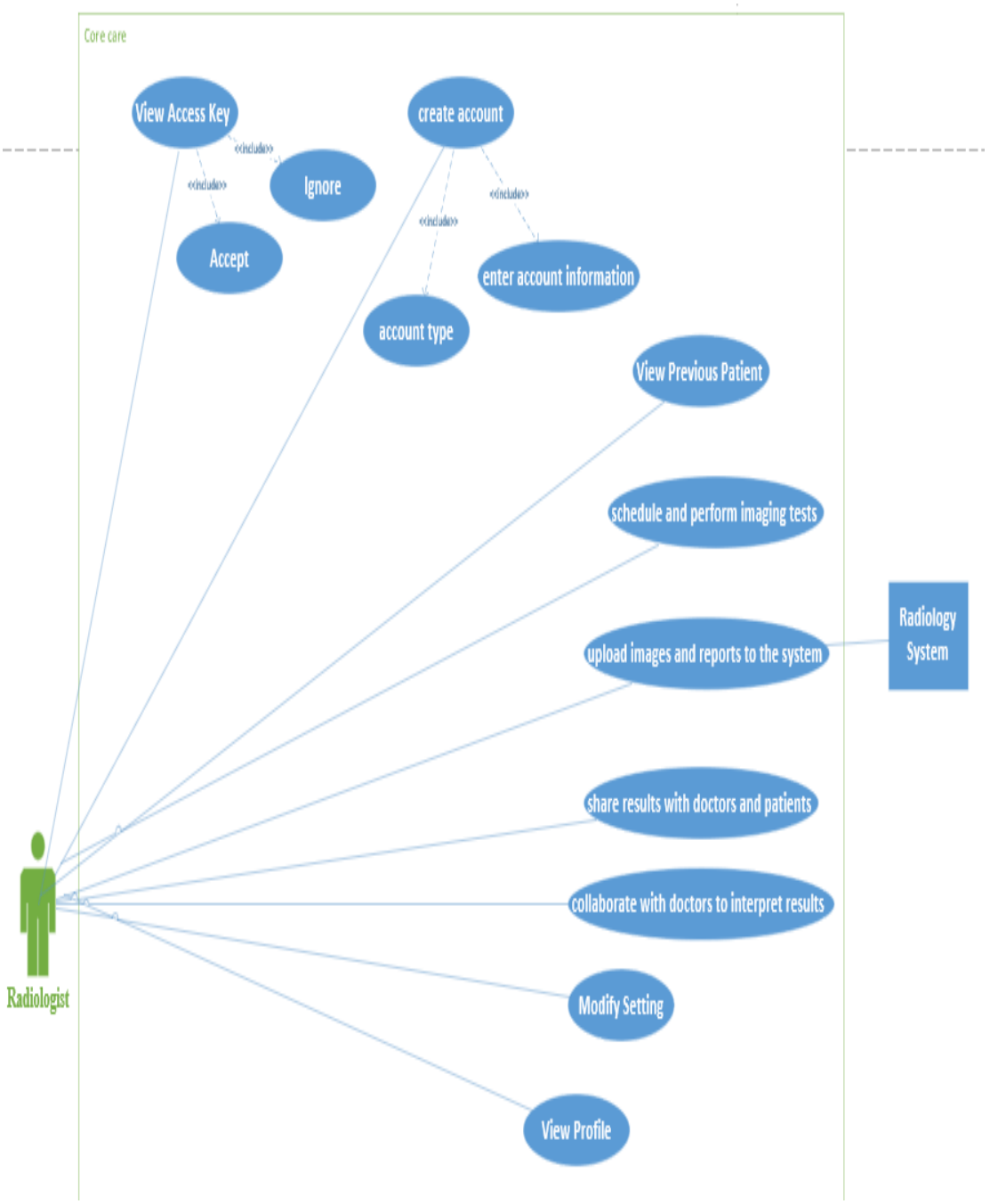


Figure 3.9 Radiologist Use Case Diagram

Screenshot of radiologist usecase diagram that demonstrates what a radiologist can do on our platform

3.7.2 Some Sequence Diagrams

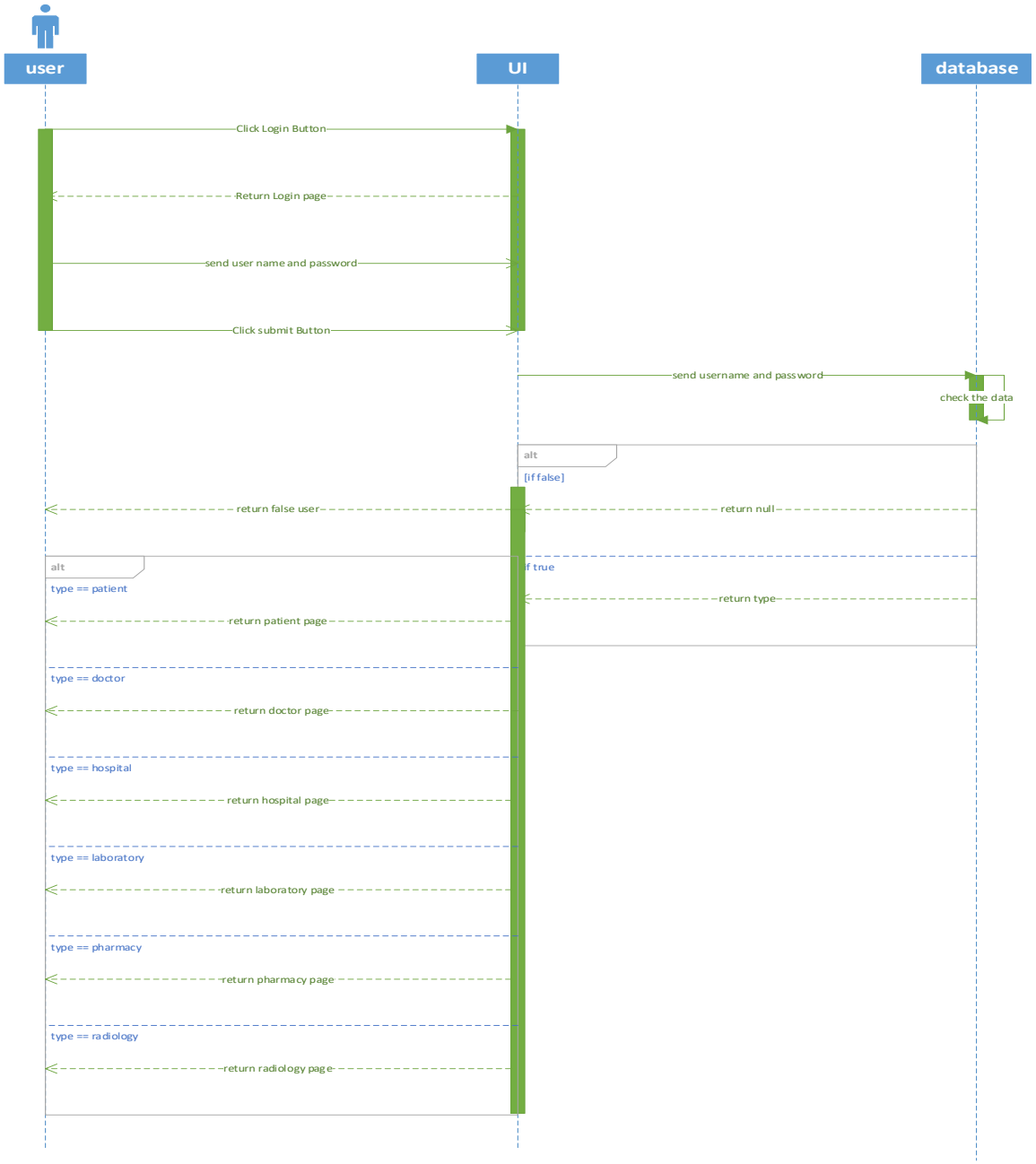


Figure 3.10 Log in sequence

Screenshot of Login sequence diagram that illustrates the process of user login

Figure 3.11 Create Account sequence

Screenshot of Signup sequence diagram that illustrates the process of user signup

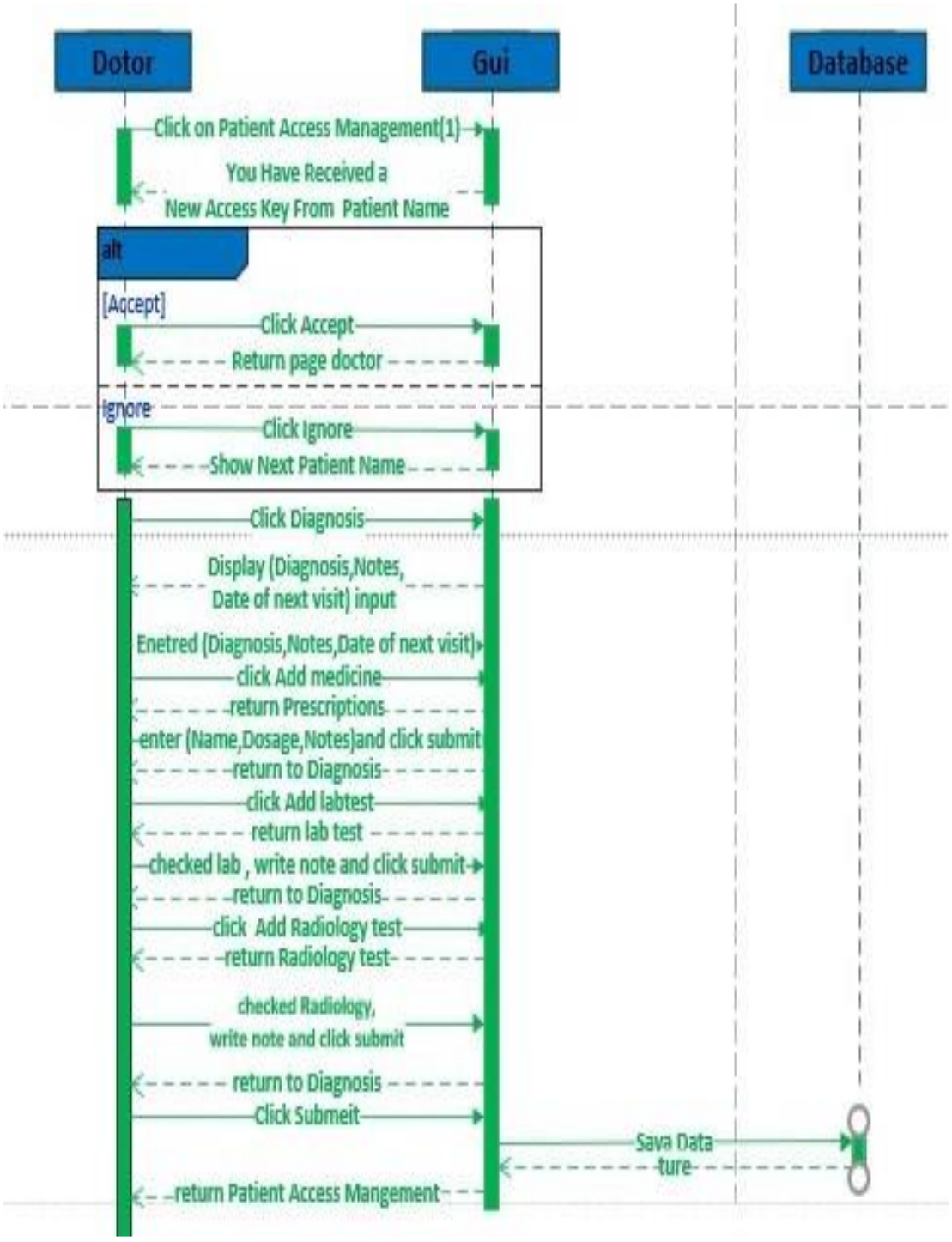


Figure 3.12 insert information diagnosis sequence

Screenshot of Login sequence diagram that illustrates the process of user login

3.7.3 Activity Diagrams

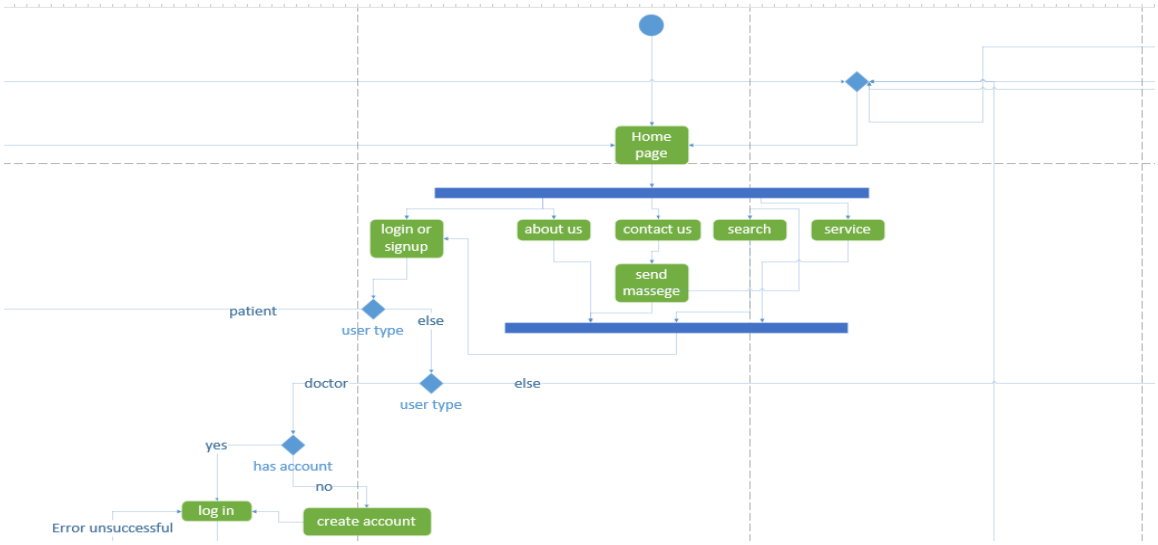


Figure 3.13 Create Account Activity

Screenshot of create account activity diagram that illustrates the flow of the process

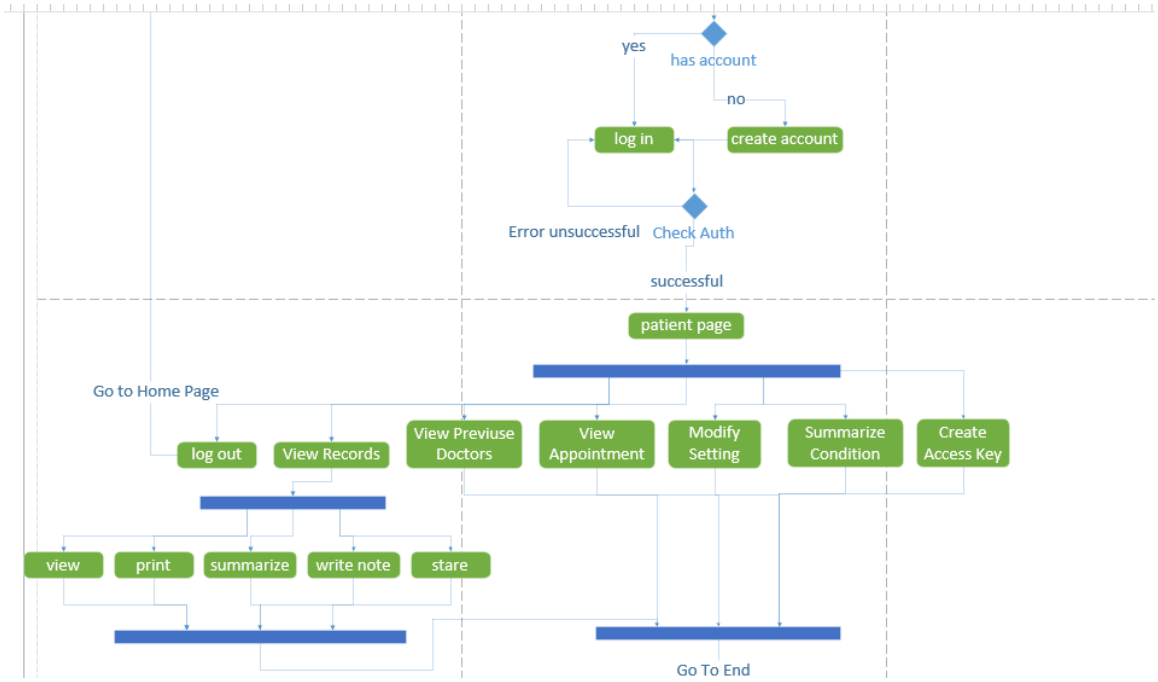


Figure 3.14 Patient Activity Diagram

Screenshot of patient activity diagram that illustrates the flow of the processes

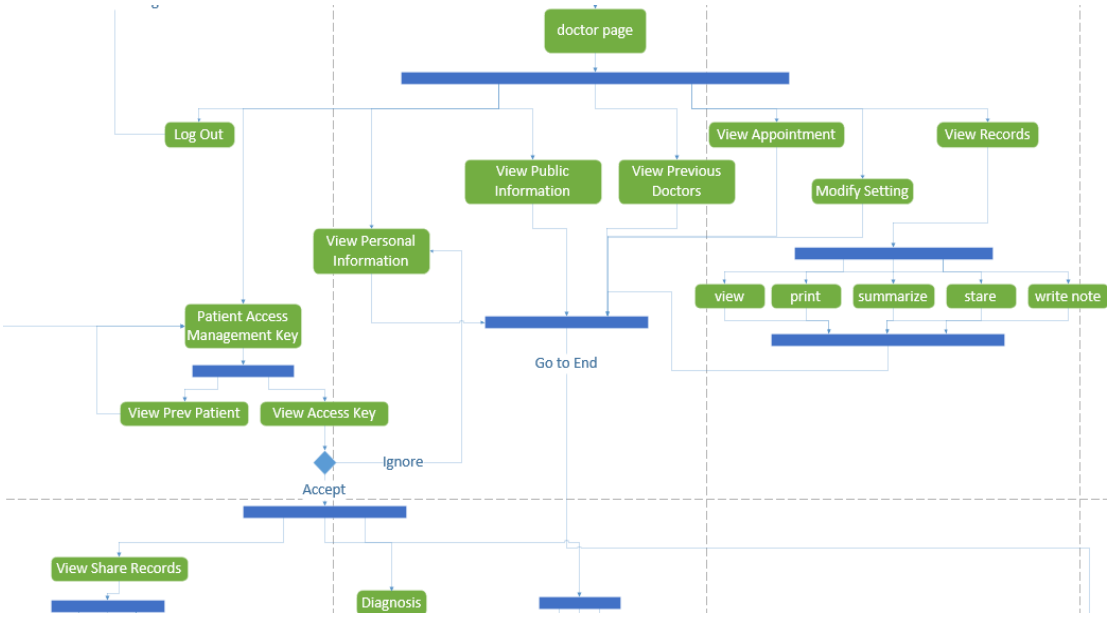


Figure 3.15 Doctor Activity Diagram

Screenshot of doctor activity diagram that illustrates the flow of the processes as a normal user

