

**Title: *Centralised Sharing of Medical Records Using Cloud Platform***



**By  
*Hareesh Kakarla***

***Course: MSc Computer Network Administration and Management***

***Project Unit: M32616-2023/24-PAYYEAR***

***Supervisor: Vasileios Adamos***

***September 2024***

***Word count: 12094***

Please tick

<input checked="" type="checkbox"/>	I hereby declare that this dissertation is substantially my own work
<input checked="" type="checkbox"/>	I do consent to my dissertation in this attributed format (not anonymous), subject to final approval by the Board of Examiners, being made available electronically in the Library Dissertation Repository and/or Department/School/Subject Group digital repositories. Dissertations will normally be kept for a maximum of ten years
<input checked="" type="checkbox"/>	I understand that if I consent, this dissertation will be accessible only to staff and students for reference only
<input checked="" type="checkbox"/>	This permission may be revoked at any time by e-mailing <a href="mailto:data-protection@port.ac.uk">data-protection@port.ac.uk</a>

Date:14-Sep-2024

# ABSTRACT

A centralised medical records sharing system ensures that the right information is available to the right healthcare provider at the right time thus eliminating errors that could result from incomplete or incorrect information. It also improves the data security and compliance, allows real-time update and improves the administrative functions. This paper aims to propose and assess the effectiveness of a cloud-based medical records sharing system that was built with AWS Lambda, AWS API Gateway and Python. The system is intended to provide the user with a fast and secure way of accessing patient's data through a server-less approach and the use of API Endpoints for data retrieval and integration. Some of the main ones are; UI testing aimed at checking the usability of the user interface of the application, load balancing testing which is used to split API calls between different Lambda functions, API testing to check the accuracy of the data exchanged between the API and the client, and latency testing to check the time taken in receiving a response from the API. From the study, it is evident that to ensure that load balancing is efficient and that errors do not affect the system, some consideration must be made. The implementation shows how each of these elements supports a large, scalable, fault-tolerant, and easy-to-use medical record management system that can operate under different levels of traffic and guarantee reliable access to data despite possible errors.

# ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor Mr. Vasileios Adamos for showing me guidance, support and encouragement right from the beginning of this Engineering project. They were vital to the success of this project, and their knowledge and input was much needed.

I would also like to thank my moderator Ms. Zaynab Lamoyero and the faculty and staff of School of Computing for providing the resources and conducive environment.

I am too grateful to my family and to friends their encouragement and moral support made it possible to complete with flying colours the many chores inherent in this task.

# TABLE OF CONTENTS

1. Introduction .....	10
1.1 Project Context .....	10
1.1.2 Need for centralised system .....	11
1.2 Aims & Objectives .....	11
1.2.1 Aim .....	11
1.2.2 Research question.....	12
1.2.3 Project Objectives.....	12
1.3 Limitations.....	12
1.4 Report overview .....	13
2.Literature Review.....	14
2.1 Outline .....	14
2.2 Importance of Health Level Seven (HL7).....	14
2.3 Review of Cloud-Based Medical Record Systems.....	15
2.4 Standards and Protocols for Medical Data Transfer .....	16
2.5 Impact of HL7 Implementation on Healthcare Data Interoperability .....	17
2.6 Research gap.....	18
2.7 Implications for the Proposed Project.....	18
2.8 Conclusion .....	19
3.Project Management .....	21
3.1 Project Methodology choice .....	21
3.2 Professional Issues .....	21
3.3 Conclusion .....	22
4.Methodology.....	24
4.1 Specification and Discussion of the requirements .....	24
4.2 Analysis and Discussion of the IT Methodology .....	24
4.2.1 Methodological Framework .....	24
4.3 Implementation Strategy .....	25
4.3.1 Time Management .....	25
4.4 Summary .....	26
4.5 Conclusion .....	26

5. Implementation .....	28
5.1 Materials / Tools / Data Collection .....	28
5.1.1 Method of Implementation .....	28
5.1.2 Tools Used.....	28
5.2 Implementation Timeline .....	29
Week 1: Initiating the project and project preparation.....	29
Week 2: Design and Initial Setup. ....	29
Week 3: Focused on the development and integration.....	29
Week 4: Testing & Optimization. ....	30
Week 5: Front-end development and the final adjustments .....	30
5.3 Data Collection.....	31
5.3.1 HL7 – Health Level Seven International – Medical Data Format. ....	31
5.3.2 HL7 Medical Data Format.....	32
5.3.3 AWS Lambda Function Creation.....	34
5.3.3.1 Creation of Function in AWS Lambda .....	34
5.3.3.2 Steps Included in creating.....	34
5.3.3.3 Lambda Handler Function .....	35
5.3.3.4 Patient Data as Output .....	35
5.3.3.5 Output Data Format.....	35
5.4 API Gateway .....	35
5.4.1 API and Integration of Function with API Gateway .....	35
5.4.2 Purpose of Using API Call for Lambda Function .....	36
5.4.3 Steps in API Gateway Configuration .....	36
5.4.3.1 Integrate Lambda Function .....	36
5.4.3.2 Configure Method Request Settings .....	36
5.4.3.3 Deploy Using Stage .....	37
5.4.4 Creation of Two API Endpoints.....	38
5.5 Testing .....	39
5.5.1 Creation of User Interface .....	40
5.5.1.1 Front Page: Fetch Patient .....	40
5.5.1.2 Output page : output.html.....	41
5.5.1.3 Main File : app.py.....	41
5.5.1.4 Load Balancing.....	42

5.5.1.5 Handling API Failure .....	42
5.5.1.6 Testing User Interface .....	43
5.6 Conclusion .....	45
6.Results .....	46
6.1 Introduction .....	46
6.1.1 Testing Load Balancing .....	46
6.1.2 Testing with API Failures .....	47
6.1.3 Testing Latency .....	48
6.1.4 Testing Latency After API Failure (ohio server 2) .....	49
6.1.5 Testing Latency After API Failure (Virginia - server 1) .....	50
6.2 Analysis of Results.....	52
6.3 Discussion of results .....	52
6.4 Evaluation Against Requirements / Research Goals .....	53
6.4.1 Evaluation Against Research Aim.....	53
6.4.2 Evaluation Against Research Objectives .....	54
6.4.3 Evaluation Against Research Question.....	56
6.4.4 Technical Challenges .....	56
6.5 Summary .....	57
7. Conclusions.....	58
7.1 Summary of conclusions .....	58
7.2 Recommendations .....	59
7.3 Future work.....	59
References.....	61
Appendix A: Ethics certificate .....	64
Appendix B: Project Specification .....	65
Appendix C: Project Code .....	76

# LIST OF FIGURES

Figure 1 Gantt chart .....	25
Figure 2 Architecture flow for the project .....	28
Figure 3 JSON template.....	32
Figure 4 Testing Result of the API Endpoint 1 .....	40
Figure 5 Testing Result of the API Endpoint 2.....	40
Figure 6 Dialogue box to fetch patient data .....	44
Figure 7 Displaying the patient details .....	44
Figure 8 Patient details displayed .....	45
Figure 9 Executing 1st Time .....	46
Figure 10 Executing 2nd time .....	47
Figure 11 Testing with API failures .....	47
Figure 12 Executing 1st Time .....	47
Figure 13 Executing 2nd time .....	48



# LIST OF TABLES

Table 1 Sample record developed using the suggested format .....	33
Table 2 Details of category and settings.....	37
Table 3 Creation of API endpoints.....	39
Table 4 Response times of two API endpoints Virginia and Ohio .....	48
Table 5 API failure of Ohio Server .....	50
Table 6 Latency of API failure of Virginia – Server 1 .....	51

# 1.Introduction

## 1.1 Project Context

The healthcare industry in the UK together with all other industries across the globe has various challenges in handling and sharing of medical records. At the present time, the records are usually kept in separate databases of various healthcare facilities, including hospitals and clinics. The fragmentation that occurs in this system results in such things as duplicative testing, delay in diagnosis, and less than ideal patient management. This makes it difficult for the healthcare providers to obtain a complete history of the patient, which in turn creates a problem regarding continuity of treatment (Shuaib, et al, 2022). A centralised system for medical records sharing that can help solve these issues is a single store where all the information can be put and retrieved. As a result, this system can provide the benefit of what cloud storage can offer; namely, cloud storage solutions that can accommodate large data volumes. The traditional model of patient care is characterised by sharing of information and collaboration in real time, which is essential for quality delivery of services. In addition, it is possible to get rid of the redundancy and differences stemming from the current fragmented structure of the system (Salim, et al, 2022).