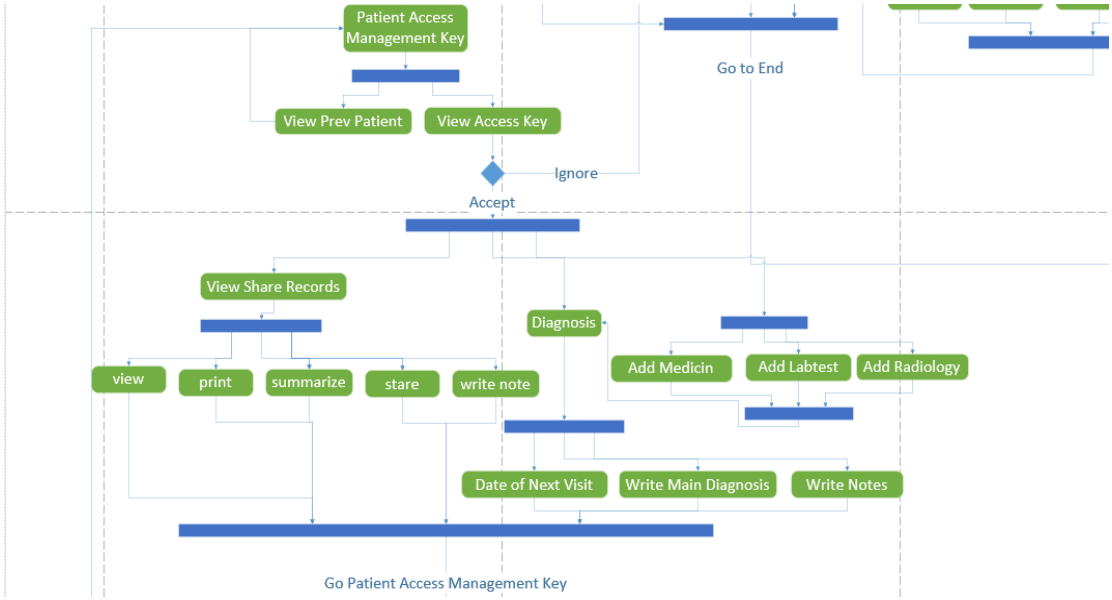


Figure 3.16 Doctor Activity Diagram

Screenshot of doctor activity diagram that illustrates the flow of the processes as a doctor

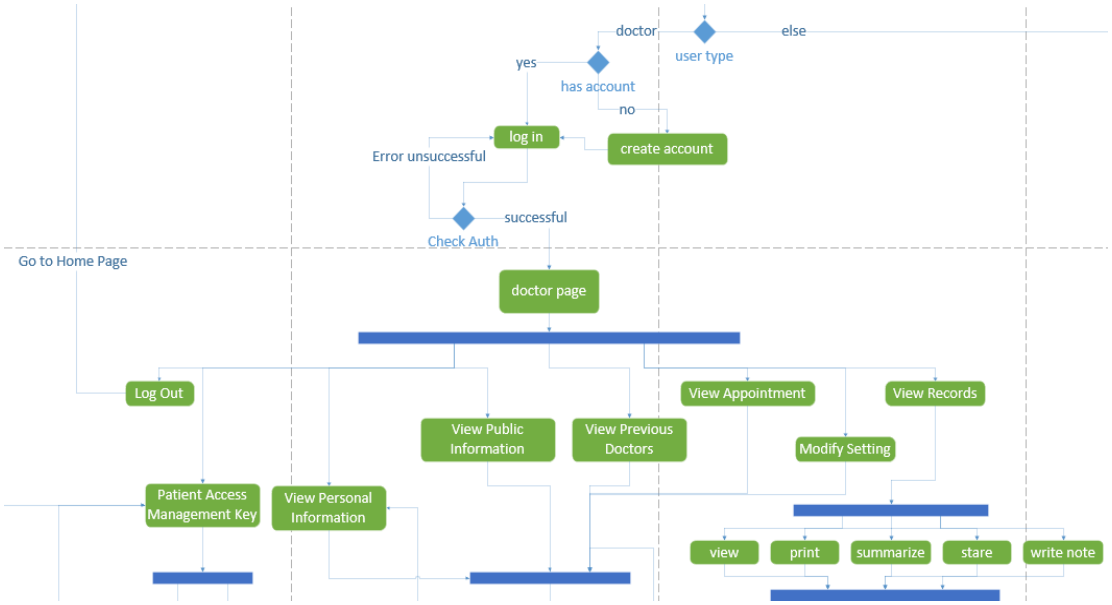


Figure 3.17 Doctor Activity Diagram

Screenshot of doctor activity diagram that illustrates the flow of the diagnosis process

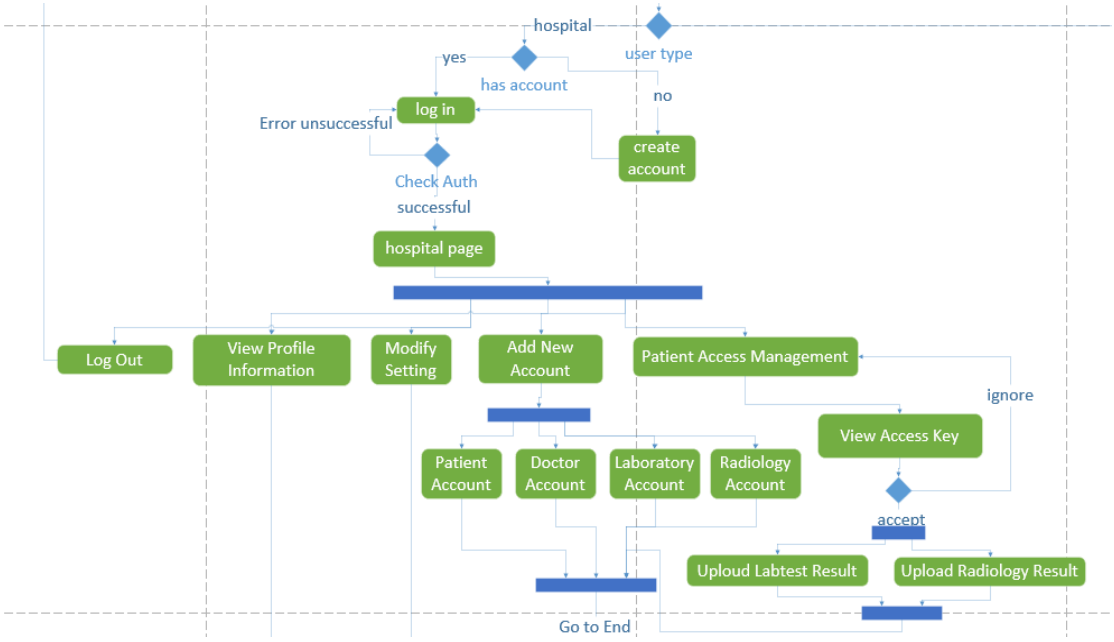


Figure 3.18 Hospital Activity Diagram

Screenshot of Hospital activity diagram that illustrates the flow of the processes

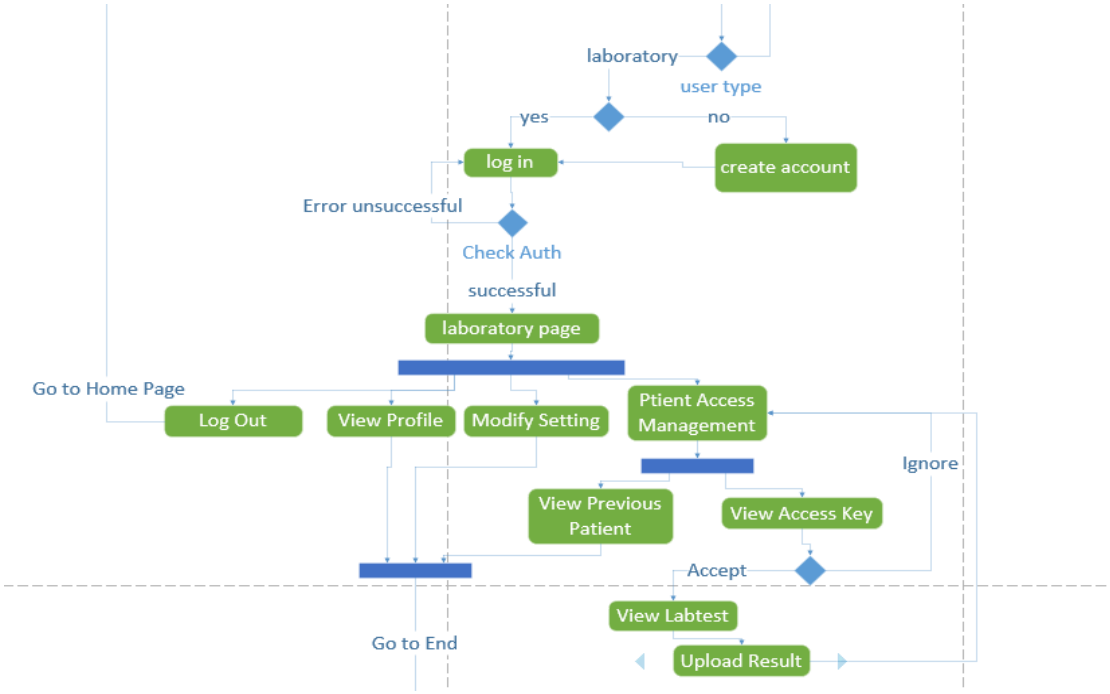


Figure 3.19 Laboratorian Activity Diagram

Screenshot of Laboratorian activity diagram that illustrates the flow of the processes

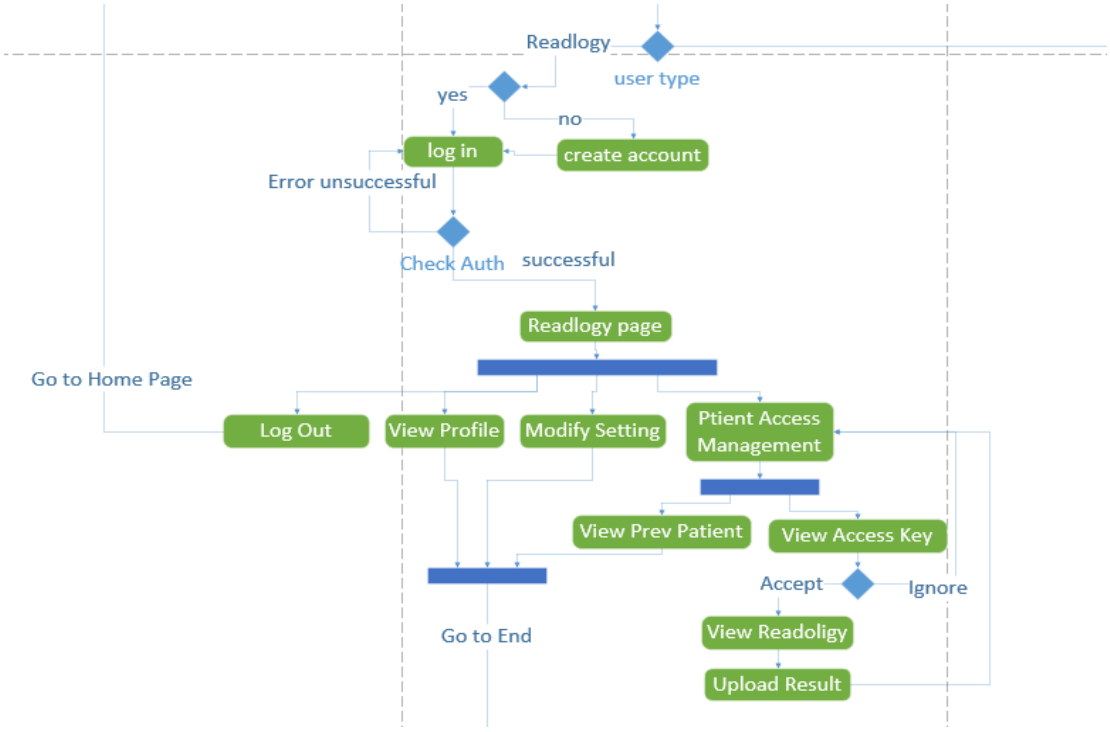


Figure 3.20 Radiologist Activity Diagram

Screenshot of Radiologist activity diagram that illustrates the flow of the processes

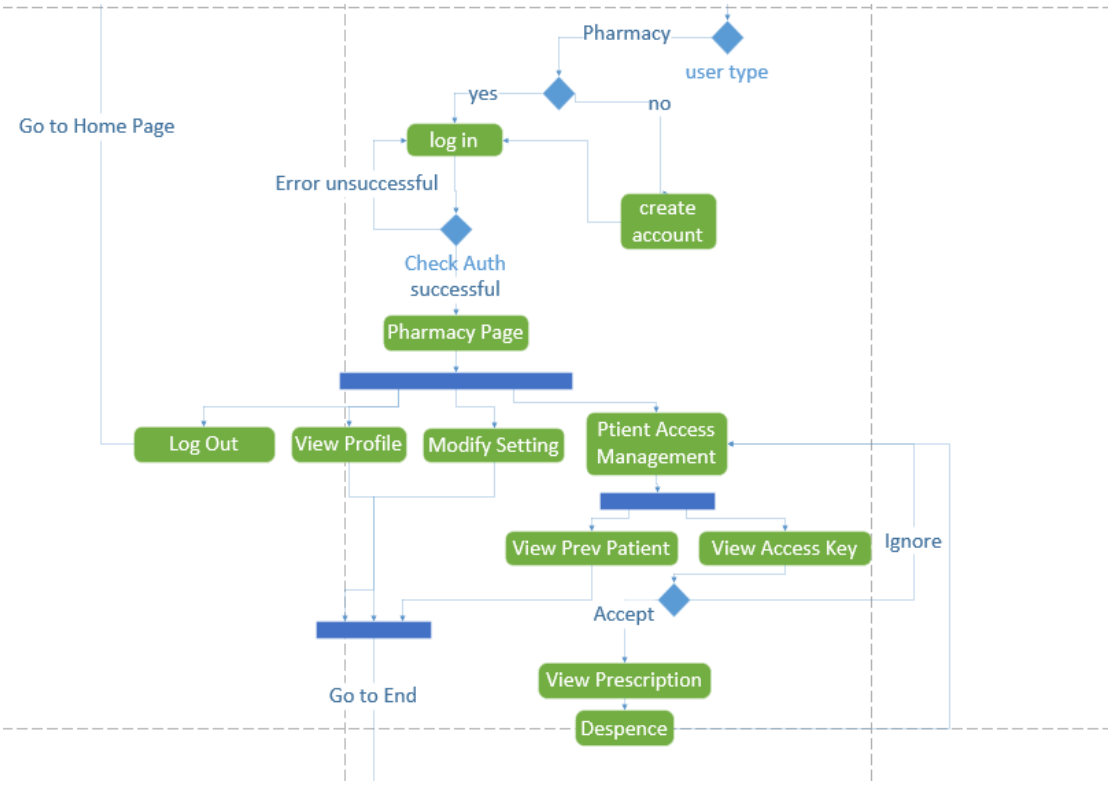


Figure 3.21 Pharmacist Activity Diagram

Screenshot of pharmacist activity diagram that illustrates the flow of the processes

3.7.4 System Architecture Diagram

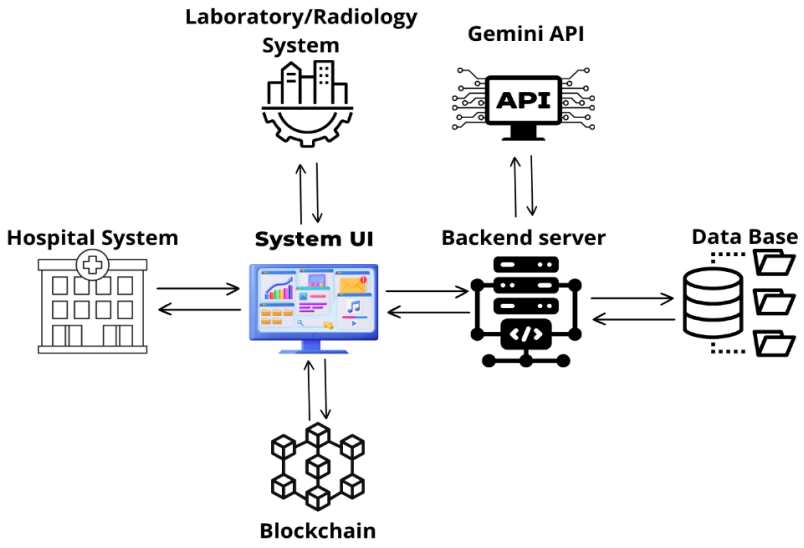


Figure 3.22 System Architecture

Screenshot of System Architecture Diagram that shows our system components and the outer components that communicate with our system from an abstract level

Chapter 4: Design

Chapter 4: Design

4.1 Introduction

The Design section of this documentation outlines the transformation of system specifications, requirements, and behavioral models into a coherent and operational structure. This section serves as the blueprint for the development team, providing a detailed guide to ensure the project meets the specified objectives effectively.

1.2 User Interface Design

1.2.1 Home Page

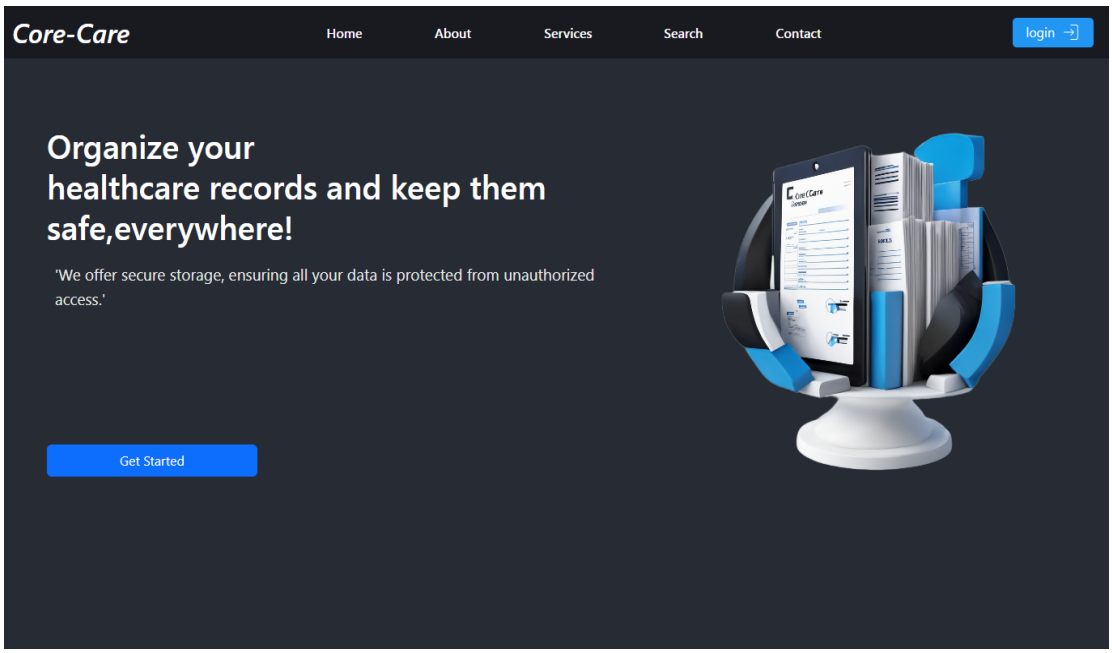


Figure 4.1 Screenshot of the Home page without login in.