There are various benefits of cloud platforms for a centralised medical records system. They offer strong security features to ensure that patients' data is not exposed to the wrong people, such as encryption, permission levels, and security checks. The cloud-based systems are easily scalable which means that the healthcare organisations can be able to expand their storage as and when they require it. They afford high availability and reliability to ensure that the records of the patients are easily retrievable by the health care givers at any time without any form of interruption (Sun, et al, 2020). The last one is that cloud platforms make it simple to connect with other digital health technologies and services that can be used with the centralised system to increase its effectiveness. Thus, locating the medical records in one place allows the healthcare providers to have the right and updated information about the patient. This results in the enhancement of patient care as the healthcare givers will be in a position to make right decisions regarding the patient through the patient's history. Also, a centralised system helps in managing administrative procedures

as it cuts down the time and resources needed in data management. This in its turn helps healthcare staff to spend more time on the patients and less time on paperwork and, thus, enhances the general productivity of the organisation (Abouali, et al, 2020).

### 1.1.2 Need for centralised system

In the healthcare system in the UK, the absence of a unified system for exchanging medical records brings about major difficulties which hamper the delivery of efficient patient care and communication between the healthcare providers. Actually, there are many different hospital IT systems in use with a variety of cloud servers and data repositories, and their decentralised approach to data hampers the effective transfer of essential patient data when and where it is required (Makhdoom, et al, 2020). Such dispersion not only entails instrumental factors such as longer time of diagnosis and treatment but adds up to the danger of medical mistakes and the possibilities of redundant healthcare services. In addition, the lack of a single data format represents another hurdle. Not all systems can understand the information effectively which is provided by the rival methods. Therefore, patients may suffer in the form of excessive discontinuity of care, unnecessary diagnostic tests and poor treatment results. Besides that, healthcare providers encounter gaps in accessing the necessary details of a patient which hinders the decision-making process and ultimately the patient safety (Chen, et al, 2021). This problem can be addressed by changing the healthcare professionals' current practices through different specific methods that can ensure the secure and seamless data exchange, standardise identified data formats, and give healthcare providers timely access to comprehensive patient records, therefore significantly increasing the quality and efficiency of healthcare across the United Kingdom.

## 1.2 Aims & Objectives

### 1.2.1 Aim

To develop a centralised medical records sharing system using a cloud platform in the UK healthcare sector, facilitating seamless data transfer and access across different hospitals and healthcare organisations.

### 1.2.2 Research question

How can Health Level Seven (HL7) be effectively implemented in a cloud-based environment for medical data standardisation?

### 1.2.3 Project Objectives

- To design and implement a cloud-based infrastructure capable of securely storing and managing medical records.
- To integrate HL7 as a standardised format for medical data exchange within the system.
- To develop robust database management mechanisms ensuring data integrity, security, and compliance with regulatory standards.
- To enable efficient data upload and retrieval functionalities for healthcare organisations using the system.
- To create a user-friendly front-end interface using Python for easy access and interaction with the system.

## 1.3 Limitations

The main limitation that is associated with the project is categorised into 4 main areas of using cloud platforms in medical records sharing in the health care sector in the UK. Firstly, applying HL7 standards for a cloud-based infrastructure is technically challenging since it needs HL7 solutions that consist of the integration standards for various healthcare organisations. Secondly, the issue of security and privacy is very important even with all the measures in place as the system is dealing with patient's records which are considerate, sensitive in nature and prone to exposure. Specifically, language barriers, the need to meet GDPR and other healthcare regulations increase the level of changes companies need to make in line with the legal requirements. Thirdly, Certain complexities may be encountered in the scaling of the system, so accommodating a large record, data storage and the incorporation with other Information communication technologies may prove to be difficult. Finally, other objectives such as user adoption in care facilities and controlling the costs of institutional deployment also present other considerations that need a lot of planning with

the key stakeholders. Overcoming these limitations is central to realising the successful implementation and, thus, effective usage of the cloud-based medical records system in the UK healthcare context.

## 1.4 Report overview

The first chapter of the study discusses the background and reasons for choosing to design a centralised medical record sharing system on a cloud platform in the UK healthcare sector. It underlines the role of centralised medical records in sharing information between hospitals and the role of HL7 standards in data integration with the project's goals, objectives, and constraints. The literature review chapter discusses the current cloud-based medical record systems, standards, data transfer protocols, and the role of HL7 in enhancing interoperability and points to gaps in the literature. The methodology chapter describes the research approach, technologies and tools employed for system development, and evaluation methods. The experiments chapter captures the application of methodologies, system performance, and evaluation. The results chapter of the study focuses on data collected from experiments, system performance assessment, HL7 integration, and efficiency in relation to the set project goals. The conclusion reviews the main findings of the project, assesses its success, outlines implications for the UK healthcare sector, identifies potential research directions, and offers recommendations for implementing similar systems.

## 2.Literature Review

### 2.1 Outline

In the chapter on background research, a comprehensive analysis of cloud based medical record systems will be done involving a look at their structure, operation and efficiency in managing and securing medical records. It will explain general guidelines for the transfer of medical data and concerning data transfer and their significance to the general data protection and security. Based on HL7 Implementation, the changes in the interoperability of healthcare data will be analysed to describe how this standard helps to share information with other care organisations. Moreover, prior works in the literature pertaining to medical data transfer will also be discussed to evaluate the current research scenario, idea, and development in this area. Last, the chapter will come up with the research questions to fill the established gaps and form the research questions for the current study which will focus on the efficient, scalable and secure cloud-based system for predicting phishing URLs.

### 2.2 Importance of Health Level Seven (HL7)

Health Level Seven (HL7) is an international standard for the exchange, integration, sharing, and retrieval of health information. These standards are important since they facilitate interoperability between different healthcare systems. HL7 has a list of standard interfaces to which all HIT systems can align so that they can interoperate effectively. HL7 standards utilised in centralised medical records sharing systems assist in offering accurate and standard data directing between multiple healthcare givers. The HL7 standards refer to the way in which clinical information is presented to reduce the chances of mistakes as well as inconsistencies in the flow of that information. It is also critical to pathological diagnoses in a centralised system as there are many data sources with which the data must be consolidated and standardised (Setyawan, et al, 2021).