The main interface of the system is available to all users and can be accessed through the web application. It is the first interface that appears when you visit the site. The interface contains the objectives of the system, the advantages of the system, how and how easy it is to register in it, and the disadvantages and risks of previous traditional systems

1.2.2 Register Page

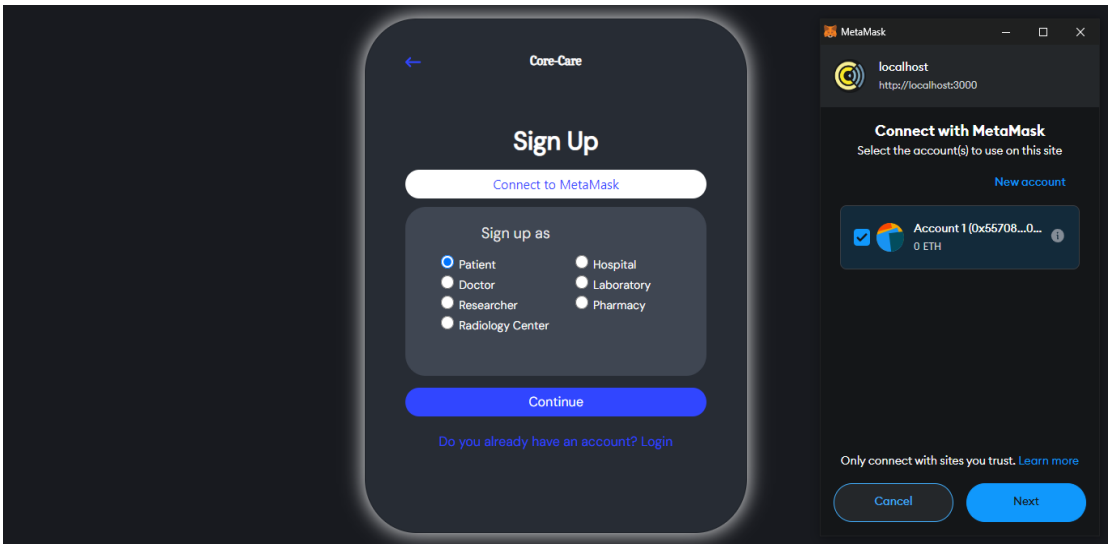


Figure 4.2 Screenshot of the register Interface.

This interface allows the visitors to create new accounts. It contains check boxes through which the user selects the type of account to be created.

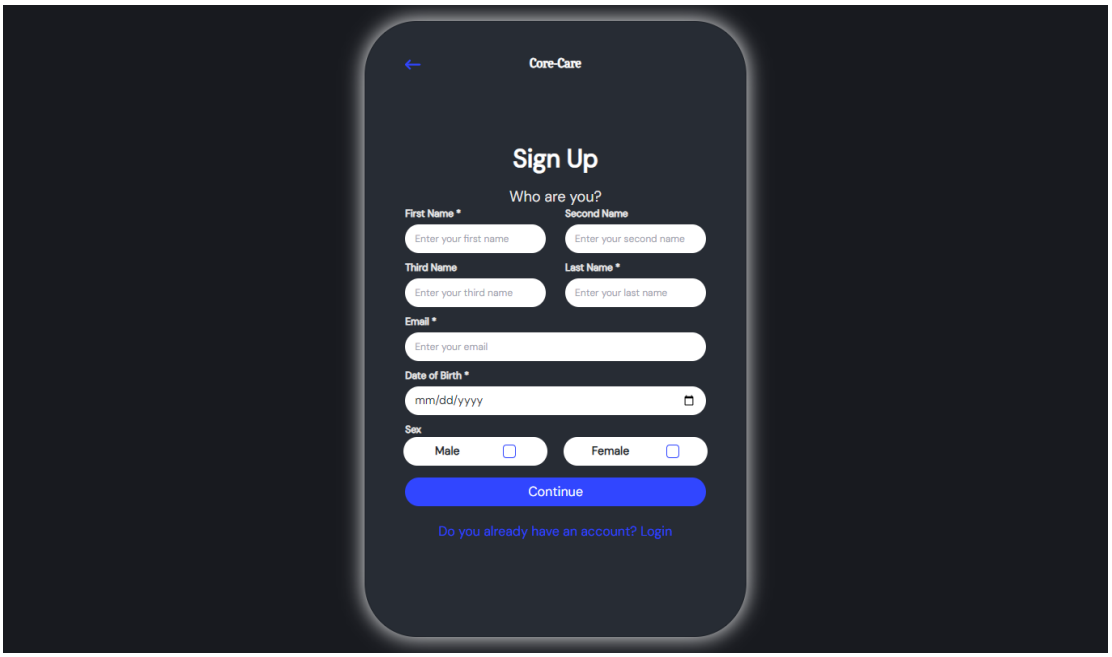


Figure 4.3 Screenshot of the register Interface step1/5.

This interface asks the user to enter their general information as fullname, email, date of birth and sex.

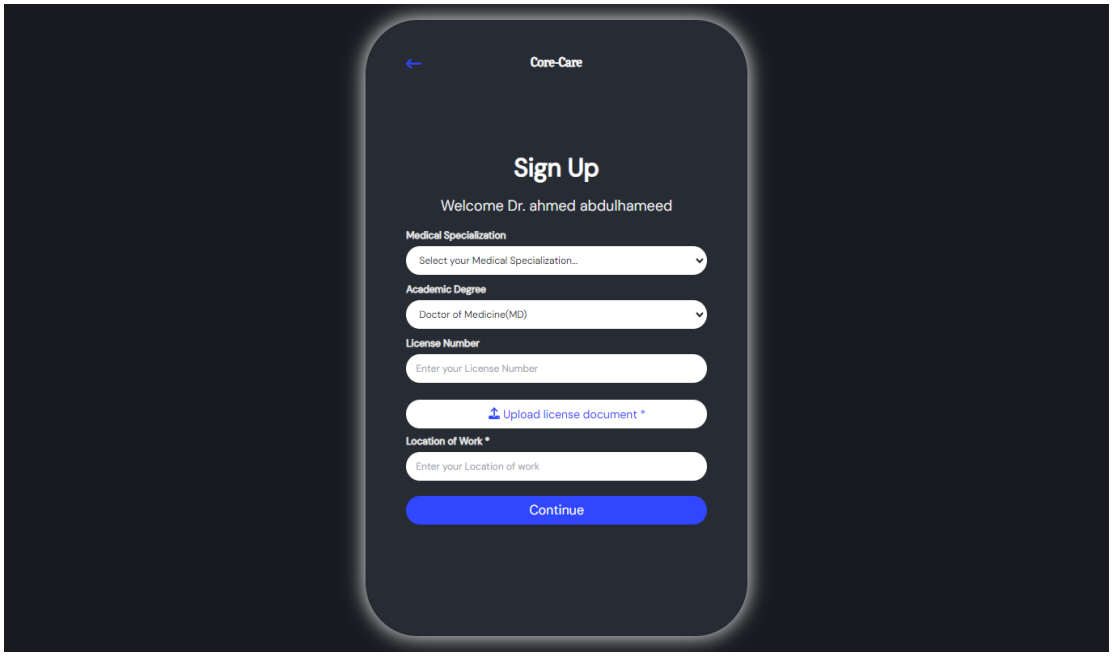


Figure 4.4 Screenshot of the register Interface step2/5.

In these interfaces, personal data is entered for sign up to a create a new account.

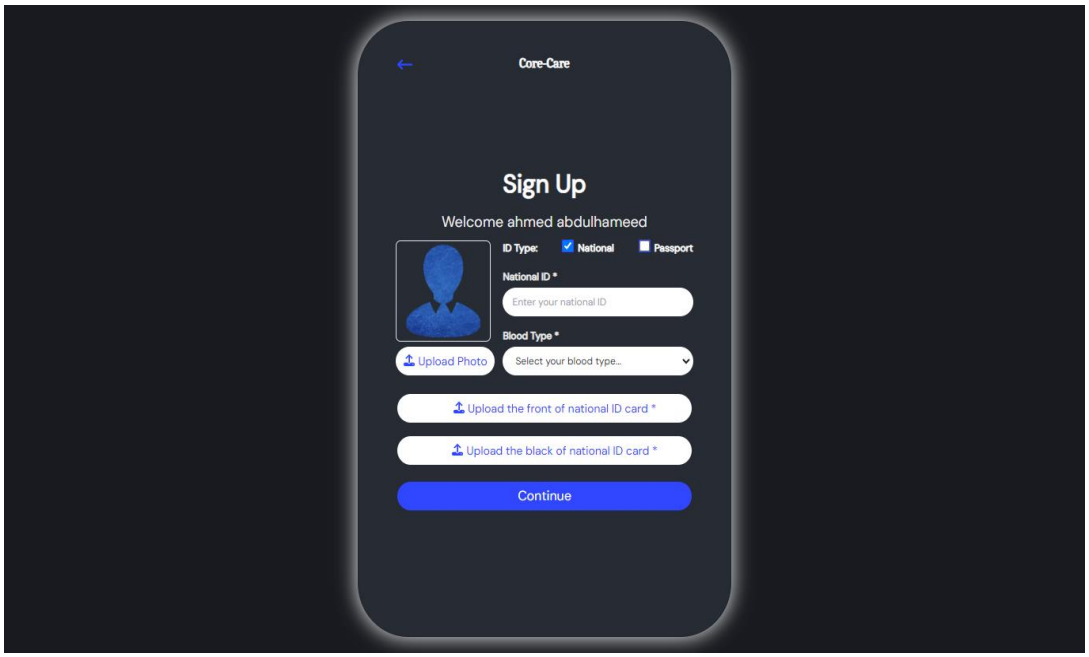


Figure 4.5 Screenshot of the register Interface step3/5.

In this interface, personal documents are uploaded in order to verify identity

Figure 4.6 Screenshot of the register Interface.

This page is intended for doctors to add specialty, license, and workplace

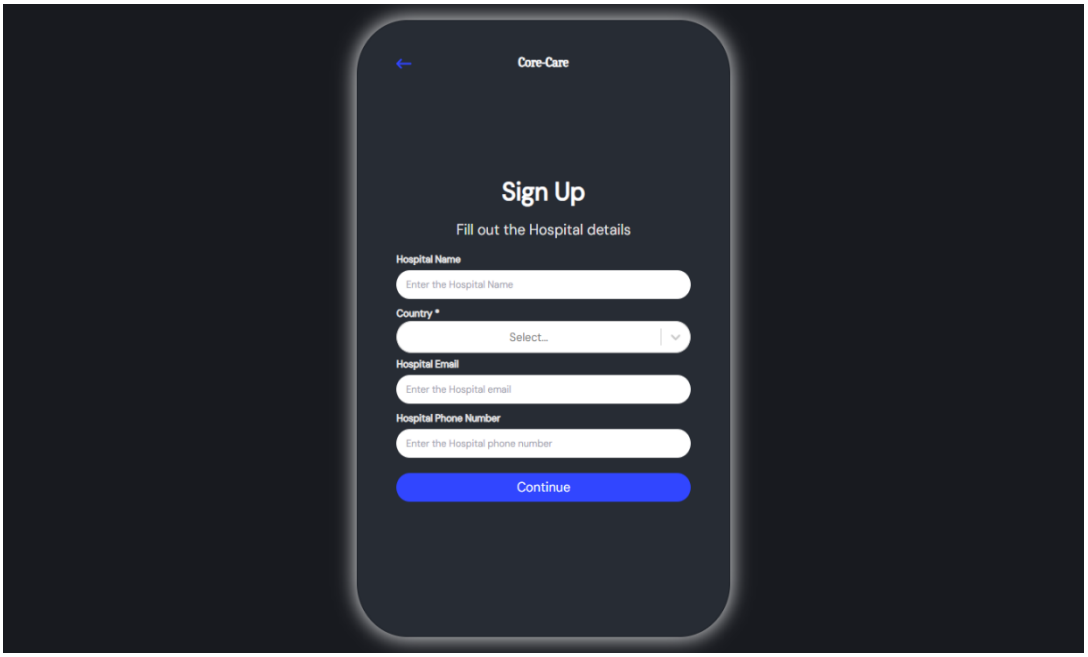


Figure 4.7 Screenshot of the register Interface.

Through this interface, the healthcare organization's general data and information are entered to create the account

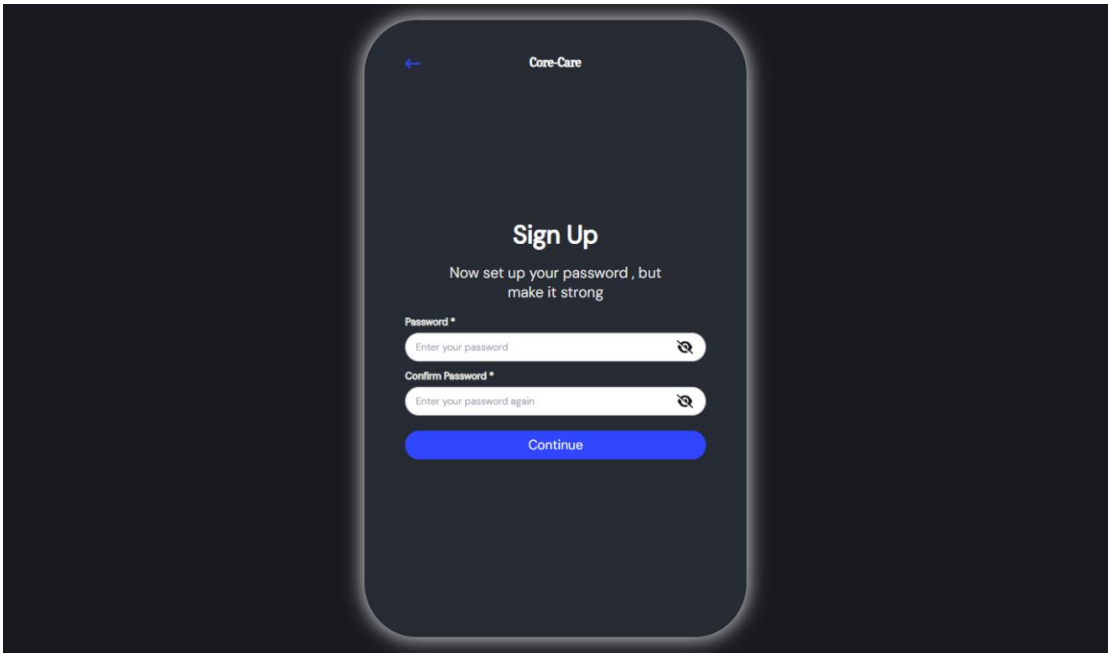


Figure 4.8 Screenshot of the register Interface.

In this interface, the password is generated and confirmed

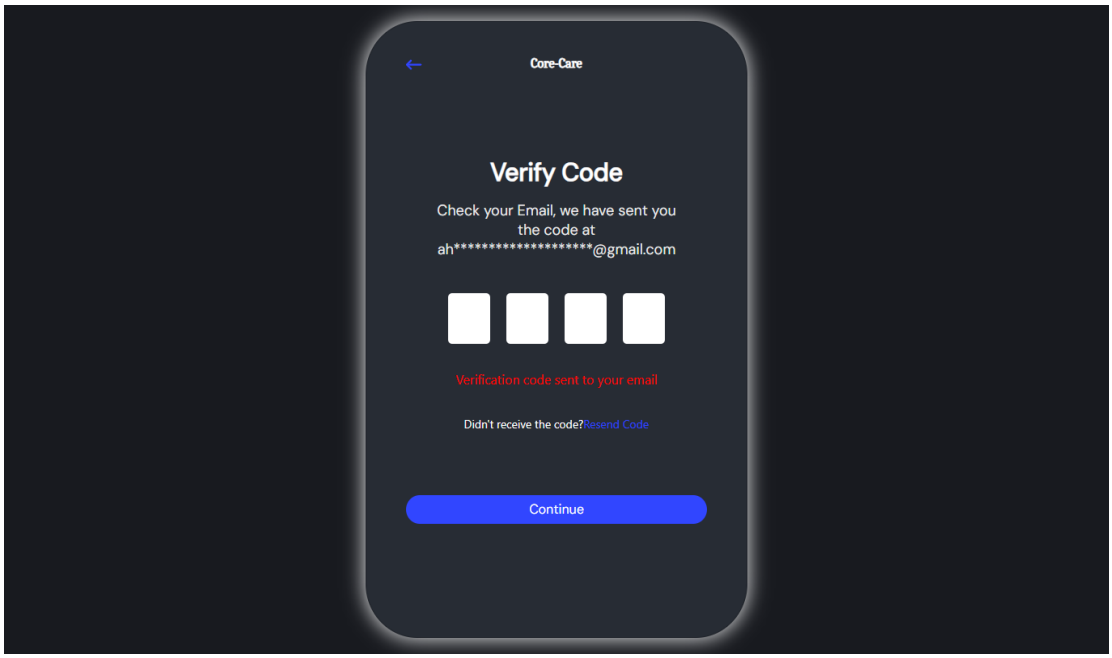


Figure 4.9 Screenshot of the register Interface.

In this interface, the verification code sent by the system to the user's email is entered to complete the registration process

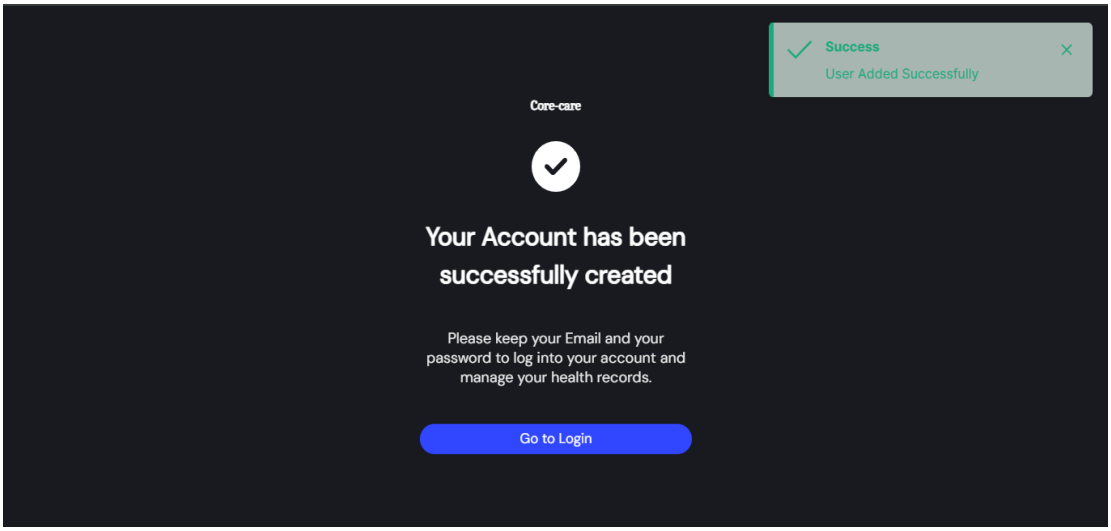


Figure 4.10 Screenshot of the register Interface.