The HL7 standards must be implemented in a medical records system that is hosted on the cloud has several advantages. It also enhances the system as per the evolution in interfacing

and also makes it rather convenient for healthcare providers to transfer and obtain patient data over distinct interfaces. This is important in order to enhance cooperation/interprofessional relations in the processes of care delivery especially the circumstances where many caregivers are involved. Moreover, HL7 standards assist in the achievement of the legal factor as it has guidelines on the patient's health information privacy and security. Appropriate transmission of data benefits the patient by making their care process more efficient with HL7 integration. This shall assist the health care providers to have a full and or updated information on the particular patient, hence provide him or her with the required attention and care. Also, the use of standardised data formats ensures compliance with healthcare regulations and thus minimises the chances of incurring legal and financial consequences for non-compliance. This in turn increases confidence among patients and healthcare providers thereby improving the efficiency of the healthcare system (Maxi, et al, 2021).

## 2.3 Review of Cloud-Based Medical Record Systems

Electronic health record systems have nowadays become innovative solutions in the healthcare industry as they provide boundless benefits in usage, expansiveness, and cooperation among attendants of medical departments. These systems utilise elements of cloud computing to enable storing, archiving, and transferring of medical records through the internet eradicating limitations prevalent when records are in paper form or when they are stored and utilised in specific locations only (Mahajan, et al, 2022). Some of the significant advantages arising from the use of cloud-based medical record systems include; An advantage of cloud-based medical record systems is that patient records can be accessed over a number of health facilities hence enabling coordinate care delivery. One advantage of getting medical records in the cloud is that all the records can be made central and are easily available to healthcare practitioners in real-time regardless of their location thus increasing their chances of making proper decisions for their clients (Zhou, et al, 2021).

Also, cloud platforms for healthcare organisations provide scalability, thus enabling such organisations to upgrade the total quantity of storage and computation without many capital expenses on infrastructure. This scalability is important when addressing the growing influx

of medical information from healthcare structures within an organisation to include patients' data, diagnostic images, and physiological data from the medical equipment (Sivan, et al, 2021). Users should have control of the confidentiality of their records as well as secure storage of their records in case of illness. Major providers of cloud services employ high levels of security features including the use of encryption, access control, among other measures used to prevent breaches of patient sensitive data in the cloud including those required under HIPAA in America among others. They also help ensure that the information concerning patients is retained as confidential while at the same time allowing other relevant personnel in the delivery of healthcare services to access such information easily and efficiently (Benil, et al, 2020).

## 2.4 Standards and Protocols for Medical Data Transfer

Health information exchange is based on the use of exchange specifications and the use of a common standard that facilitates the sharing of data between healthcare systems. This paper will focus on two most commonly used frameworks known as HL7 v2 and HL7 v3 that describe how data should be formatted and Interoperability and exchanged in clinical and administrative structures. HL7 v2, which is firmly established and currently in the process of wide-scale deployment, defines formats for different healthcare transactions such as patient admit/discharge/transfer, laboratory tests, prescription, etc. It facilitates connectivity since through this standard, the healthcare application and systems which are involved have a universal language they can understand (Yaqoob, et al, 2022). While HL7 is the most memorable interoperation standards for medical data, other standards and protocols are also crucial to interoperation. The DICOM (Digital Imaging and Communications in Medicine) is used for the communication and management of medical images, especially concerning modality work and as a medical image print interface. FHIR (Fast Healthcare Interoperability Resources) is a contemporary standard of HL7 which is useful in exchanging healthcare information in an electronic manner. FHIR has adopted the use of intuitive web technologies such as Restful APIs consequently enhancing the real-time sharing of data amongst various healthcare applications/ software (Chang, et al, 2021). The standardised guidelines/protocols presented in this paper should be effectively implemented in the pursuit of interoperability in the healthcare domain thus allowing different healthcare providers to



securely and efficiently exchange patients' information between them. This process helps in the clinical and operational realisation of standardised data in healthcare settings to lower integration costs and improve the quality of patient care, particularly with respect to synthesising large datasets. With healthcare staying in the process of digitization, we continue experiencing interoperability issues, meaning it is critical to follow these standards to achieve benefits from a well-developed integrated healthcare delivery system (Pradhan, et al, 2021).

## 2.5 Impact of HL7 Implementation on Healthcare Data Interoperability

The use of HL7 standards has significantly affected the interoperability of healthcare data as it changes the approach to how clinical applications exchange data between various systems. HL7 is a type of framework that aims to establish a standard to allow one healthcare unit to understand and process data and formats of another dissimilar unit of healthcare. This feature is indispensable for linking the flows of information between hospitals, clinics, laboratories, and other related entities, which helps to improve cooperation and continuity of the medical processes (Morid, et al, 2021). Another effect of HL7 implementation that can be considered as a very important one is the change in clinical workflow. HL7, as a framework for exchanging health information, eliminates time consuming data entry and data matching tasks by prescribing the format and much of the content of the messages being transferred. As a result, it eliminates cases of high administrative cost which is linked to manual process and also reduces error hence enhancing the reliability of patient records. In this model of care, the healthcare providers have the capability of accessing the total patient information in real time so as to aid in clinical decision making and ultimately result in better patient outcomes (Trigui, et al, 2020). This is especially because HL7 is also used in the practical implementation of EHR systems to guarantee that they can communicate with other health IT solutions. This seamless integration is very necessary to facilitate a well-connected continuum of care to enable patients records to be shared seamlessly between the various systems. For instance, HL7 messaging standards would make it easier to compile and transfer laboratory results, radiology reports, and medication orders between systems, key components that would give a patient's full medical record (Romero, et al, 2020).

## 2.6 Research gap

Although there is progress in the extension of cloud computing across their healthcare system, and with standardisation of data across multi-disciplinary domains in healthcare, there are still several key issues with HL7 implementation and adoption that cannot be effectively bridged with a centralised cloud based medical records system. Previous research work has mainly only been targeted on individual features of cloud security, exchange of data, and standardisation whereas the interdependent integration of all these factors has not been studied in detail and has a large number of issues. Although HL7 offers comprehensive models for the exchange of medical data, its instantiations in a cloud computing setting remain under-discussed despite their potential as current studies suggest. In particular, the active character of the platforms and the rather vast spectre of applications in the spheres of healthcare demand a stable and flexible concept of interacting with HL7 that guarantees the flawless exchange of data between the systems and involved institutions. Security and privacy of medical records in the cloud-based environment have been explored in detail; however, issues remain because of the incorporation of technologies such as cloud. These advanced security measures are not accompanied by HL7 integration models in the literature that offer holistic protection of these medical records especially during the transmission and storage segments hence the need to fill this gap enhancing protection of patients' medical records.

## 2.7 Implications for the Proposed Project

The proposed project seeks to design a medical record sharing platform through cloud systems in the UK health sector. It is the kind of advancement that aims at enabling entities within different hospitals and health care institutions to share and exchange data fluently. Through the implementation of Health Level Seven (HL7) as the organisation's standard for HL7 health information exchange format, the system strives to achieve compatibility standards when sending the medical information. Sound database management tools for the maintenance of data accuracy, security and compliance with the relevant standards will also be instituted. Data uploading and downloading functions will help to rationalise the administrative tasks, whereas the Python-based interface will contribute to the improvement

of the usability and the interaction for health care providers. In summary, this project gives direction to increase data compatibility, increase security, boost health approaches, and commonly, raise the degree of patient care across the health care platform in the United Kingdom.

## 2.8 Conclusion

In the background research chapter, more information on the developed solutions has been researched with emphasis on medical record systems based on cloud computing environment, architecture, functionality, and advantages. It has stressed the roles of standards and protocols such as HL7, DICOM, and FHIR in sharing data with adherence to common communication formats involving the use of standards by different systems in the context of healthcare. The chapter also analysed the great improvement on the interoperability of healthcare data after implementing HL7, and how it will help enhance clinical work process and decisions. Next, the literature has been analysed to present the recent research on the medical data transfer in which the blockchain, elliptic curve cryptography, and multiparty homomorphic encryption have been introduced as the progressive solutions for the secure medical data. However, the conducted research reveals a gap that exists within the implementation of these measures alongside HL7 standards within a cloud environment. Therefore, there is more research needed on integrated protection models that incorporate safe transmission and archiving of health information along with the advantages of cloud environments.

The current studies have revealed that block chain can prove useful in guaranteeing the integrity as well as the transparency of the information, ECC can be efficient in encryption of data, and finally, the multiparty homomorphic encryption can be used as a protocol for protecting the privacy of data transferred in healthcare. These technologies present potential solutions to protect such information in decentralised networks, which is crucial for the suggested, centralised medical records sharing system. The deployment of the high-level security measures in combination with existing healthcare specifications as HL7 in cloud environments raise important challenges. Albeit each technology does well in addressing specific aspects of security, there is still a weakness in how they integrate well. This goes to show that it is not easy to ensure total protection of the medical records throughout the

transmission stage as well as in storage stages inclusive of cloud storage. As earlier pointed, there are doubts on the security and the privacy of medical data even as technology enhances. These and other events including hacking, unauthorised access and meeting regulatory requirements such as GDPR remain important to call for a good security measure. Such findings call for the creation of a holistic security framework that covers medical data protection consistent with the regulation and norms.

## 3. Project Management

### 3.1 Project Methodology choice

The waterfall model is a sequential project management approach where progress flows in one direction through defined phases: Analysis, design, the construction phase, the testing phase, installation, and the servicing and upgrading of the system. This method is obviously suitable for the specific case of the centralised medical records sharing system project, because its work is built on clear phases and systematic approach. It confirms that every phase is accomplished adequately and with precision, which is significant when dealing with patients' records and personal information, as well as preserving compliance with legislation such as GDPR and HIPAA. The documentation of the phases within the waterfall model is rigid and ensures the inclusion of key stakeholders in the project, something that is vital when creating a strong foundation of security on cloud computing, the integration of HL7 standard and proper development of the databases. According to the same methodology, the project seeks to implement a functional and sustainable system which can significantly improve access, integration and utilisation of the data in multiple organisations to support health care delivery.

### 3.2 Professional Issues

From the analysis of the proposed research an array of professional issues emerges when implementing a project on the centralization of medical records sharing through a cloud platform in the UK health care sector.

First of all, regulatory compliance, especially in light of the GDPR and the HIPAA, should be a primary concern due to the nature of the data in question – patients' data.

Secondly, obtaining HL7 standards' interoperability and standardisation data across different healthcare systems present technical issues that must be solved.

Overcoming the challenges of achieving greater integration and standardisation by implementing HL7 standards into healthcare systems is problematic on a number of

technical considerations. The problems of data integration and mapping stem from the use of different data sources and sharing of data meaning across organisations. It is important to always make sure that the data collected and stored is of good quality and free of any inconsistencies. Real-time applications require scaling up and dealing with various aspects of performance for dealing with higher volumes of data without delay. Issues to do with security and privacy, like the protection of data while in transit and putting in place the proper authentication mechanisms become crucial.

Thirdly, the security of patient's information will need to be safeguarded during transmission and storage in clouds, besides the security of data from threats. However, the scalability of the used system to deal with data volume and still having efficiency and performance indicators increases the level of difficulty.

Lastly, ensuring the use of the system by involving user-interface design that makes it easy for the users to use the system as well as training of the healthcare practitioners so that they can use the system optimally. Solving these professional issues will be central to fulfilling the goals of the project in improving avenues of healthcare and patients' care in the United Kingdom.

### 3.3 Conclusion

The proposal for centralising medical record sharing through a platform in the UK healthcare system has a number of professional challenges, but these can be conquered. The politics and regulations of storing personal, especially Americans' and Europeans' medical records limit the application of AI as GDPR and HIPAA compliance is crucial. Some of the technical challenges that need to be sorted out are: As HL7 standards have to be incorporated for the integrated healthcare systems interoperability the appropriate design for the secure exchange of the data through the cloud has to be established and effective security measurements for the cloud data transfer and storage have to be developed to protect the patients' information. The project also strives to be scalable so as to accommodate large amounts of data as well as being efficient and possess good performance. However, the universal concerns remain with the user-interface design along with adequate education of the health-care professionals for the proper utilisation of the system. Through addressing

these issues methodologically and with regard to professional guidelines, the project aims at increasing the quality of healthcare, promoting better patient outcomes, as well as optimising the data processing in the healthcare facilities in the UK.

## 4.Methodology

### 4.1 Specification and Discussion of the requirements

With the main goal of designing a cloud environment for the efficient data sharing and access by the multiple organisations within the United Kingdom's healthcare sector. Some of the important goals also include creating a suitable architecture necessary for the management of large volumes of medical records which are often sensitive and large HL7 compatibility to ensure proper data exchange along with good methods of handling large amounts of data which conform to regulatory requirements for instance GDPR and HIPAA. Moreover, the system shall support data upload and retrieval functions to ensure that the healthcare providers get timely and comprehensive patients' information. In our case, Python and the Flask framework are recommended to create a comprehensible and convenient front-end that defines interactions. The project also concerns the methods of security enhancement, such as encryption, access controls, and secure communication channel for the patient's sensitive data. Regular testing, monitoring and compliance audit form a strategically vital component in the functionality and efficiency of the system. These clear objectives and needs set the direction for the project and make sure it will tackle the main issue of disparate health information and improve patients' outcomes and organisational performance in the sphere of healthcare.

### 4.2 Analysis and Discussion of the IT Methodology

#### 4.2.1 Methodological Framework

- Design a cloud system of high scalability and security for a centralised database of medical records, which guarantees its accessibility and reliability for medical professionals.
- Introduce HL7 as the emerging industry protocol for sharing medical data between multiple healthcare providers; by this way data exchange is easier and more accurate.
- Use reliable data management practice to keep data non-manipulable, secure, and compliance with data protection laws and regulations with sensitive patient information protected from unauthorised access or data leaks.



- Implement protocols for healthcare organisations to upload and pull up their medical data on the internet quickly and securely reducing the delays which commonly occur in accessing patients' critical information and at the same time boosting the operational efficiency.
- Develop using Python an accessible and user-friendly front-end interface which will allow the healthcare professionals to have trouble-free access to medical records and communication channel to the system's central node.