Through this interface, the user is notified that the account has been created successfully

1.2.3 User profile

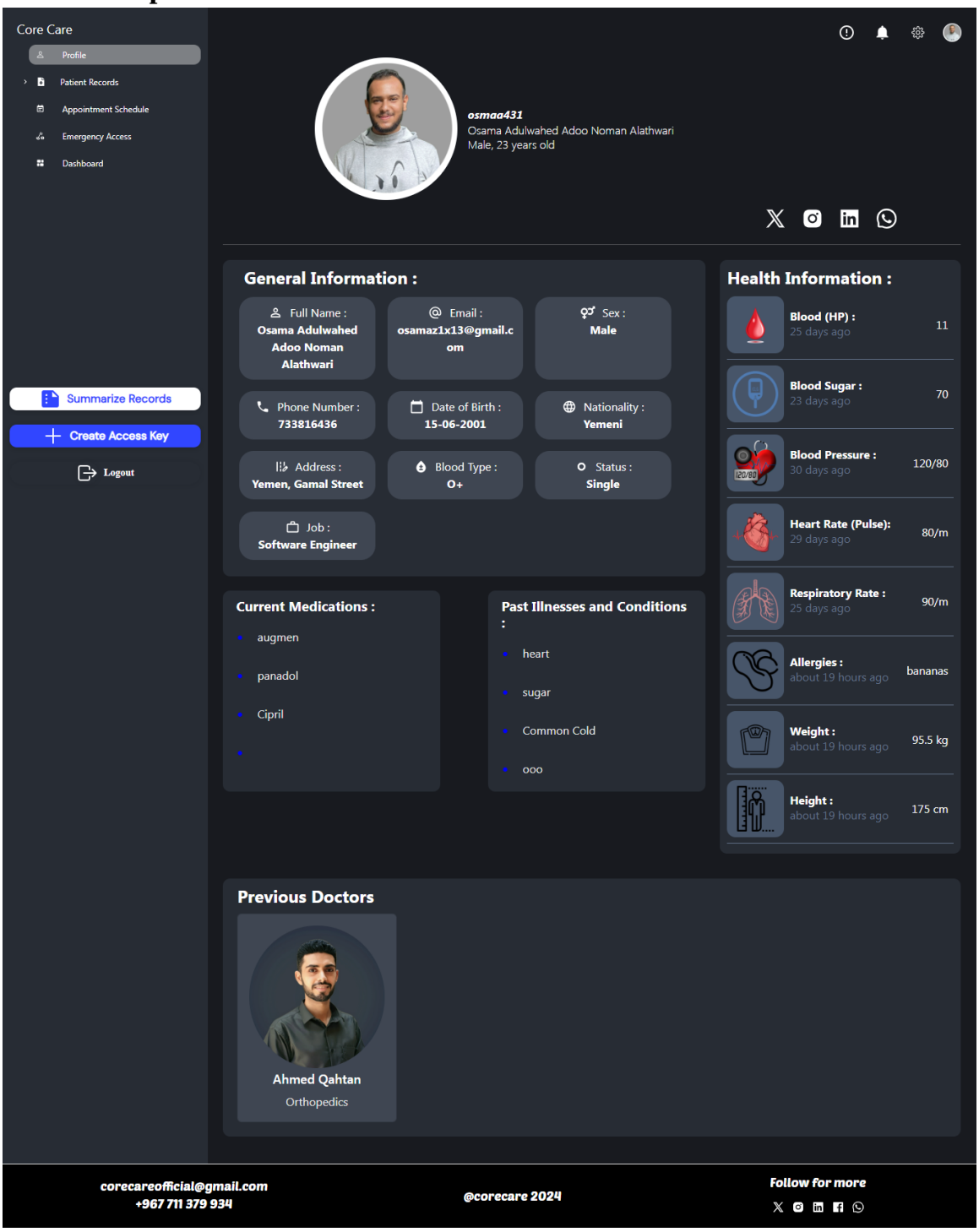


Figure 4.11 Screenshot of the user profile Interface.

The patient's personal and health information is displayed on this page.

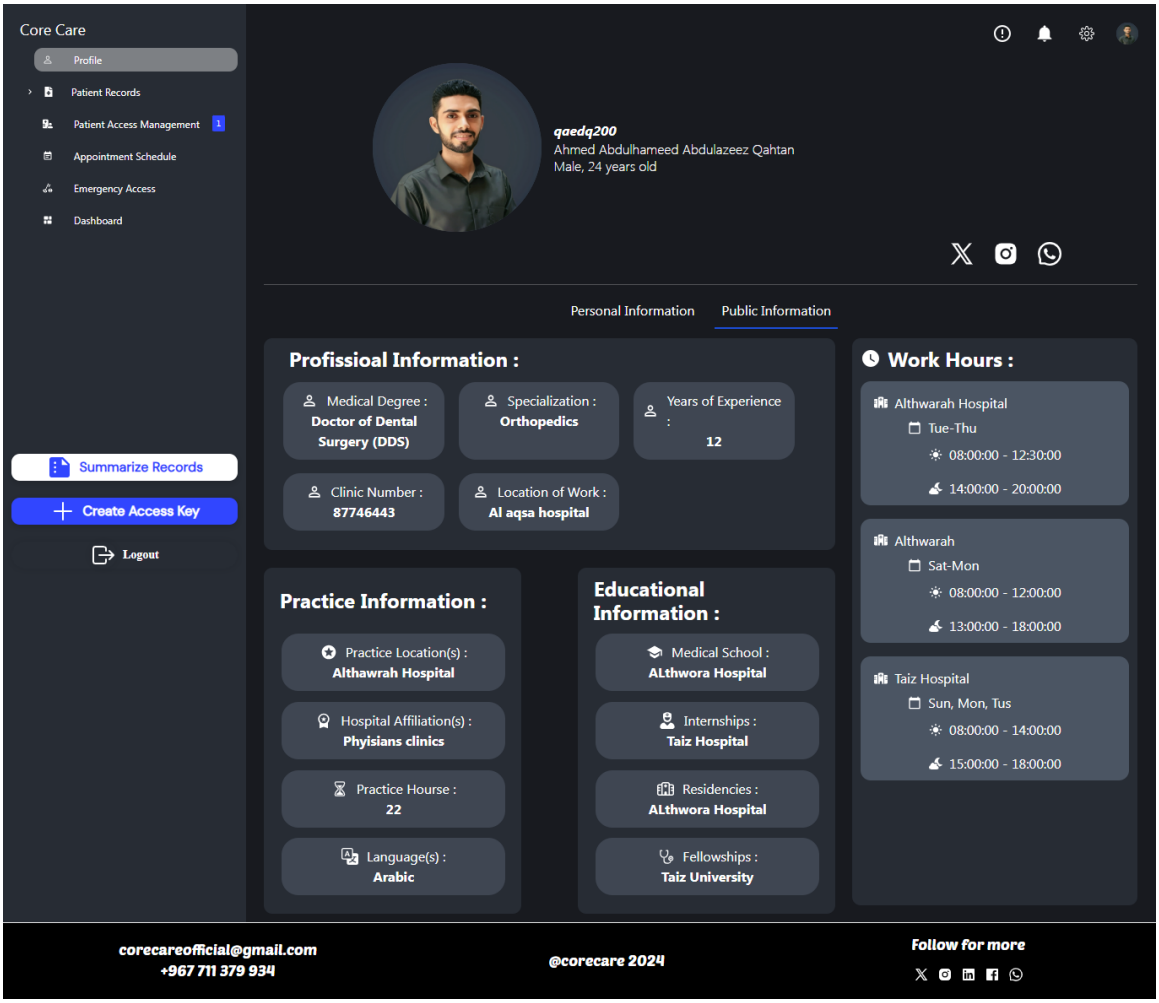


Figure 4.12 Screenshot of the user profile Interface.

The doctor's professional information, working hours, and years of experience are displayed on this page.

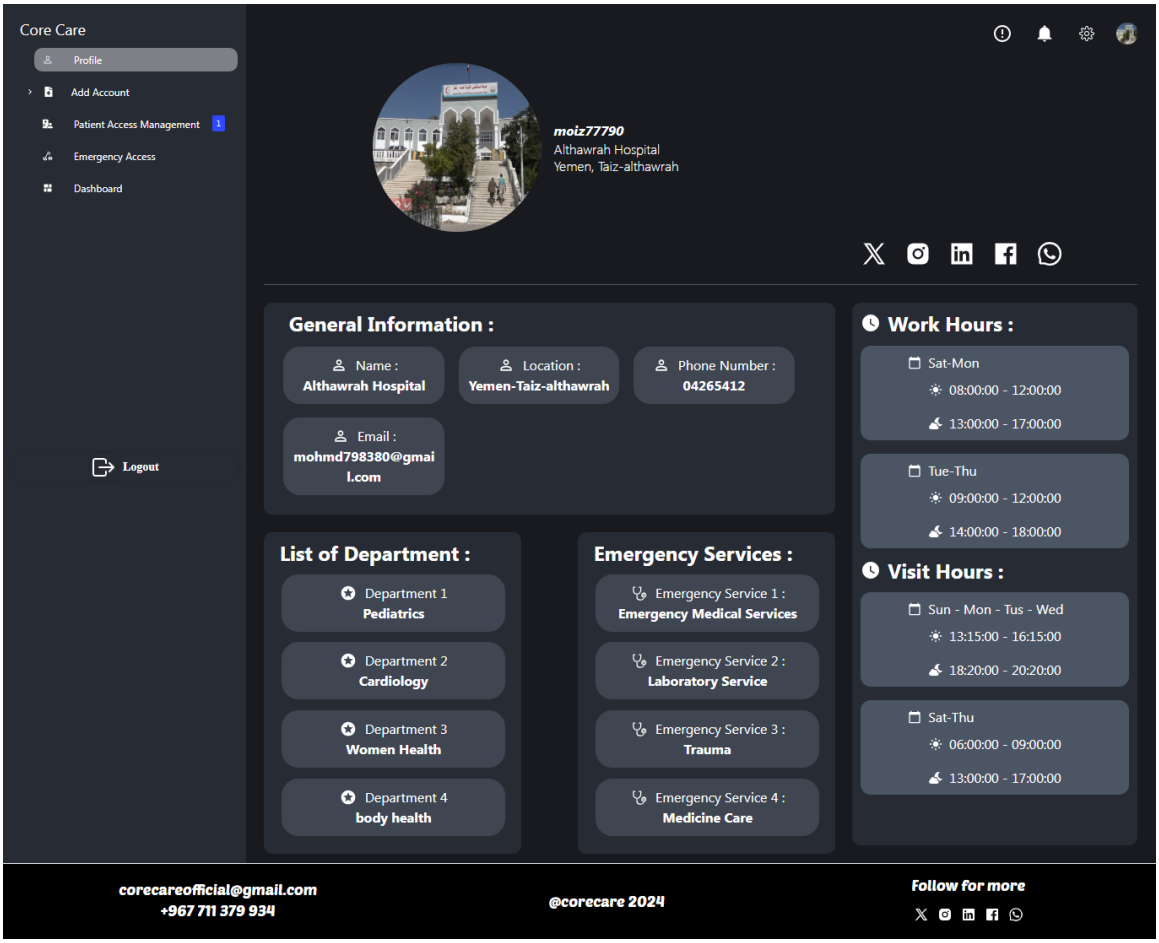


Figure 4.13 Screenshot of the user profile Interface.

General information about the hospital and its working and visiting hours are displayed

4.3.4 Settings page

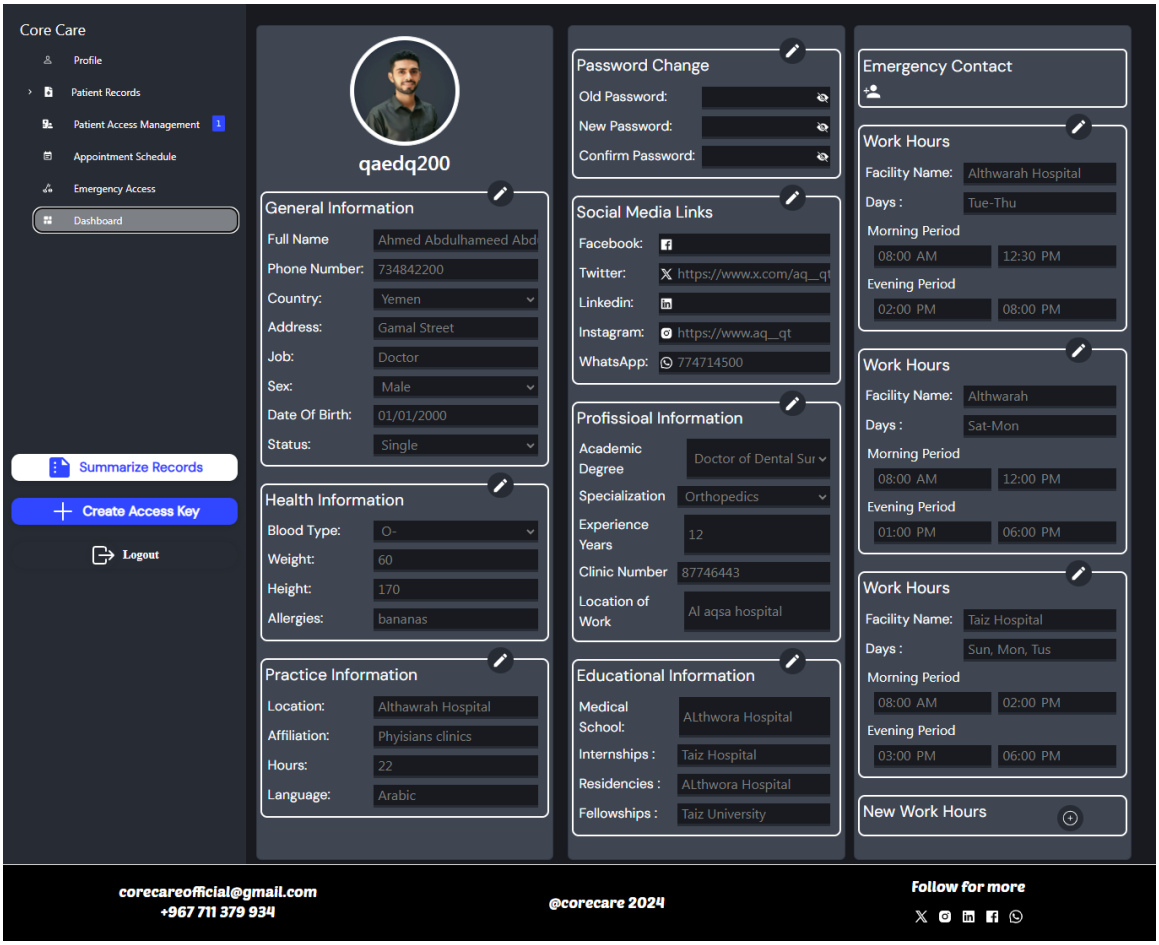


Figure 4.14 Screenshot of the doctor settings Interface.

Doctor information can be edited from settings page

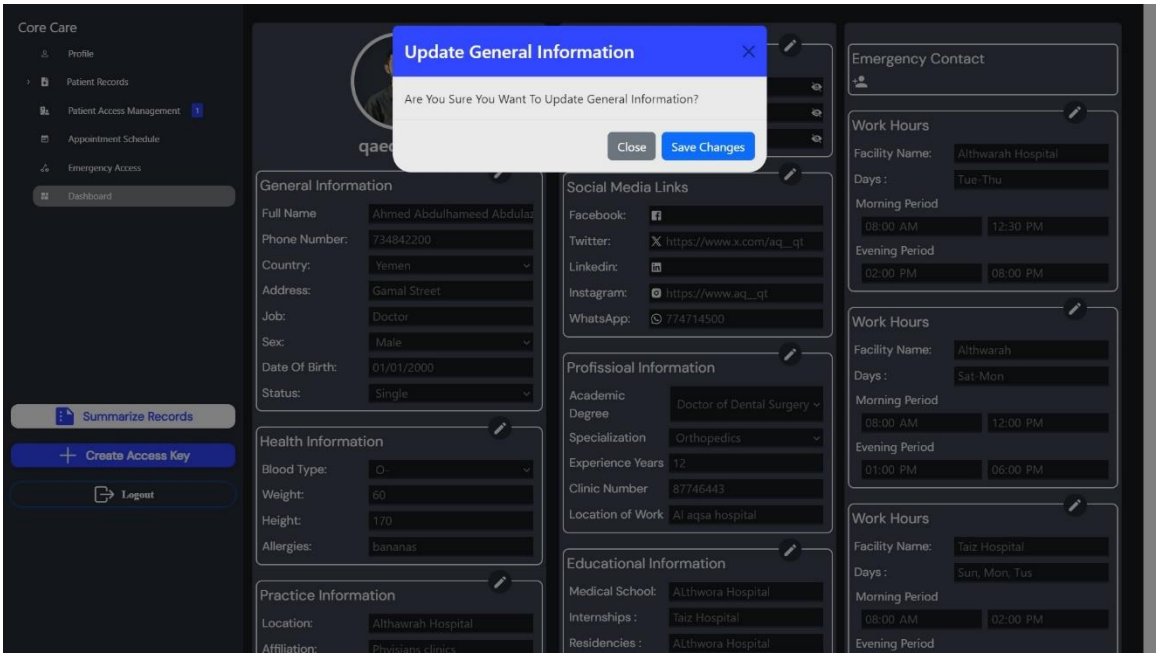


Figure 4.15 Screenshot of the doctor settings Interface.

After editing user information user should confirm this dialog

4.3.5 Patient Appointments

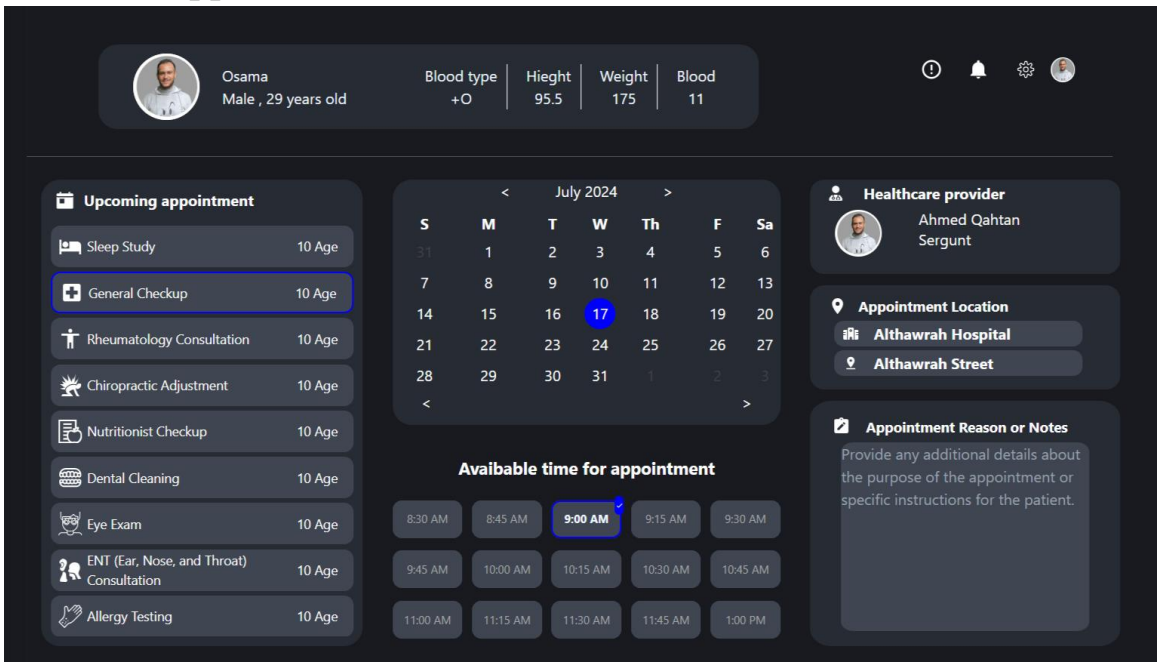


Figure 4.16 Screenshot of the patient Appointments Interface.