## 4.3 Implementation Strategy

### 4.3.1 Time Management

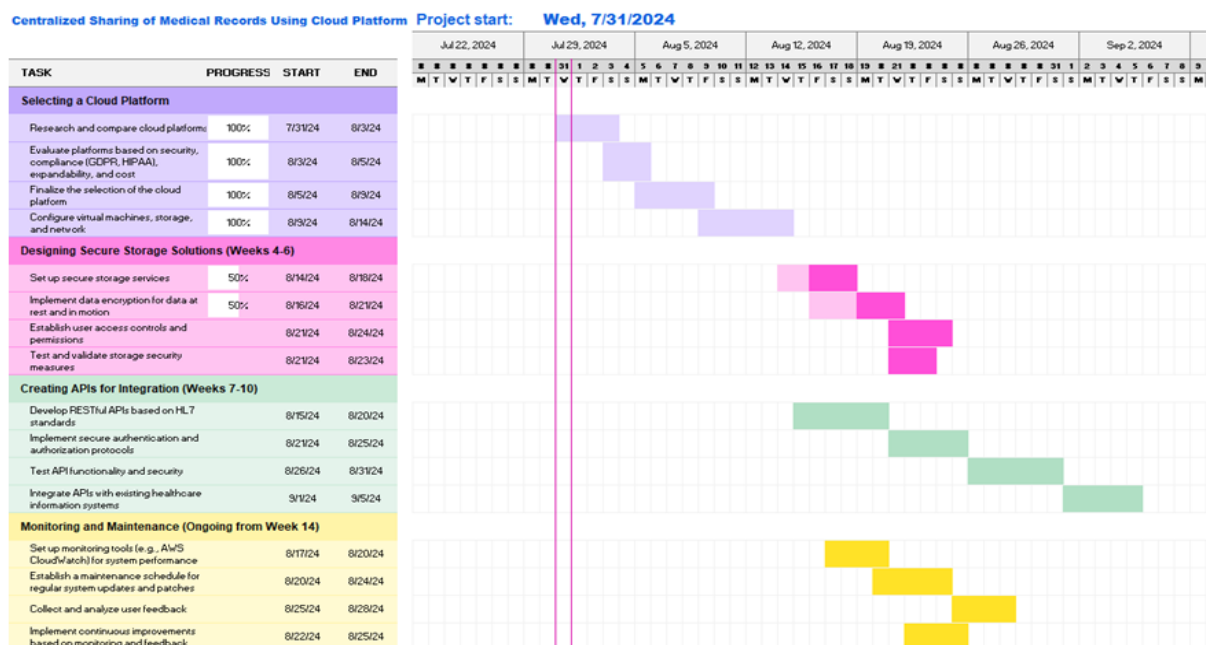


Figure 1 Gantt chart

Time management is an indispensable part of the efficient functioning of the cloud-based medical records sharing system. The project is divided into separate activities, which have their specific start date and end date, main dependencies and their durations in the Gantt chart. Starting from the identification of the right cloud platform and moving to the architecture of safe storage, the development of APIs for integration, the adherence to the standards, and the implementation of the monitoring and the maintenance procedures, each stage is designed to work cohesively. Analogue to their dependencies, the corresponding

tasks are planned in a way to avoid getting stuck at a specific phase of the process. Assessments as well as changes based on feedback and status updates are important in sustaining the project's progress and time-bound. Well, this is not only a strong working frame for the project tracking and resource management but also a guarantee for completing all works on schedule and meeting the deadline for orienting a result in the construction of a sound and secure medical record sharing system.

## 4.4 Summary

The centralised medical records sharing system project lays down the proposal of a cloud-based framework, which is central and more specialised in the sharing of data with all the healthcare associations in the United Kingdom. Some of the characteristic goals are to develop a highly available IT architecture, adopt the HL7 protocols for transferring information, and meet requirements outlined by legal frameworks like GDPR or HIPAA. The project management framework also entails the application of the waterfall model where there are clear stages ranging from requirements gathering, system design, system implementation, system testing, deployment and system maintenance. During the requirements specification, goals like low-cost data sharing, record storage, and addressing statutes are defined. During the system design phase, the emphasis is on the system security and its architecture along with data storage, API and HL7. This stage entails writing backend services in secure scalable languages and creating an appealing front-end with Python Flask. Testing confirms if the entire system is working properly and the actual setting up of the system typically credited to deployment. Maintenance also refers to continuing assessment as well as enhancement of the working environment.

## 4.5 Conclusion

The creation of the unified medical records sharing platform is the important step that would contribute to the increase of the overall healthcare sector effectiveness and the quality of British healthcare services delivery for the people of the United Kingdom. Through having a cloud infrastructure as the underlying support with secure means of data sharing accompanied by HL7 standards, the project should solve the problem of lacking cohesiveness within data and healthcare systems. The use of python and flask in frontend

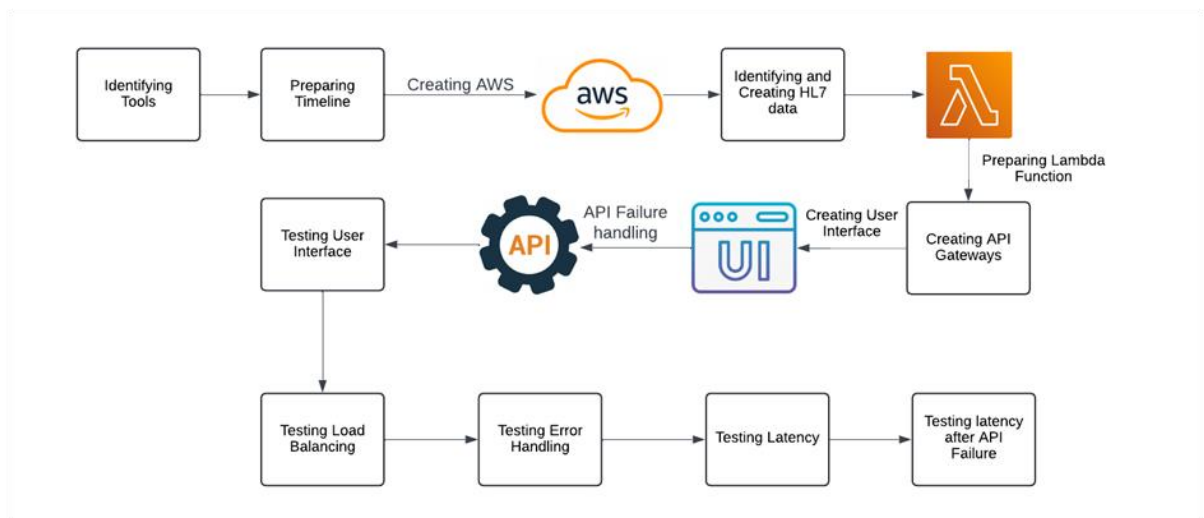
would enhance the modality and ease of operating the records as well as access to patient details.

## 5. Implementation

The experiment chapter explains the methodology of the cloud-based medical records sharing system with the help of AWS Lambda, AWS API Gateway and Python programming. It explains how to create API Endpoints, how to perform load balancing and how to do performance testing. The chapter also compares the system's performance, response time and how it handles API errors. Results of these experiments are then presented in order to evaluate the general efficiency of the proposed solutions and identify further improvement possibilities.

### 5.1 Materials / Tools / Data Collection

#### 5.1.1 Method of Implementation



**Figure 2 Architecture flow for the project**

#### 5.1.2 Tools Used

AWS Lambda was selected due to its capability to execute code in response to specific triggers without requiring the setup of new servers, thus being the right choice for this project's server less model. It supports auto scaling, better resource utilisation and cost optimization since the user pays only for the actual time of computing. AWS Lambda is also

compatible with AWS API Gateway, thus making it easy to manage API and request and response handling.

AWS API Gateway was used in developing, deploying and managing API, which rendered a strong interface for invoking the Lambda function. It enables load balancing, monitoring, and authorization therefore making the system reliable and secure. For the implementation of API and handling the data, Python with its libraries like Flask, JSON, requests and time was used as it is very easy to use. It uses Flask for building lightweight Web applications, JSON for parsing data and requests for working with HTTP requests. All these tools put together make the work much easier and effective in the development and management of the cloud-based medical records system.

## 5.2 Implementation Timeline

*Week 1: Initiating the project and project preparation.*

- Day 1-2: Identify the project scope, the purpose and the goals of the project and the requirements that need to be met.
- Day 3-4: Configure the AWS and the required permission.
- Day 5-7: Plan the project in detail including the timelines and the milestones to be achieved.

*Week 2: Design and Initial Setup.*

- Day 1-2: Plan the architecture of the cloud-based platform that will contain the medical records.
- Day 3-4: We will set up AWS Lambda and AWS API Gateway.
- Day 5-7: Design the first set of API and incorporate them with AWS lambda functions.

*Week 3: Focused on the development and integration.*

- Day 1-3: Create API endpoints in Python with the help of Flask and JSON.
- Day 4-5: Implement HL7 for the format of medical data storage and management.

- Day 6-7: Load balancing techniques to be developed and put in practice.

*Week 4: Testing & Optimization.*

- Day 1-2: Carry out latency tests and thus assess the performance of the API endpoints.
- Day 3-4: Introduce a test for the load balancing and the error handling when an API returns an error.
- Day 5-7: Tweak the system according to the results of the test and fine tune the system.

*Week 5: Front-end development and the final adjustments*

- Day 1-3: Used to design and implement the UI of the system that will be used in interacting with the data.
- Day- 4-5: Integration of the front end with the back-end services in the subsequent days four and five.
- Day 6-7: Final fine tuning of the system and checking the system's stability.

*Week 6: Final Review and Deployment of the Software Project.*

- Day 1-2: Perform a final check on the system in terms of functionalities and user friendliness.
- Day 3-4: Create deployment packages and user manuals.
- Day 5-7: Install the system in the production environment and perform a last check to make sure that all is well.

## 5.3 Data Collection

### 5.3.1 HL7 – Health Level Seven International – Medical Data Format.

The FHIR (Fast Healthcare Interoperability Resources) specification of the “Patient” resource gives a comprehensive approach on how to capture and exchange any demographic and administrative details about a person, human or an animal that is a recipient of health care services. This is useful in managing patients’ information in different health care services such as medical, psychiatric, social services and many others.

The specification describes the basic data elements of the Patient resource including the active and other identifiers, the name, the contact details, the gender, the birth date and time, the marital status, and many others. It also has information on how to address certain situations such as the deceased, twins and transfer of patient information from one system to another.

These are the management of patient identifiers, the merging and linking of duplicate patient records, and the distinctions between the Patient resource and others like the Person, Related Person, and Linkage. The Patient resource is intended for proper and standardised description of patient information; the resource is defined by certain rules and constraints to improve data quality and create a common basis for information exchange between different systems. The specification also features JSON, XML and Turtle for the representation of the resource in other systems.

The HL7 medical data format is available from the following website.

<https://hl7.org/fhir/patient.html>

## 5.3.2 HL7 Medical Data Format

### JSON Template

```
{
  "resourceType": "Patient",
  // from Resource: id, meta, implicitRules, and language
  // from DomainResource: text, contained, extension, and modifierExtension
  "identifier": [{ Identifier }], // An identifier for this patient
  "active": <boolean>, // Whether this patient's record is in active use
  "name": [{ HumanName }], // A name associated with the patient
  "telecom": [{ ContactPoint }], // A contact detail for the individual
  "gender": "<code>", // male | female | other | unknown
  "birthDate": "<date>", // The date of birth for the individual
  // deceased[x]: Indicates if the individual is deceased or not. One of these 2:
  "deceasedBoolean": <boolean>,
  "deceasedDateTime": "<dateTime>",
  "address": [{ Address }], // An address for the individual
  "maritalStatus": { CodeableConcept }, // Marital (civil) status of a patient
  // multipleBirth[x]: Whether patient is part of a multiple birth. One of these 2:
  "multipleBirthBoolean": <boolean>,
  "multipleBirthInteger": <integer>,
  "photo": [{ Attachment }], // Image of the patient
  "contact": [{ // A contact party (e.g. guardian, partner, friend) for the patient
    "relationship": [{ CodeableConcept }], // The kind of relationship
    "name": { HumanName }, // I A name associated with the contact person
    "telecom": [{ ContactPoint }], // I A contact detail for the person
    "address": { Address }, // I Address for the contact person
    "gender": "<code>", // male | female | other | unknown
    "organization": { Reference(Organization) }, // I Organization that is associated with the contact
    "period": { Period } // The period during which this contact person or organization is valid to be contacted relating to
    his patient
  }],
  "communication": [{ // A language which may be used to communicate with the patient about his or her health
    "language": { CodeableConcept }, // R! The language which can be used to communicate with the patient about his or her health
    "preferred": <boolean> // Language preference indicator
  }],
  "generalPractitioner": [{ Reference(Organization|Practitioner|PractitionerRole) }], // Patient's nominated primary care provider
  "managingOrganization": { Reference(Organization) }, // Organization that is the custodian of the patient record
  "link": [{ // Link to a Patient or RelatedPerson resource that concerns the same actual individual
    "other": { Reference(Patient|RelatedPerson) }, // R! The other patient or related person resource that the link refers to
    "type": "<code>" // R! replaced-by | replaces | refer | seealso
  }]
}
```

Figure 3 JSON template

The sample record in figure 3 is derived from the FHIR (Fast Healthcare Interoperability Resources) format, which organises patient data in a precise and coherent manner. Some of the information that this record contains include; The patient's complete name, phone contact, email contact, gender, date of birth, marital status, and the patients' address. It also has information on the patient being active, the patient's general practitioner and the entity attending to the patient. Also, the patient's details on the next of kin, and the preferred language that should be used to communicate with the patient and the relationship with other patients is also provided. This format helps in the proper organisation and easy retrieval of health care information.



The following table 1 is a sample record developed based on the suggested format.

**Table 1 Sample record developed using the suggested format**

<pre>{   "resourceType": "Patient",   "identifier": [     {       "use": "official",       "system": "http://hospital.smarthealth.org/patients",       "value": "P123456"     }   ],   "active": true,   "name": [     {       "use": "official",       "family": "Doe",       "given": [         "John"       ]     }   ],   "telecom": [     {       "system": "phone",       "value": "+1-555-123-4567",       "use": "mobile"     },     {       "system": "email",       "value": "johndoe@example.com",       "use": "home"     }   ],   "gender": "male",   "birthDate": "1985-04-23",   "deceasedBoolean": false,   "address": [     {       "use": "home",       "line": [         "123 Main St"       ],       "city": "Springfield",       "state": "IL",       "postalCode": "62704",       "country": "USA"     }   ],   "maritalStatus": {</pre>	<pre>"coding": [   {     "system": "http://terminology.hl7.org/CodeSystem/v3-MaritalStatus",     "code": "M",     "display": "Married"   } ],   "multipleBirthBoolean": false,   "photo": [     {       "contentType": "image/jpeg",       "url": "http://hospital.smarthealth.org/patients/P123456/photo.jpg"     }   ],   "contact": [     {       "relationship": [         {           "coding": [             {               "system": "http://terminology.hl7.org/CodeSystem/v2-0131",               "code": "N",               "display": "Next of Kin"             }           ]         }       ],       "name": {         "family": "Doe",         "given": [           "Jane"         ]       }     }   ],   "telecom": [     {       "system": "phone",       "value": "+1-555-765-4321",       "use": "mobile"     }   ],   "address": {     "line": [       "456 Oak St"     ],     "city": "Springfield",     "state": "IL",     "postalCode": "62705",     "country": "USA"   },   "gender": "female",</pre>
--	---

```

    "organization": {
      "reference": "Organization/1",
      "display": "Springfield General Hospital"
    },
    "period": {
      "start": "2020-01-01"
    }
  },
  "communication": [
    {
      "language": {
        "coding": [
          {
            "system": "urn:ietf:bcp:47",
            "code": "en",
            "display": "English"
          }
        ]
      },
      "preferred": true
    }
  ],
  "generalPractitioner": [
    {
      "reference": "Practitioner/123",
      "display": "Dr. Emily Smith"
    }
  ],
  "managingOrganization": {
    "reference": "Organization/1",
    "display": "Springfield General Hospital"
  },
  "link": [
    {
      "other": {
        "reference": "Patient/P654321",
        "display": "Jane Doe"
      },
      "type": "seealso"
    }
  ]
}

```

### 5.3.3 AWS Lambda Function Creation

#### 5.3.3.1 Creation of Function in AWS Lambda

The given function is an AWS Lambda function which is developed in Python language. AWS Lambda enables you to execute code in response to a specific event, for instance, receiving HTTP requests or modification of data. This function `lambda_handler` is the starting point of the AWS Lambda execution where the code is going to be written.