Patient Appointments can be Viewed from appointment page

4.3.6 Patient's records page

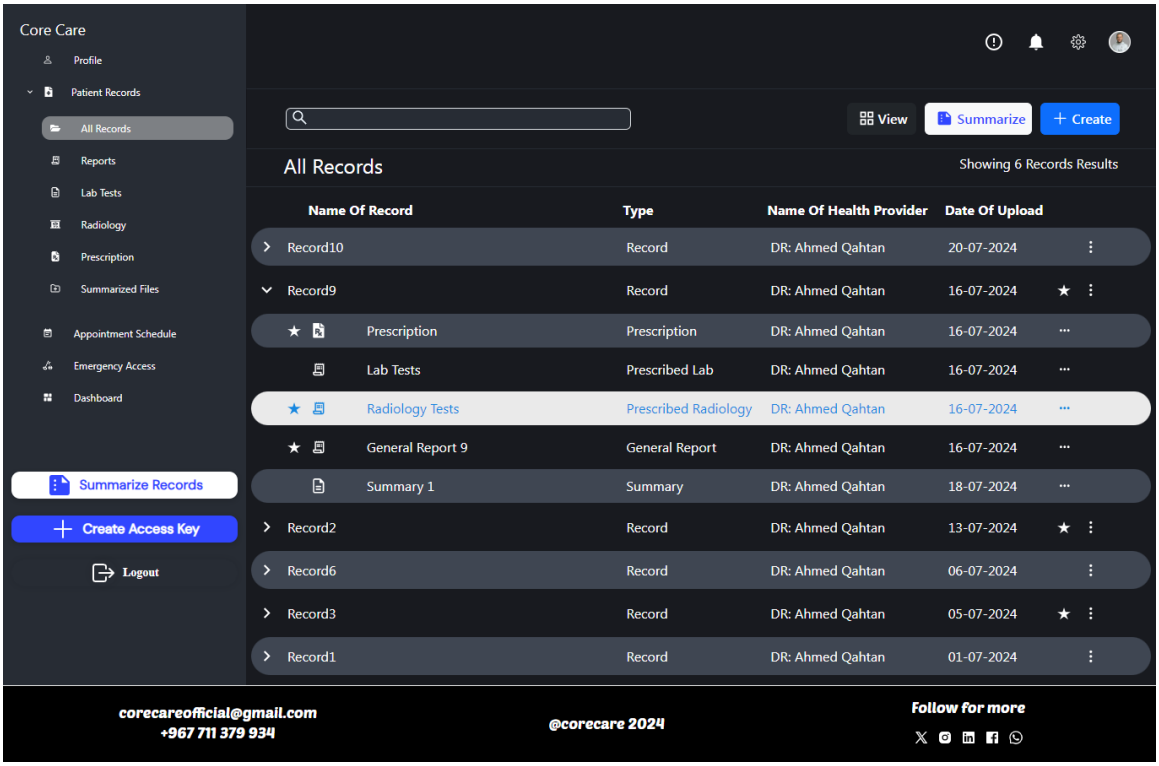


Figure 4.17 Screenshot of the patient records Interface.

Patient records can be viewed from records page

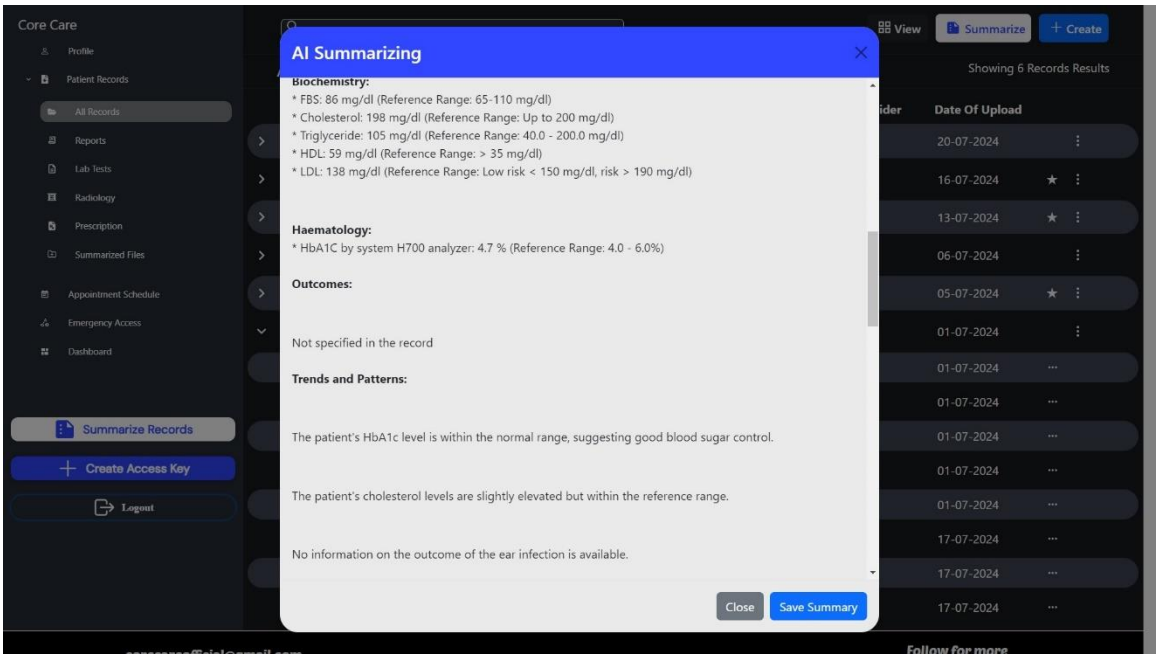


Figure 4.18 Screenshot of the patient records Interface.

User Can Summarized Records Using AI

4.3.7 Patient Access Management

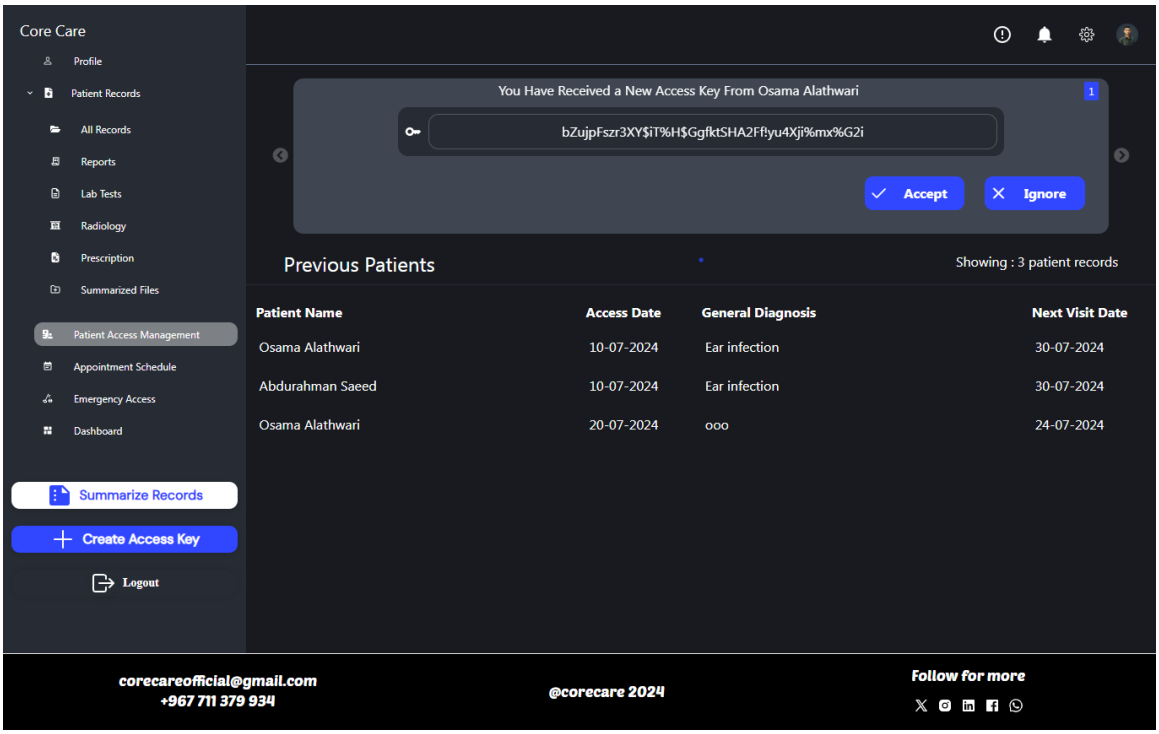


Figure 4.19 Screenshot of the patient Access Management Interface.

Doctor receive access keys from patients in this page and can view, accept or ignore them

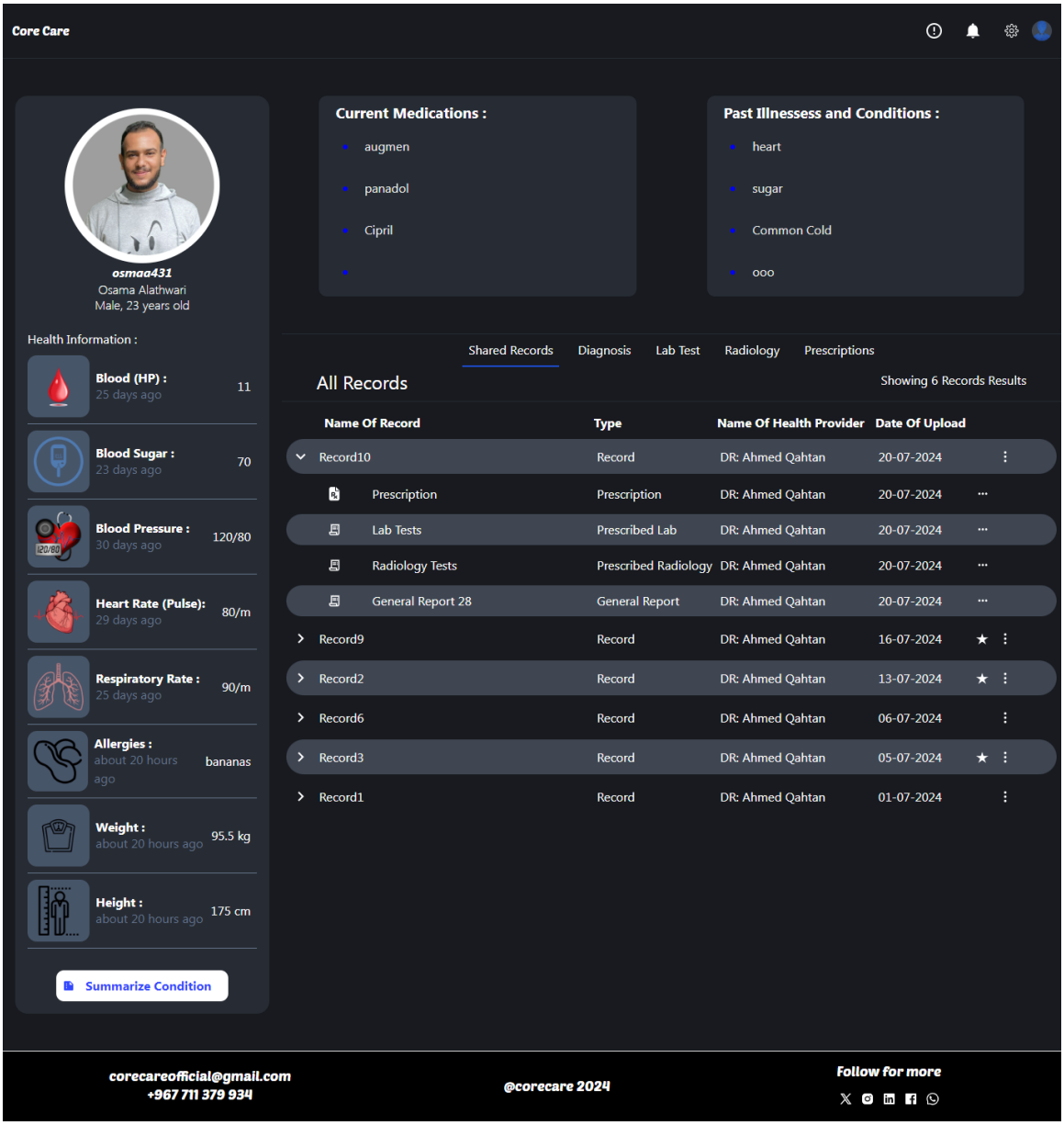


Figure 4.20 Screenshot of the shared patient records Interface.

Shared Patient records can be viewed from this page

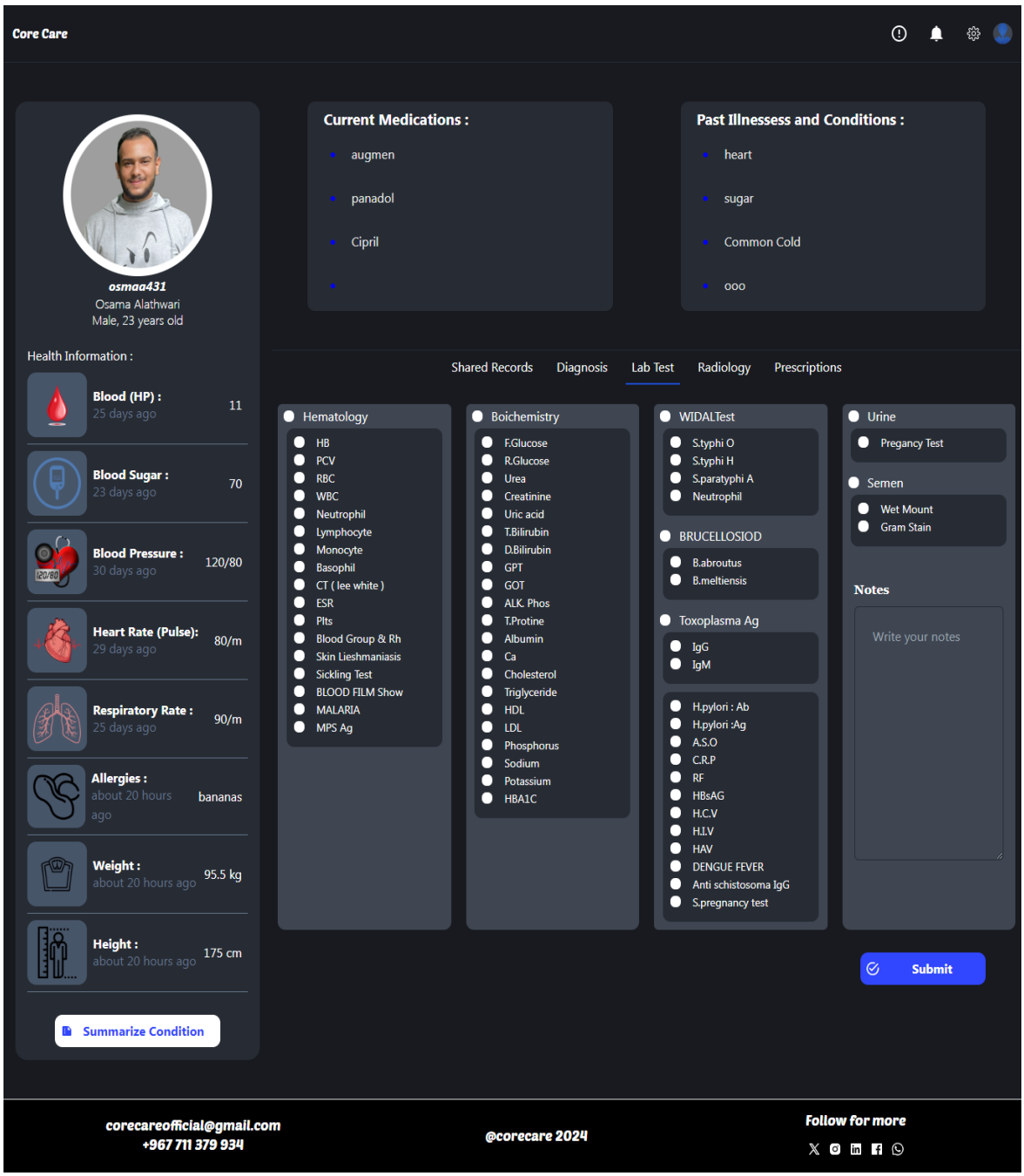


Figure 4.21 Screenshot of the lab tests Interface.

Doctor can prescribe lab test from this page

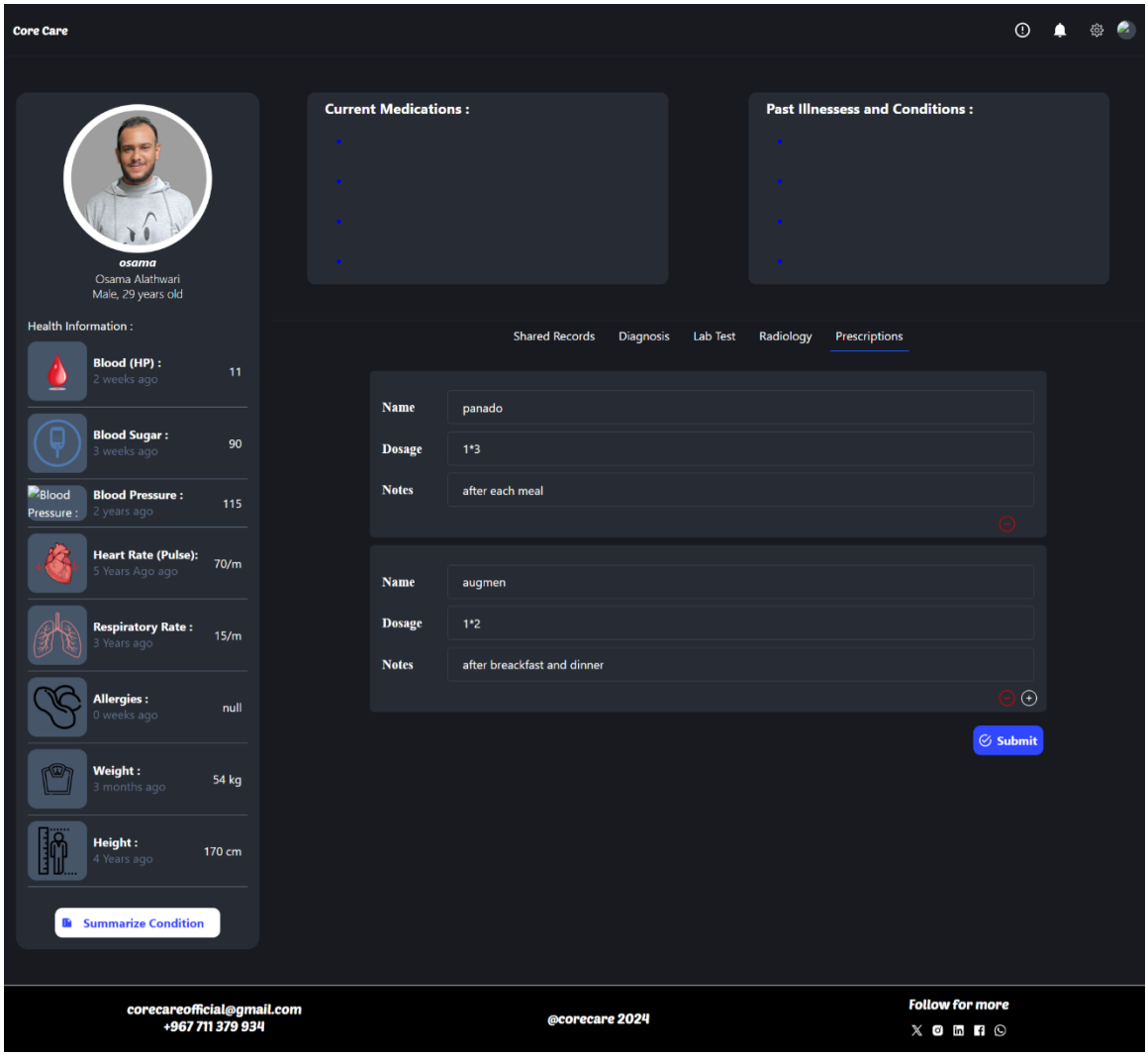


Figure 4.22 Screenshot of the prescription Interface.

Doctor can prescribe medicine from this page

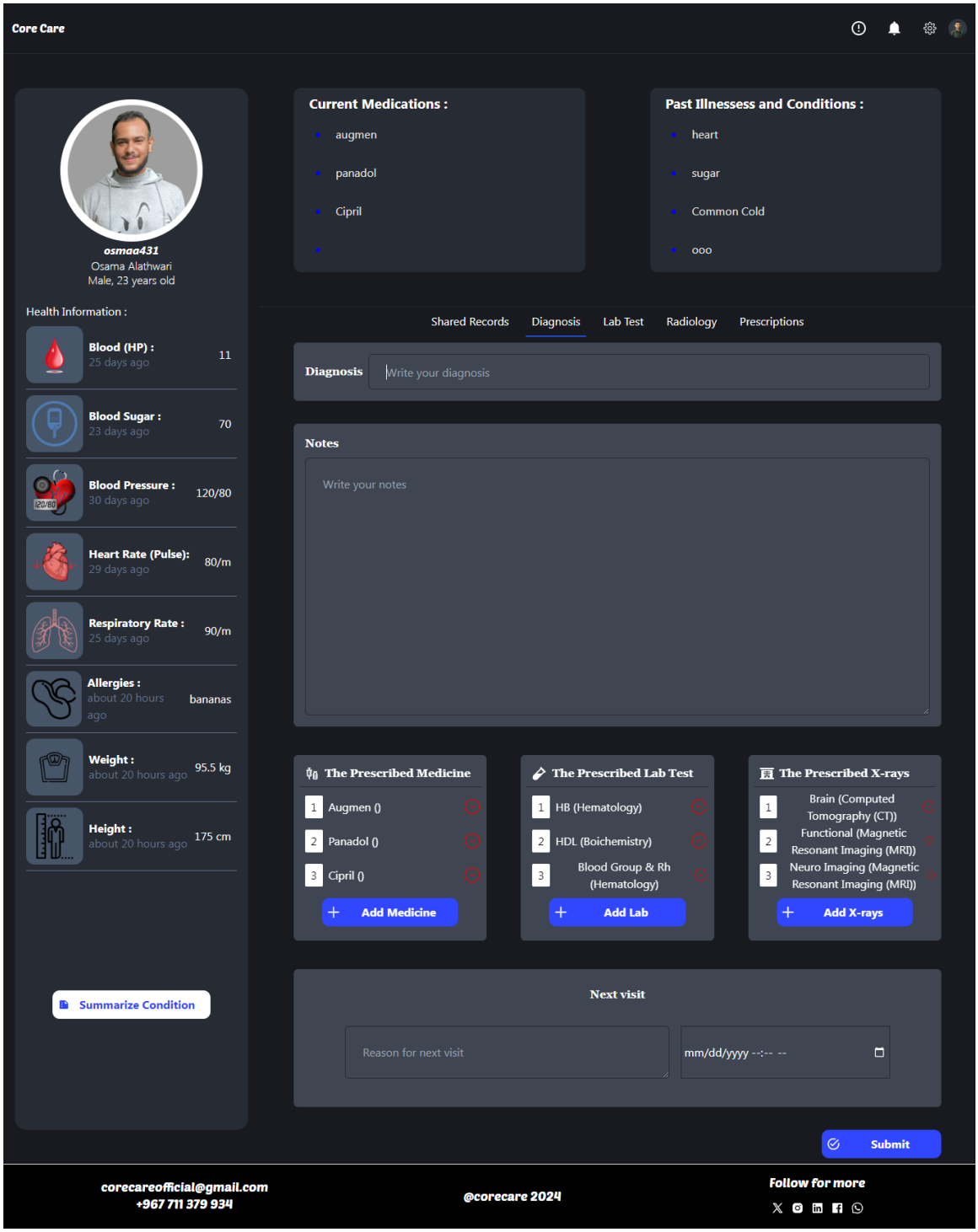


Figure 4.23 Screenshot of the diagnosis Interface.

Doctor make the diagnosis and upload it to patient record from this page

1.3 API Documentation

Base URL: The base URL for all API endpoints is: `http://localhost:5000` (assuming the server is running on port 5000)

Endpoints:

4.3.1 Authentication APIs

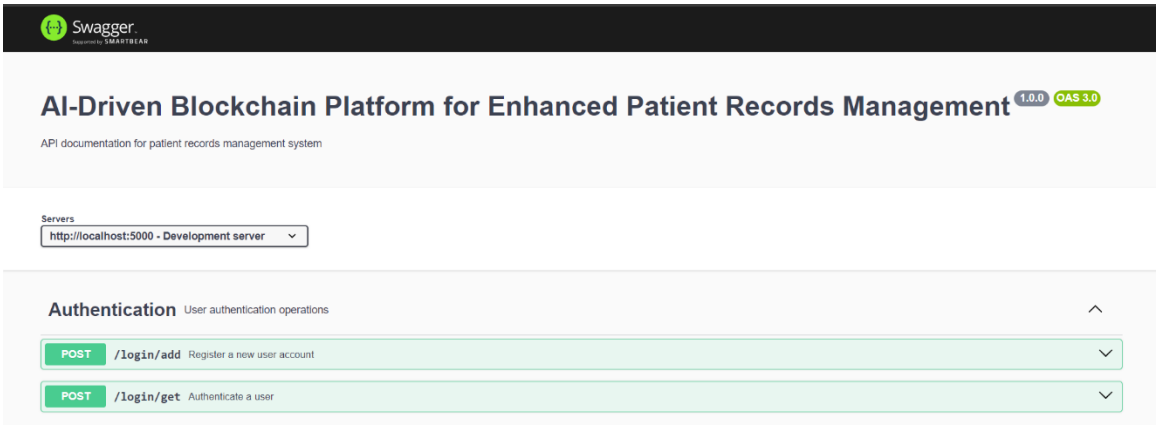


Figure 4.24 Screenshot of the Authentication API

Screenshot of the Authentication endpoint and the operations related to it

4.3.2 Patients APIs

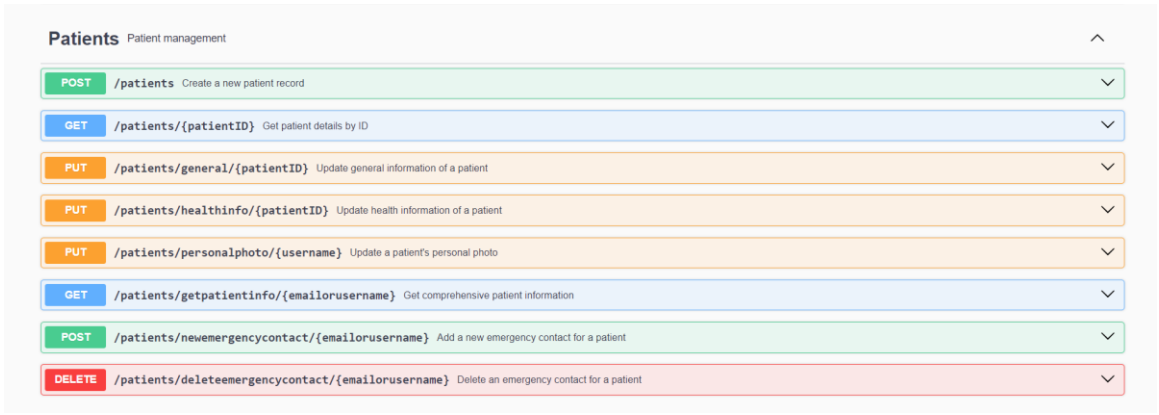


Figure 4.25 Screenshot of the Patients API

Screenshot of the Patients endpoint and the operations related to it

4.3.3 Doctors APIs

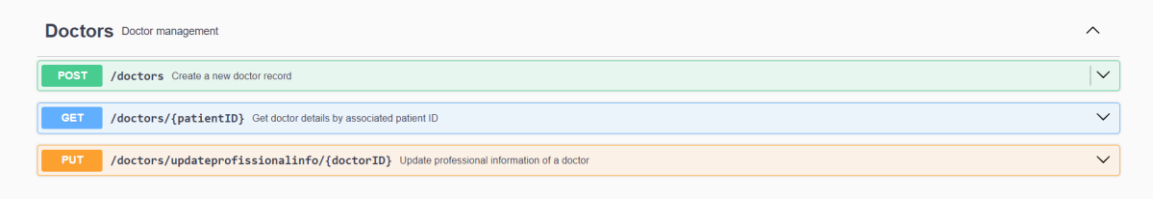


Figure 4.26 Screenshot of the Doctors API

Screenshot of the Doctors endpoint and the operations related to it

4.3.4 Healthcare Providers APIs

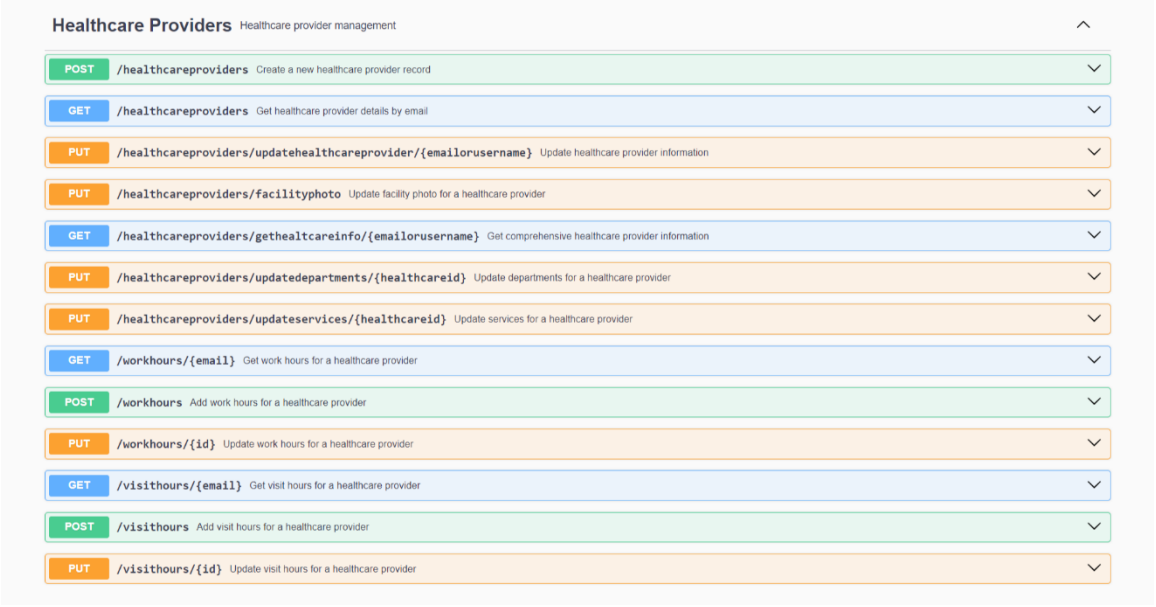


Figure 4.27 Screenshot of the healthcare providers API

Screenshot of the Healthcare Providers endpoint and the operations related to it

4.3.5 Records, Prescriptions and Verification APIs

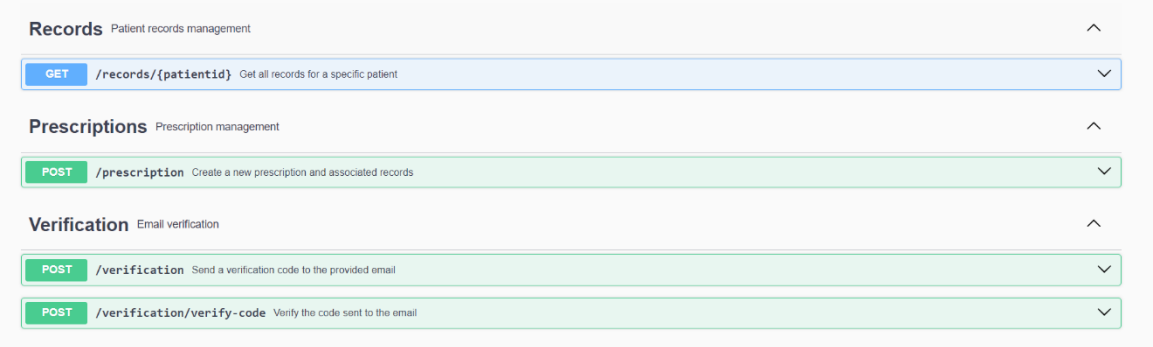


Figure 4.28 Screenshot of the Records, Prescriptions and Verification API

Screenshot of the Records endpoint and the operations related to it

4.4 Entity Relationship Diagram (ERD)

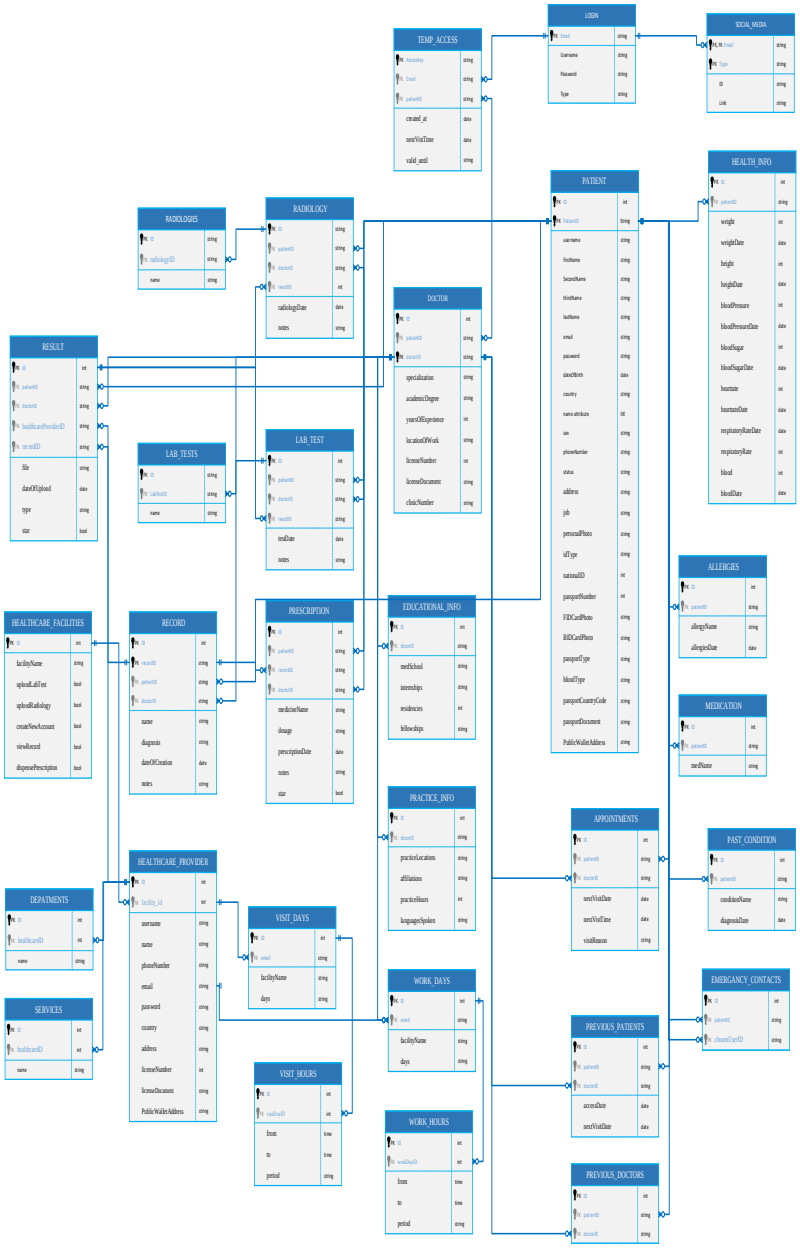


Figure 4.29 ERD Diagram

Screenshot of ERD that illustrates the tables in the database and relationship between them

4.5 Database Schema

SOCIALMEDIA (id, email, type, link)

PATIENT (id, patientid, firstname, secondname, thirdname, lastname, email, password, dateofbirth, country, sex, phonenumber, status, address, job, personalphoto, idtype, fidcardphoto, bidcardphoto, passporttype, passportcountrycode, passportdocument, publicwalletaddress, username, nationalid, passportno, bloodtype)

HEALTH_INFO (id, patientid, weight, height, bloodsugar, heartrate, respiratoryrate, weightdate, heightdate, bloodpressuredate, bloodsugardate, heartratedate, respiratoryratedate, blood, blooddate, bloodpressure)

ALLERGIES (id, patientid, allergyname, allergiesdate)

MEDICATION (id, patientid, medname)

PAST_CONDITION (id, patientid, conditionname, diagnosisdate)

EMERGANCY_CONTACTS (id, patientid, chosenuserid)

DOCTOR (id, doctorid, patientid, specialization, academicdegree, yearsofexperience, locationofwork, licensenumber, licensedocument, clinicnumber)

PRACTICE_INFO (id, doctorid, practicelocation, affiliations, practicehours, languagesspoken)

EDUCATIONAL_INFO (id, doctorid, medschool, internships, residencies, fellowships)

WORK_DAYS (id, email, facilityname, days)

WORK_HOURS (id, workdayid, "from", "to", period)

APPOINTMENTS (id, patientid, doctorid, nextvisitdate, nextvisittime, visitreason)

PREVIOUS_DOCTORS (id, patientid, doctorid)

PREVIOUS_PATIENTS (id, doctorid, patientid, accessdate, nextvisitdate)

HEALTHCARE_FACILITIES (id, facilityname, uploadlabtest, uploadradiology, createnewaccount, viewrecord, dispenseprescription)

HEALTHCARE_PROVIDER (id, username, name, phonenumber, email, country, address, licensenumber, licensedocument, publicwalletaddress, facility_id, facilityphoto)