#### 5.3.3.2 Steps Included in creating

- Step 1: Declare the `lambda_handler` function that is the entry point for AWS Lambda.
- Step 2: Generate a dictionary named `patient_details` that will hold all the information that the patient has including the format used by FHIR.

- Step 3: Now, we need to convert the `patient_details` dictionary into JSON string format and that can be done using `json.dumps()`.
- Step 4: Create a response entity that has a status code of 200 and contains the patient details in JSON format as well as the appropriate HTTP headers.
- Step 5: Return the response dictionary, which AWS Lambda will pass back to the requester.

#### *5.3.3.3 Lambda Handler Function*

The `lambda_handler` function is created with the purpose to provide patient information in the JSON format that is conforming to FHIR. It stands as a fake API which can be invoked by an event like an HTTP request via Amazon API Gateway. The function takes the event and formats the patient information and gives the formatted patient information as the response.

#### *5.3.3.4 Patient Data as Output*

The data of the patient is presented in a structured manner using the FHIR (Fast Healthcare Interoperability Resources) standard. This is a protocol for interchanging healthcare information in electronic format. In this function the data is presented as a dictionary in Python language.

#### *5.3.3.5 Output Data Format*

The function returns an HTTP response which contains the status code 200, which shows that the operation was successful. The body of the response is a JSON string which contains the patient details. The response also includes a header specifying Content-Type: It uses 'Content-Type: application/json', meaning that the body of the request is in JSON.

## 5.4 API Gateway

### 5.4.1 API and Integration of Function with API Gateway

API stands for Application Programming Interface which is basically a way by which two or more software applications can interact with each other. AWS API Gateway is a service which provides creation, publication, management, monitoring and securing of APIs. When

it comes to the use of Lambda function with API Gateway, API Gateway is a front-end that triggers Lambda function whenever an HTTP request is received. For example, there is the Lambda function that has been described in the previous section, which returns patient information in the FHIR format. This Lambda function can then be deployed through the API Gateway which will give it a RESTful interface which can be used to make HTTP requests to the function.

### 5.4.2 Purpose of Using API Call for Lambda Function

The primary reason is to make the function callable via HTTP/HTTPS as a simple RESTful web service. This approach enables the client such as a web or mobile application to call the Lambda function via HTTP. For instance, the Lambda function which is responsible for retrieving patient information can be invoked through API Gateway upon an HTTP event. API Gateway is the interface that supports this kind of interaction and it takes the request of the API and pass it to Lambda function. This way the user can define his own logic to be performed within the Lambda function without having to worry about any server setup.

### 5.4.3 Steps in API Gateway Configuration

#### 5.4.3.1 *Integrate Lambda Function*

Define Integration Settings: In API Gateway, go to New API and create a new API then go to Resources and create a new resource. Set the way of accessing the function (for example using GET method) and connect it with the Lambda function. For the example function, set the Function name and the region and turn on Lambda Proxy Integration to pass through the entire request and response.

#### 5.4.3.2 *Configure Method Request Settings*

Authorization and Validation: If authentication is not required then set the authorization type to NONE. It is required to state whether an API key is required and enable request validation such as to check the request body to meet the expected format.

#### 5.4.3.3 Deploy Using Stage

Create and Deploy to a Stage: New stage should be added for example prod and then deploy the API. This step exposes the API endpoint to the clients. You can set up stage specific configurations including the source of API key, content encoding and the type of endpoint that you require for instance Regional. Table 2 gives the information about the category, setting and details.

**Table 2 Details of category and settings**

Category	Setting	Details
<b>Method Request</b>	Authorization	NONE
	API key required	FALSE
	Request validator	Validate body
	SDK operation name	Generated based on method and path
<b>Integration Settings</b>	Integration type	Lambda Proxy Integration
	Lambda Region	eu-north-1
	Lambda function	function1

	Timeout	Default (29 seconds)
<b>Stage Settings</b>	API key source	Header
	Content encoding	Inactive
	API endpoint type	Regional

#### 5.4.4 Creation of Two API Endpoints

Two AWS Lambda functions, namely function1 and function2 have been created and deployed in different geographical locations to attend to the requests and provide patient information. Every function is linked to a certain REST API created through the use of API Gateway. These endpoints are configured to act as entry points that would invoke the specific Lambda functions within the system to handle the <https://y18va6f6rc.execute-api.us-east-1.amazonaws.com/Server1>.

The URL for this endpoint is https: The culture change initiative is to be implemented in the organisation and it is referred to as y18va6f6rc. execute-api. us-east-1.amazonaws.com/Server1. When a request is made to this endpoint, it calls function1 and response is made in Json format of patient details provided that the status code is 200.

The second REST API endpoint is located in the Ohio region in US East (US-EAST-2). Its URL is <https://unbf0byjdc.execute-api.us-east-2.amazonaws.com/Server2>. This endpoint shall invoke function2 and also send the patient details in the JSON format with status code 200. Table 3 shows the details about the API endpoints creation including details such as location, URL, lambda function, response code and format.

**Table 3 Creation of API endpoints**

Endpoint	Location	URL	Lambda Function	Response Status Code	Response Format
API Endpoint 1	N. Virginia (US-EAST-1)	https://y18va6f6rc.execute-api.us-east-1.amazonaws.com/Server1	function1	200	JSON
API Endpoint 2	Ohio (US-EAST-2)	https://unbf0byjdc.execute-api.us-east-2.amazonaws.com/Server2	function2	200	JSON

The both endpoints are supposed to return patient information when they are called and the status code 200 means everything is OK. The JSON response from each endpoint will include patient data which is formatted in the FHIR protocol. This setup is beneficial in the sense that the Lambda functions can be accessed from various regions thereby making them readily available and may also provide faster response to users in those regions.

## 5.5 Testing

Testing the lambda function API end points which has been created in the Virginia and Ohio regions and it contains the patient's data. It can reflect the patient data when someone hits the endpoint. This is the backend test when we are hitting end point from AWS. So, this has been integrated with the UI so that we will get the patient data in the customised way. However, now we are testing the API end point in AWS.