RESULT (id, doctorid, patientid, healthcareproviderid, file, dateofupload, type, recordid, star)

LAB_TEST (id, doctorid, patientid, testdate, notes)

LAB_TESTS (id, labtestid, name)

PRESCRIPTION (id, doctorid, patientid, medicinename, dosage, prescriptiondate, notes, recordid, star)

RADIOLOGY (id, doctorid, patientid, radiologydate, notes)

RADIOLOGIES (id, radiologyid, name)

RECORD (id, recordid, diagnosis, dateofcreation, notes, patientid, doctorid, name)

DEPATMENTS (id, healthcareid, name)

LOGIN (email, password, type, username)

SERVICES (id, healthcareid, name)

TEMP_ACCESS (accesskey, patientid, email, created_at, valid_until)

VISIT_DAYS (id, email, facilityname, days)

VISIT_HOURS (id, visitdayid, "from", "to", period)

4.6 Reports

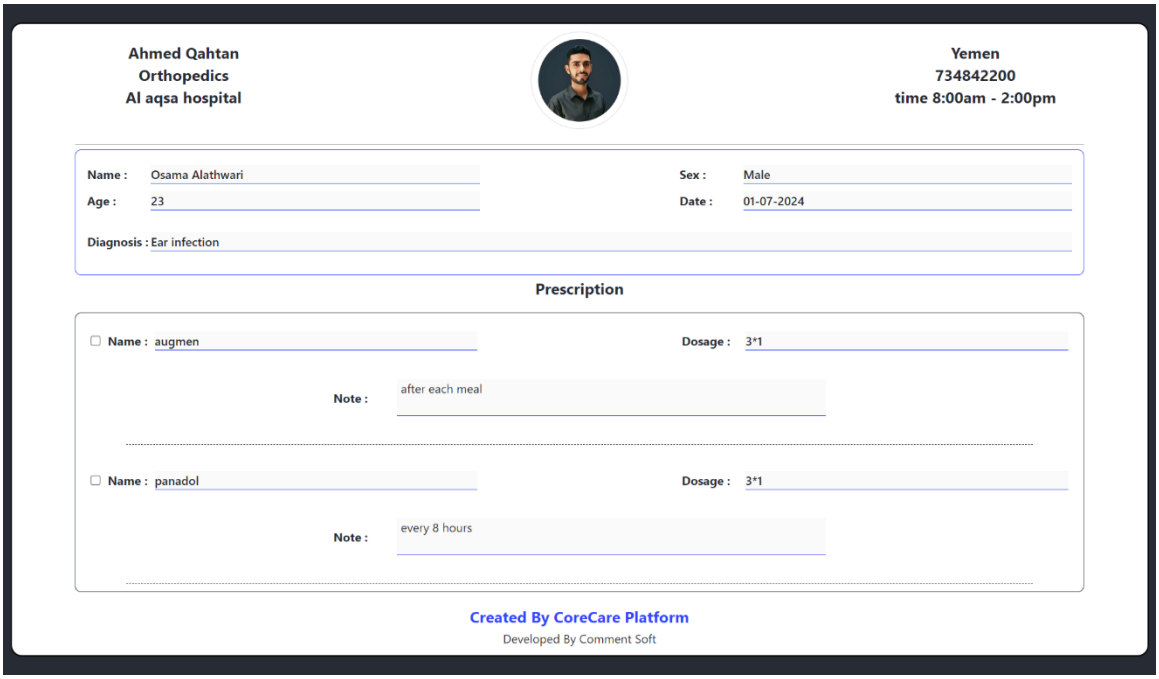


Figure 4.30 Screenshot of the prescription report Interface.

Patient and pharmacy can view this report and pharmacies can dispense them

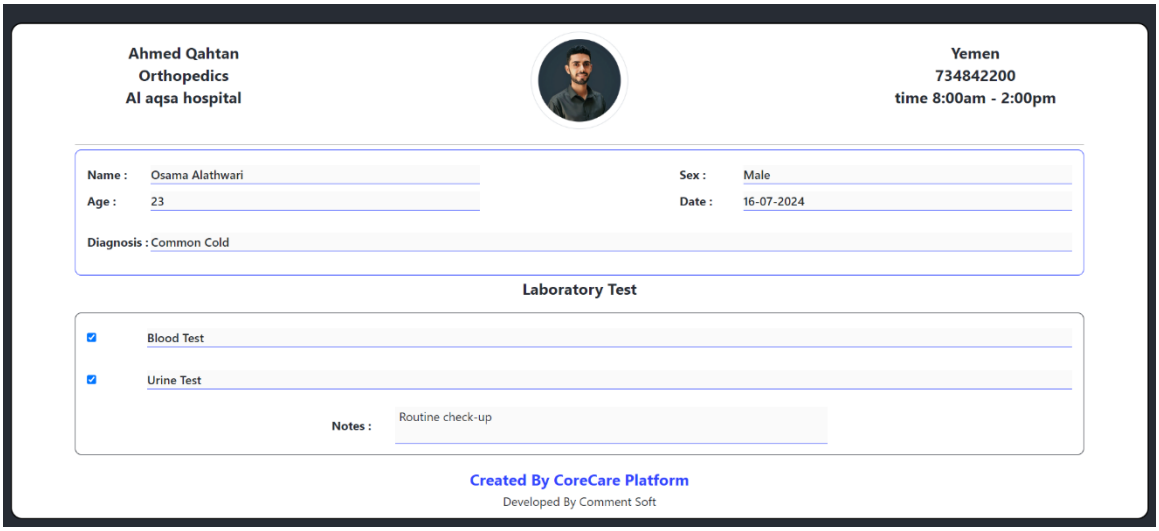


Figure 4.31 Screenshot of the Laboratory tests report Interface.

Patient and Laboratories can view this report

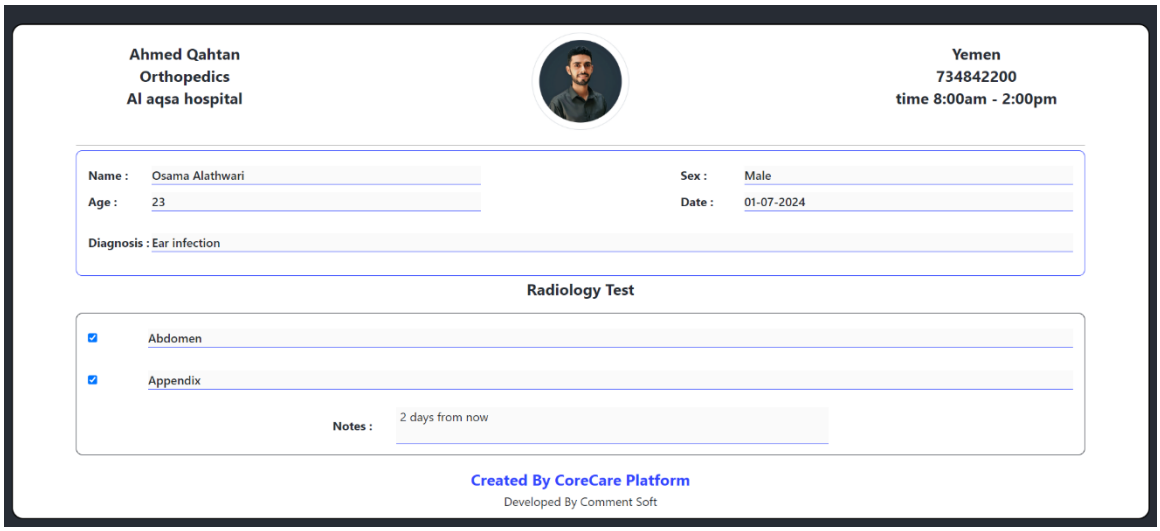


Figure 4.32 Screenshot of the Radiology tests report Interface.

Patient and Radiology center can view this report

Figure 4.33 Screenshot of the General report Interface.

General Report Display the Doctors General Diagnoses

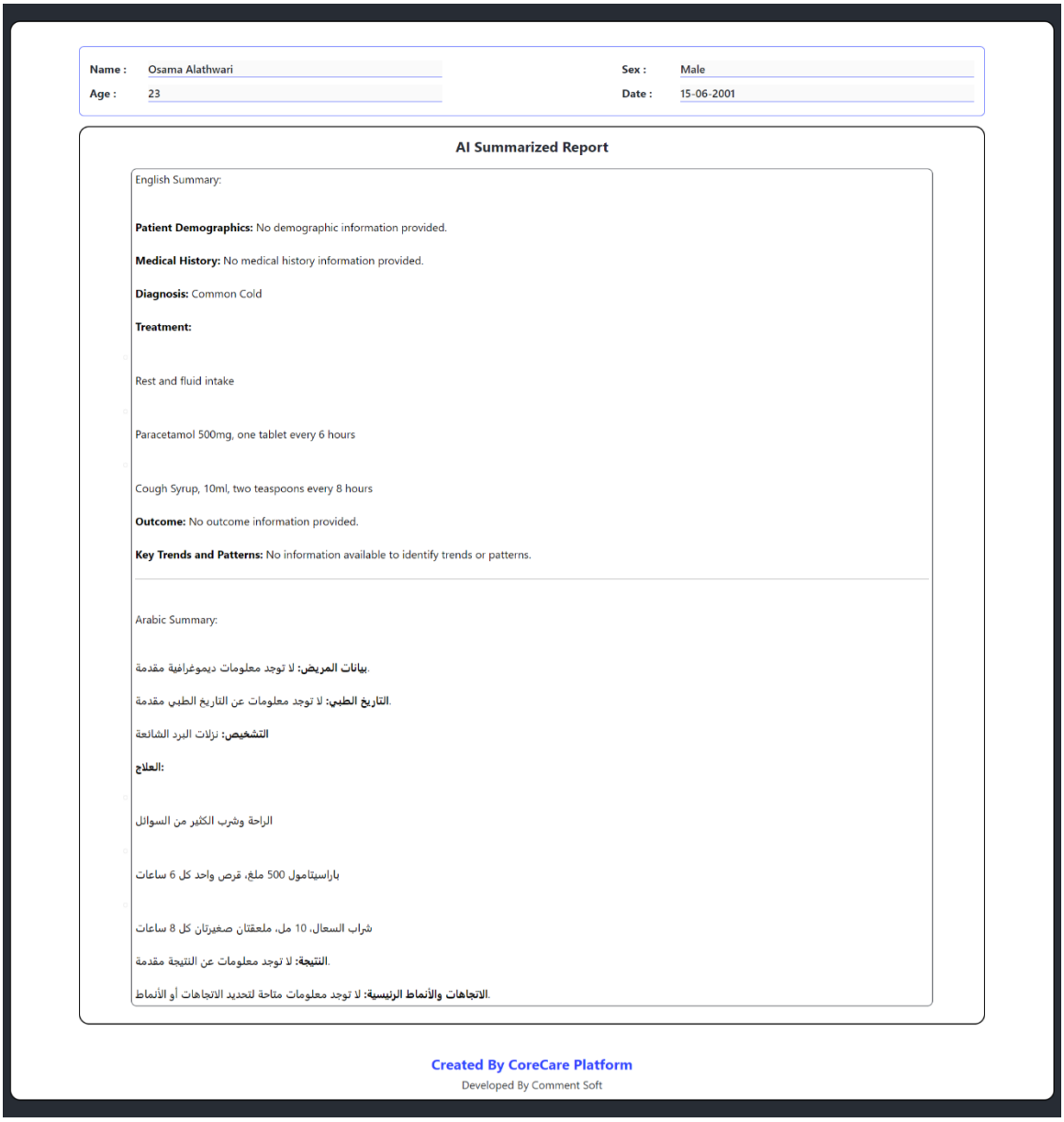


Figure 4.34 Screenshot of the General report Interface.

Display Previous Saved Summarize

Chapter 5: Implementation and Testing

Chapter 5: Implementation and Testing

5.1 Introduction

This chapter details the implementation phase of the project, where the specifications and requirements defined in the analysis chapter are translated into an executable program. This involves constructing the high-level structure of the source code, providing descriptions of key code fragments, illustrating the flow of data throughout the system, and outlining the actions that occur on each page as discussed in the previous chapter.

5.2 Front-end Implementation

5.2.1 Project Folder Structure (Front-end)

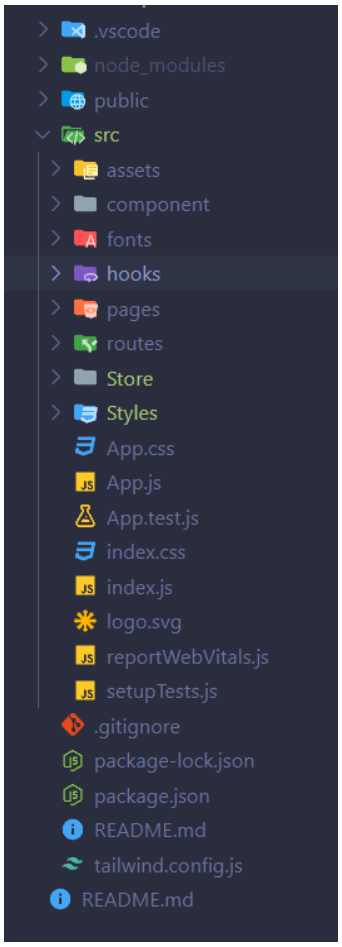


Figure 5.1 Screenshot of the Front-end Folder Structure

5.2.2 Recoil State Management

The following part of code shows the Recoil store that used to store the data of the web application. In Recoil we can store the data make it readable and maintainable.

```
import { atom } from 'recoil';

export const userInfo = atom({
    key: 'userInfo',
    default:
    {
        typeUser: 'Patient',
        firstName: '',
        secondName: '',
        thirdName: '',
        lastName: '',
        email: '',
        dateOfBirth: '',
        sex: '',
        country: '',
        phoneNumber: '',
        job: '',
        address: '',
        status: '',
        photo: null,
        idType: 'National',
        id: '',
        bloodType: '',
        FIdCardPhoto: null,
        BIdCardPhoto: null,
        passportType: '',
        passportCountryCode: '',
        passportPhoto: null,
        userName: '',
        password: '',
        medicalSpecialization: '',
        academicDegree: '',
        licenseNumber: '',
        licenseDocument: null,
        locationOfWork: '',
        PublicWalletAddress: '',
        isForgetton: false
    }
});

export const GeneralData = atom({
    key: 'GeneralData',
```

```
    default: {
        password: '',
        confirmedPassword: '',
        steps: 0,
        isForgetton: false,
    }
});

export const loginInfo = atom({
    key: 'loginInfo',
    default: {
        login: '',
        userName: '',
        email: '',
        password: '',
        patientId: '',
    }
})

export const HealthcareFacilityInfo = atom({
    key: 'HealthcareFacilityInfo',
    default: {
        facilityType: '',
        userName: '',
        name: '',
        country: '',
        address: '',
        phoneNumber: '',
        email: '',
        licenseNumber: '',
        licenseDocument: null,
        password: '',
        type: 'Government',
        facilityPhoto: null,
        PublicWalletAddress: ''
    }
})

export const userHealthInfo = atom({
    key: 'userHealthInfo',
    default:
    {
        radiology: {
            selectedList: [],
            notes: ''
        },
```

```
        labTests: {
            selectedList: [],
            notes: ''
        },
        prescription: [],
        diagnosis: '',
        notes: '',
        dateOfNextVisit: '',
        reasonOfNextVisit: '',
    }

});
```

5.2.3 Back-end Data Connectivity (Axios)

The simple data fetching module for REST API connectivity using all the required methods like GET, POST, PUT, DELETE. we use to fetch data from API to the React JS side (store) and post data from react to the API.

5.2.3.1 Fetch API

The following code used to fetch user info for profile page

```
    const getUserData = async (fetchText, param) => {
        console.log('param;', param);
        console.log('fetchText:;', fetchText);
        try {
            const response = await fetch(fetchText, {
                method: "GET",
                headers: {
                    "Content-Type": "application/json",
                },
                params: JSON.stringify(param),
            });
            const jsonData = await response.json();
            setUserInfo(jsonData);
        } catch (err) {
            setError(err.message);
        }
    };
```

5.2.3.2 Post API

```
try {
            const response = await fetch(`http://192.168.137.1:5000/workhours`, {
                method: "POST",
```

```
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify(data)
            });
            console.log("res = " + response);
            const jsonData = await response.json();
            console.log('message from server: ' + jsonData.message);
            if (jsonData.message === "Work Day and Work Hours added
successfully") {
                toast.current.show({ severity: 'success', summary: 'Success',
detail: 'Work Day and Work Hours added successfully' });
            } else {
                toast.current.show({ severity: 'error', summary: 'Error', detail:
jsonData.message });
            }
        } catch (error) {
            console.error(error.message);
            toast.current.show({ severity: 'error', summary: 'Error', detail:
'Error Work Day and Work Hours added' });
        }
```

5.2.3.3 PUT API

```
try {
            const response = await
fetch(`http://192.168.137.1:5000/visithours/${id}`, {
                method: "PUT",
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify(data)
            });
            console.log("res = " + response);
            const jsonData = await response.json();
            console.log('message from server: ' + jsonData.message);
            if (jsonData.message === "Visit Day and Visit Hours updated
successfully") {
                toast.current.show({ severity: 'success', summary: 'Success',
detail: 'Visit Day and Visit Hours updated successfully' });
            } else {
                toast.current.show({ severity: 'error', summary: 'Error', detail:
jsonData.message });
            }
        } catch (error) {
            console.error(error.message);
```

```
            toast.current.show({ severity: 'error', summary: 'Error', detail:
'Error Visit Day and Visit Hours updated' });
        }
```

5.2.3.4 Delete API

```
 try {
            const response = await
fetch(`http://192.168.137.1:5000/emergencycontacts/${id}`, {
                method: "DELETE",
                headers: {
                    'Content-Type': 'application/json'
                }
            });
            console.log("res = " + response);
            const jsonData = await response.json();
            console.log('message from server: ' + jsonData.message);
            if (jsonData.message === "Emergency Contact Deleted Successfully") {
                toast.current.show({ severity: 'success', summary: 'Success',
detail: 'Emergency Contact Deleted Successfully' });
            } else {
                toast.current.show({ severity: 'error', summary: 'Error', detail:
jsonData.message });
            }
        } catch (error) {
            console.error(error.message);
            toast.current.show({ severity: 'error', summary: 'Error', detail:
'Error Emergency Contact Deleted' });
        }
    };
```

5.3 Back-end Implementation

The backend side (Nodejs) deals with validate, and store the data in the database. In this section we will introduce some fractions of the codes that do this work.

5.3.1 Routes

Contain all URL paths or routes. Each route is connected to a specific controller which deals with data of a specific table (select, delete, insert, …). The Controller component acts as a mediator between the frontend and Model components. It receives a request from the client, it coordinates with the Model component to query for data (or update data), then return the data as JSON to the

frontend. In other word all operations of store, query, add, update, delete are done in controller, in simple word the requests are handled in controller

```
app.use('/login', loginRoutes);
app.use('/patients', patientRoutes);
app.use('/doctors', doctorRoutes);
app.use('/healthcareproviders', healthcareProviderRoutes);
app.use('/healthinfo', healthInfoRoutes);
app.use('/socialmedia', socialMediaRoutes);
app.use('/allergies', allergiesRoutes);
app.use('/medications', medicationRouts);
app.use('/pastconditions', pastConditionRoutes);
app.use('/previousdoctors', previousDoctorsRoutes);
app.use('/practiceinfo', practiceInfoRoutes);
app.use('/educationalinfo', educationalInnfoRoutes);
app.use('/workhours', workHoursRoutes);
app.use('/visithours', visitHoursRoutes);
app.use('/departments', departmentsRoutes);
app.use('/services', servicesRoutes);
app.use('/emergencycontacts', emergencyContacts);
app.use('/records', records);
app.use('/prescription', prescriptionRoutes);
app.use('/results', resultRoutes);
app.use('/verification', verification);
```

5.3.2 Login Route

```
router.post('/get', async (req, res) => {
    const { email, password } = req.body;
    console.log(email);
    console.log(password);
    try {
        const checkLogin = await pool.query('SELECT type, password, username,
email FROM LOGIN WHERE email = $1 or username = $1', [email]);
        if (checkLogin.rowCount === 0) {
            return res.status(404).send('User not found');
        }

        // Access the first row of the result
        const { type, password: hashedPassword, username, emaill } =
checkLogin.rows[0];

        const isMatch = await bcrypt.compare(password, hashedPassword);

        if (!isMatch) {
            return res.status(400).send("Invalid username or password");
```

```
        }

        res.status(200).send({ message: "Login successfully", userType: type,
username: username, email: emaill });

    } catch (err) {
        res.status(500).send(err.message);
    }
});
```

5.3.3 Code Verification Routes

```
router.post('/verify-code', (req, res) => {
    const { email, code } = req.body;
    if (verificationCodes[email] && verificationCodes[email] === code) {
        // Code is correct, remove it from the store
        delete verificationCodes[email];
        res.status(200).send('Code verified successfully');
    } else {
        res.status(400).send('Invalid code');
    }
});
```

5.3.4 Diagnosis Route

```
router.post('/', async (req, res) => {
    const { patientid, doctorid, diagnosis, notes, prescribedMedicine,
prescribedLabTests, prescribedXrays, labTestsNotes, radiologyNotes,
nextVisitDate, nextVisitReason } = req.body;
    let status = false;
    const REC = "REC-";
    try {

        const idResult = await pool.query('SELECT MAX(ID) as maxID FROM record');
        const lastID = idResult.rows[0].maxid || 0;
        const newID = lastID + 1;
        const recordID = REC + newID;
        const recordName = 'Record' + newID;

        const createRecordQuery = await pool.query('INSERT INTO record (recordid,
doctorid, patientid, diagnosis, notes, dateofcreation, name) VALUES ($1, $2, $3,
$4, $5, $6, $7) RETURNING *', [recordID, doctorid, patientid, diagnosis, notes,
new Date(), recordName]);
        for (let i = 0; i < prescribedMedicine.length; i++) {
            const { medicineName, dosage, notes } = prescribedMedicine[i];
```

```
            const prescriptionQuery = await pool.query('INSERT INTO prescription
(doctorid, patientid, medicinename, dosage, notes, prescriptiondate, recordid)
VALUES ($1, $2, $3, $4, $5, $6, $7) RETURNING *', [doctorid, patientid,
medicineName, dosage, notes, new Date(), recordID]);
            const prescription = prescriptionQuery.rows[0];
            console.log(prescription);
            status = true;
        }

        if (prescribedLabTests) {
            const labTestsQuery = await pool.query('INSERT INTO labtests
(doctorid, patientid, testdate, notes) VALUES ($1, $2, $3, $4, $5, $6) RETURNING
id', [doctorid, patientid, new Date(), labTestsNotes]);
            const labTests = labTestsQuery.rows[0].id;

            for (let i = 0; i < prescribedLabTests.length; i++) {
                const { name } = prescribedLabTests[i];
                const labTestQuery = await pool.query('INSERT INTO lab_tests
(labtestid, name) VALUES ($1, $2) RETURNING *', [labTests, name]);
                const labTest = labTestQuery.rows[0];
                console.log(labTest);
                status = true;
            }
        }

        if (prescribedXrays) {

            const radiologyQuery = await pool.query('INSERT INTO radiology
(doctorid, patientid, radiologydate, notes) VALUES ($1, $2, $3, $4) RETURNING
id', [doctorid, patientid, new Date(), radiologyNotes]);
            const radiology = radiologyQuery.rows[0].id;

            for (let i = 0; i < prescribedXrays.length; i++) {
                const { name } = prescribedXrays[i];
                const radiologyQuery = await pool.query('INSERT INTO
radiology_tests (radiologyid, name) VALUES ($1, $2) RETURNING *', [radiology,
name]);
                const radiologyTest = radiologyQuery.rows[0];
                console.log(radiologyTest);
                status = true;
            }
        }
        if (nextVisitDate || nextVisitReason) {
            const nextVisitQuery = await pool.query('INSERT INTO appointments
(doctorid, patientid, nextvisitdate, visitreason) VALUES ($1, $2, $3, $4)
RETURNING *', [doctorid, patientid, nextVisitDate, nextVisitReason]);
```

```
        }



        if (!status || createRecordQuery.rows.length === 0) {
            return res.status(400).json({ message: 'Diagnosis not created' });
        }

        res.status(200).json({ message: 'Diagnosis created successfully' });
    } catch (err) {
        res.status(500).json({ message: err.message });
    }
});
```

5.4 Blockchain Implementation

In this project, we employ Ethereum as the blockchain platform, Solidity for smart contract development, and MetaMask for interacting with the blockchain. This architecture leverages the strengths of Ethereum's decentralized network, Solidity's robust smart contract capabilities, and MetaMask's seamless integration for end-user interactions.

5.4.1 Ethereum Blockchain

Ethereum is a decentralized, open-source blockchain platform that enables the creation of smart contracts and decentralized applications (dApps). It uses a consensus algorithm called Proof of Stake (PoS) to validate transactions and secure the network.

Ethereum's main features include:

- **Smart Contracts**: Self-executing contracts with the terms directly written into code.
- **EVM (Ethereum Virtual Machine)**: A Turing-complete virtual machine that executes smart contracts.
- **Gas**: A unit that measures the amount of computational effort required to execute operations.

5.4.2 Solidity

Solidity is a high-level programming language designed for writing smart contracts on the Ethereum blockchain. It is statically typed and supports complex data structures and operations.

Key aspects of Solidity:

- **Contract**: The basic building block of Ethereum applications, containing data (state) and functions (code).
- **Modifiers**: Functions that can change the behavior of other functions.
- **Events**: Mechanisms for logging that allow smart contracts to communicate with the frontend.

5.4.3 MetaMask

- MetaMask is a browser extension and mobile app that functions as a cryptocurrency wallet and a gateway to Ethereum-based dApps. It allows users to manage their Ethereum private keys and interact with smart contracts and dApps directly from the browser.
- MetaMask features:
  - **Account Management**: Creation and management of multiple Ethereum accounts.
  - **Transaction Signing**: Securely signing transactions and interactions with smart contracts.
  - **Integration**: Seamless integration with dApps through web3.js.

5.5 Artificial Intelligence Implementation

The AI components are seamlessly integrated into the overall system architecture, ensuring seamless data flow and collaboration between the AI engine, the blockchain network, and the user-facing interfaces.

- **Data Preprocessing**: Patient data, including medical records, test results, and imaging, is preprocessed and transformed into a format suitable for AI analysis. This may involve data cleaning, normalization, and feature engineering.

- **Document Authentication**: The AI-driven document authentication mechanism leverages computer vision and deep learning techniques to verify the integrity and authenticity of user-related documents. When a document is uploaded, the system extracts the text from the document using Optical Characters Recognition then the extracted text is sent to Gemini via api to analyze it and verify the authenticity of a document

- **API Integration**: The AI-powered functionalities are exposed through a set of APIs, allowing seamless integration with the front-end and backend components of the platform. This ensures that users can access the AI-driven insights and recommendations within the context of their daily workflows.

5.5 Testing and Evaluation

This section includes the methods that were used for testing, and evaluating the system. Testing is the process of operating the system with the intention of finding errors in the system, if any, and also fulfilling all results and requirements. System testing and evaluation was done in two phases as follows:

➢ **The first phase:**

In this phase we test the system where the team compared the actual performance of the system with the expected and by ensuring that the system achieve all the functional and nonfunctional requirements obtained in the analysis phase and also verify that the system performs all the processes in the platform and make sure that it completes its work as Required without mistakes.

➢ **The second phase:**

In the second phase we collect opinions from stackholders like patents, doctors, pharmacies, laboratories, radiology centers and doctors to know their impression and control the way they use the platform by using it on their phones and computers give them a chance to try to evaluate and give their comments.

5.6 Summary

The AI-Driven Blockchain platform for patients' records management system was implemented in three phases: in the first phase we implemented the front-end of our platform using ReactJS and Tailwindcss to make user-friendly and Responsive for all users. In the second phase we implemented the back-end of our platform using PostgreSQL as the database and NodeJS (Express.js) to implement the server and create the APIs to handle the dataflow between the front-end and the back-end, we also integrated with AI via API to Analyze Patient’s history and give insights for both patients and doctors. In the third phase we implemented the blockchain network using Ethereum, Solidity and IPFS to store patients’ records in a more secure and decentralized way to maintain their privacy. In the final phase we used AI to summarize patient history and records and provide valuable insight. The chapter also outlines the testing steps undertaken to evaluate the system's performance and functionality

Chapter 6: Conclusion

Chapter 6: Conclusion

6.1 Introduction

In this chapter, we will discuss the results of our AI-Driven Blockchain platform for patients' records management system. We will evaluate how well the project met its objectives, highlight the key achievements, and reflect on the challenges faced during the development process. Additionally, we will outline the directions for future studies and potential improvements.

6.3 Objectives Achieved

The project successfully achieved its primary objectives, which were outlined in the introductory chapter. These objectives include:

- ✓ **Secure and Decentralized Platform**: The project developed a robust and decentralized platform for storing and managing patient records using blockchain technology. The Ethereum blockchain infrastructure ensures the security, immutability, and transparency of patient data.
- ✓ **Integration of AI Algorithms**: Advanced AI algorithms were successfully integrated into the platform, enabling comprehensive analysis of patient data. The AI components provide personalized insights, treatment recommendations, and decision support to healthcare providers, enhancing the quality of care delivered.
- ✓ **Robust Access Control Mechanism**: The platform implemented a sophisticated access control mechanism using smart contracts and cryptographic techniques. This ensures that patient data is only accessible to authorized parties, maintaining the privacy and confidentiality of sensitive medical information.
- ✓ **Seamless Data Sharing**: The project facilitated seamless and secure sharing of patient records among healthcare stakeholders. The platform's interoperability features enable efficient collaboration and coordination of care, improving the overall healthcare ecosystem.
- ✓ **User-Friendly Interface**: The development of intuitive and user-friendly interfaces for both healthcare professionals and patients was successfully achieved. The platform's design prioritizes ease of use, ensuring that users can navigate and interact with the system effectively.

6.4 Challenges

During the development and implementation of the AI-Driven Blockchain Platform for Secure and efficient Patient Records Management, several challenges were encountered. These challenges include:

1. **Validating personal documents uploaded by the users to verify their identity due to lack of automated system from the Government**
2. **Validating licenses uploaded by healthcare providers to verify their identity due to lack of automated system from the Ministry of Health**
3. **Difficulties in integration with Gemini API due to the structure of their API**
4. **Lack of resources in blockchain and AI integration cause they are new technologies**
5. **Poor internet connectivity**

6.5 Future Work

While the AI-Driven Blockchain Platform for Secure and efficient Patient Records Management has achieved significant milestones, there are several areas for future exploration and improvement. These include:

1. **Adding support for other languages including Arabic**
2. **Enabling patients to add their old medical paper records**
3. **Enhancing User Interface and User Experience**
4. **Expansion of AI Capabilities**
5. **Integration with IoT Devices**

REFERENCES

REFERENCES

[1] (IET Book Series On e-Health Technologies Vol. 20) Sudeep Tanwar, Sudhanshu Tyagi, Neeraj Kumar - Security And Privacy Of Electronic Healthcare Records_ Concepts, Paradigms And Solutions-IET_The Insti

[2] (Healthcare Technologies) Balusamy Balamurugan (editor), Naveen Chilamkurti (editor), T. Lucia Agnes Beena (editor), T. Poongodi (editor) Blockchain and Machine Learning for e-Healthcare Systems-Ins

[3] Imran Bashir - Mastering Blockchain_ A technical reference guide to the inner workings of blockchain, from cryptography to DeFi and NFTs, 4th Edition-Packt Publishing (2023)

[4] Stamatellis, C., Papadopoulos, P., Pitropakis, N., Katsikas, S., & Buchanan, W. J. (2020). A privacy-preserving healthcare framework using hyperledger fabric. *Sensors*, *20*(22), 6587.

[5] Le Nguyen, T. and Do, T. T. H. (2019). "Artificial intelligence in healthcare: A new technology benefit for both patients and doctors," 2019 Portland International Conference on Management of Engineering and Technology (PICMET), Portland, OR, USA, pp. 1–15, doi: 10.23919/PICMET.2019.8893884.

[6] Haddad, A., Habaebi, M.H., Islam, M.R., Hasbullah, N.F. and Zabidi, S.A., 2022. Systematic review on ai-blockchain based e-healthcare records management systems. IEEE Access, 10, pp.94583-94615.

[7] Reyna, A., Martín, C., Chen, J., Soler, E., & Díaz, M. (2018). On blockchain and its integration with IoT. Challenges and opportunities. *Future generation computer systems*, *88*, 173-190.

[8] McGhin, T., Choo, K. K. R., Liu, C. Z., & He, D. (2019). Blockchain in healthcare applications: Research challenges and opportunities. *Journal of network and*

computer applications, 135, 62-75.

[9] Shetty, J., Dash, D., Joish, A. K., & Guruprasad, C. (2020). Review paper on web frameworks, databases and web stacks. *Intern Res J Eng Technol (IRJET)*, *7*(5).

[10] Corinne Bernstein "TechTarget digital health (digital healthcare) " [online].Available : https://www.techtarget.com/searchhealthit/definition/digital-health-digital-healthcare#:~:text=Benefits%20of%20digital%20health,tailor%20medicine%20for%20individual%20patients. [Accessed 17 Feb 2024].

[11] Britannica "e-healthcare"[online]. Available: https://www.britannica.com/science/e-health . [Accessed 17 Feb 2024].

[12] Founda Health "Benefits of e-health"[online]. Available: https://www.foundahealth.com/resources/blog/what-is-ehealth-explanation-benefits-and-4-examples . [Accessed 17 Feb 2024].

[13] RECORD NATIONS "Traditional Patient Records vs Electronic Health Records (EHR)" [online]. Available: https://www.recordnations.com/blog/traditional-paper-records-vs-ehr/.[Accessed 17 Feb 2024].

[14] Laravel documentation [online]. Available: https://laravel.com/docs/10.x/installation#meet-laravel . [Accessed 17 Feb 2024].

[15] Node.js documentation [online]. Available: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs . [Accessed 17 Feb 2024].

[16] Ethereum documentation [online]. Available: https://ethereum.org/developers/docs/intro-to-ethereum#what-is-ethereum . [Accessed 17 Feb 2024].

[17] Ethereum documentation [online]. Available: https://ethereum.org/developers/docs/intro-to-ethereum#what-is-ether [Accessed 17 Feb 2024].

[18] IBM documentation [online]. Available: https://cloud.ibm.com/docs/blockchain?topic=blockchain-ibp-console-overview . [Accessed 17 Feb 2024].

[19] IBM documentation [online]. Available: https://cloud.ibm.com/docs/blockchain?topic=blockchain-develop-vscode . [Accessed 17 Feb 2024].

[20] Hyperledger Fabric documentation [online]. Available: https://hyperledger-fabric.readthedocs.io/en/latest/whatis.html#hyperledger-fabric . [Accessed 17 Feb 2024].

[21] Blockchain council "What is Hydrachain technology" [online]. Available: https://www.blockchain-council.org/blockchain/what-is-hyderachain-technology-how-it-works/ . [Accessed 17 Feb 2024].

[22] BigchainDB [online]. Available: https://docs.bigchaindb.com/en/latest/ . [Accessed 17 Feb 2024].

[23] Openchain documentation [online]. Available: http://docs.openchain.org/en/latest/general/overview.html . [Accessed 17 Feb 2024].

[24] Quorum Whitepaper v0.2

[25] EOS documentation [online]. Available: https://developers.eos.io/welcome/latest/index . [Accessed 17 Feb 2024].

[26] Web.dev documentation [online]. Available: https://web.dev/learn . [Accessed 17 Feb 2024].

[27] JavaScript Mdn Web Docs [online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript . [Accessed 17 Feb 2024].

[28] Y. Chen, H. Li, K. Li and J. Zhang, “An Improved P2P File System Scheme Based On IPFS and Blockchain,” 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, 2017, pp. 2652–2657, doi: 10.1109/ BigData.2017.8258226.

[29] IPFS documentation [online]. Available: https://docs.ipfs.tech/ . [Accessed 17 Feb 2024].

[30] Wallets documentation [online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/developapps/wallet.html# . [Accessed 17 Feb 2024].

[31] "angular documents," [Online]. Available: https://angular.io/docs . [Accessed 17 Feb 2024].

[32] SPEC INDIA, "SPEC INDIA" [Online]. Available: https://www.spec-india.com/blog/front-end-frameworks . [Accessed 17 Feb 2024].

[33] "Vue documents". [Online]. Available: https://vuejs.org/guide/introduction.html . [Accessed 17 Feb 2024].

[34] PostgreSQL documentation [online]. Available: https://www.postgresql.org/about/ . [Accessed 17 Feb 2024].

[35] IPFS documentation [online]. Available: https://docs.ipfs.tech/concepts/ipfs-solves/ . [Accessed 17 Feb 2024].

[36] GitHub documentation [online]. Available: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git . [Accessed 17 Feb 2024].

[37] Python documentation [online]. Available: https://www.python.org/doc/essays/blurb/ . [Accessed 17 Feb 2024].

[38] M. Swan. Blockchain: Blueprint for a New Economy. Beijing: O'Reilly. 2015.

[39] T. Poongodi, R. Sujatha, D. Sumathi, P. Suresh, and B. Balamurugan. 'Blockchain in social networking'. Cryptocurrencies and Blockchain Technology Applications. Wiley Online library; pp. 55–76. 2020.

[40] Gobal Bockchain in Heathcare Market, Focus on Industry Analysis and Opportunity Matrix – Analysis and Forecast, 2018-2025, 2018. Retrieved from https://bisresearch.com/industry-report/global-blockchain-in-healthcaremarket-2025.html.

[41] Blockchain in Insurance: Use Cases and Implementations. Retrieved from https://medium.com/swlh/blockchain-in-insurance-use-cases-and-implementations-a42a00ebcd91.

[42] https://www.insurancejournal.com/news/national/2020/02/21/559057.html.

[43] https://www.ibm.com/blockchain/industries/healthcare.

[44] Smit, J.A.R., Mostert, M., van der Graaf, R., Grobbee, D.E. and van Delden, J.J., 2024. Specific measures for data-intensive health research without consent: a systematic review of soft law instruments and academic literature. *European Journal of Human Genetics*, *32*(1), pp.21-30.