The endpoints have been tested and confirmed that working as per expected are shown below.

```
← → ↻ y18va6f6rc.execute-api.us-east-1.amazonaws.com/Server1
netty-print ✓

"statusCode": 200,
"body": "{\n  \"resourceType\": \"Patient\", \"identifier\": [{\n    \"use\": \"official\", \"system\": \"http://hospital.smarthealth.org/patients\", \"value\": \"P:\n    \"official\", \"family\": \"Doe\", \"given\": [\"John\"]}], \"telecom\": [{\n    \"system\": \"phone\", \"value\": \"+1-555-123-4567\", \"use\": \"mobile\"}, {\n    \"use\": \"home\"}], \"gender\": \"male\", \"birthDate\": \"1985-04-23\", \"deceasedBoolean\": false, \"address\": [{\n    \"use\": \"home\", \"line\": [\"123 Main\n    \"postalCode\": \"62704\", \"country\": \"USA\"}], \"maritalStatus\": {\n    \"coding\": [{\n    \"system\": \"http://terminology.hl7.org/CodeSystem/v3-MaritalStatus\", \"\n    \"multipleBirthBoolean\": false, \"photo\": [{\n    \"contentType\": \"image/jpeg\", \"url\": \"http://hospital.smarthealth.org/patients/P123456/photo.jpg\"}], \"\n    \"http://terminology.hl7.org/CodeSystem/v2-0131\", \"code\": \"N\", \"display\": \"Next of Kin\"}], \"name\": {\n    \"family\": \"Doe\", \"given\": [\"Jane\"]},\n    \"5-765-4321\", \"use\": \"mobile\"}], \"address\": {\n    \"line\": [\"456 Oak St\"], \"city\": \"Springfield\", \"state\": \"IL\", \"postalCode\": \"62705\", \"\n    \"organization\": {\n    \"reference\": \"Organization/1\", \"display\": \"Springfield General Hospital\"}, \"period\": {\n    \"start\": \"2020-01-01\"}], \"communication\": {\n    \"ietfbc\": \"47\", \"code\": \"en\", \"display\": \"English\"}], \"preferred\": true}], \"generalPractitioner\": [{\n    \"reference\": \"Practitioner/123\", \"\n    \"managingOrganization\": {\n    \"reference\": \"Organization/1\", \"display\": \"Springfield General Hospital\"}, \"link\": [{\n    \"other\": {\n    \"reference\": \"Patie\n    \"sealso\"}], \"\n    \"headers\": {\n    \"Content-Type\": \"application/json\"\n    }\n  }\n}
```

Figure 4 Testing Result of the API Endpoint 1

```
← → ↻ unb0byjdc.execute-api.us-east-2.amazonaws.com/Server2
netty-print ✓

"statusCode": 200,
"body": "{\n  \"resourceType\": \"Patient\", \"identifier\": [{\n    \"use\": \"official\", \"system\": \"http://hospital.smarthealth.org/patients\", \"\n    \"official\", \"family\": \"Doe\", \"given\": [\"John\"]}], \"telecom\": [{\n    \"system\": \"phone\", \"value\": \"+1-555-123-4567\", \"use\": \"\n    \"use\": \"home\"}], \"gender\": \"male\", \"birthDate\": \"1985-04-23\", \"deceasedBoolean\": false, \"address\": [{\n    \"use\": \"home\", \"li\n    \"postalCode\": \"62704\", \"country\": \"USA\"}], \"maritalStatus\": {\n    \"coding\": [{\n    \"system\": \"http://terminology.hl7.org/CodeSystem/v3-\n    \"multipleBirthBoolean\": false, \"photo\": [{\n    \"contentType\": \"image/jpeg\", \"url\": \"http://hospital.smarthealth.org/patients/P123456/\n    \"http://terminology.hl7.org/CodeSystem/v2-0131\", \"code\": \"N\", \"display\": \"Next of Kin\"}], \"name\": {\n    \"family\": \"Doe\", \"give\n    \"55-765-4321\", \"use\": \"mobile\"}], \"address\": {\n    \"line\": [\"456 Oak St\"], \"city\": \"Springfield\", \"state\": \"IL\", \"postalCode\n    \"organization\": {\n    \"reference\": \"Organization/1\", \"display\": \"Springfield General Hospital\"}, \"period\": {\n    \"start\": \"2020-01-01\n    \"urn:ietfbc\": \"47\", \"code\": \"en\", \"display\": \"English\"}], \"preferred\": true}], \"generalPractitioner\": [{\n    \"reference\": \"Pract\n    \"managingOrganization\": {\n    \"reference\": \"Organization/1\", \"display\": \"Springfield General Hospital\"}, \"link\": [{\n    \"other\": {\n    \"refe\n    \"sealso\"}], \"\n    \"headers\": {\n    \"Content-Type\": \"application/json\"\n    }\n  }\n}
```

Figure 5 Testing Result of the API Endpoint 2

## 5.5.1 Creation of User Interface

### 5.5.1.1 Front Page: Fetch Patient

The `fetch_patient`. An html file is created in this project to allow for an easy and convenient way of retrieving patient data from APIs. It employs the HTML and Bootstrap for its styling and the site's capability to adapt in various screen sizes. The given web page has a title labelled as "Centralised Medical Data" and has a button in the centre with a label "Fetch Patient Data". Upon clicking the button, it sends a HTTP GET request to the Flask application's endpoint `/fetch`. This endpoint is the starting step in the data extraction of the patient information from the APIs.

The layout of the page is done using Bootstrap's grid system to make sure that the content is well aligned and looks good on different devices. The `<a>` tag with the `btn` and `btn-primary` classes generate a button that navigates the user to the `/fetch` page. This approach makes



it possible for the users to be able to interact with the application and get patients' information by just clicking the button. A notion of simplicity has been adopted in the design to ensure that the users are not confused while using the app.

#### *5.5.1.2 Output page : output.html*

The output.html file is intended to present the patient data obtained from the APIs in a neat way in order to improve the readability. It also adopts the use of HTML and Bootstrap in order to provide a good looking and effective way of presenting the data. The page format is quite simple and clear, the patient's data is presented in the table with the information about the API and time taken for the request. In case data fetch is successful, then it displays patient details in the form of a table and contains fields like patient name, contact details and address. The table used here is formatted using Bootstrap classes so as to make it as easy to read and visually appealing as possible.

Besides presenting the data of the patient, the page also indicates the API endpoint that was called and the time which was taken to process the request. This is beneficial for the debugging of the code and for checking the efficiency of the application. If there is an error or if all the API calls go wrong, then the page shows an error message saying that data could not be fetched. This approach is very effective in that it returns detailed information to users and it is very easy for the users to understand the results of their data fetch request.

#### *5.5.1.3 Main File : app.py*

The app.py file is the heart of the Flask application which deals with the handling of API calls and rendering of Web pages. It sets up a Flask server with two main routes: there is the root route (/) and /fetch route.

1.API Management and Data Fetching: The app.py file continues with the list of API Endpoints from which the patient data will be pulled from. The fetch\_patient\_data() function uses these endpoints in a round-robin fashion to make HTTP GET requests to the API. In case of a successful API call, it parses the response and retrieves patient data and then returns this data together with the URL used for API call and the total time spent on errors. If all the APIs fail then it stack the error time and return

None as output. The function also has exception handling to make the program very stable and can be used in any application.

2.Route Handlers and Rendering: The root route of the application is going to be the `fetch_patient`. The second file is an HTML template that includes a button which starts the data fetching process. On the click of the button, a request is made to the `/fetch` route. This route activates the `fetch_patient_data()` function, computes the time taken by the request and then displays the result. The HTML template that is populated with the patient data, the details of the API utilised, and the time taken. In case of failure of data retrieval, an error message is displayed instead. The application is run in the debug mode, this is an advantage especially when it comes to development and problem solving.

Please refer to appendix c for the code of the above files .

#### *5.5.1.4 Load Balancing*

In `app.py` also employs a round-robin technique in order to balance the load for the API requests between the various endpoints that are available. This technique basically involves a list of API endpoints which are to be called in a particular order. When the request is sent, the `fetch_patient_data` function goes through the list of the APIs and attempts to use each of them consecutively. This approach helps to achieve load balancing where the workload is appropriately distributed across the offered APIs thus, none of the API is overworked thus enhancing reliability of data.

#### *5.5.1.5 Handling API Failure*

In `app.py` the error handling is to handle the API failure and this is done in the `fetch_patient_data` function. Here's how it is handled:

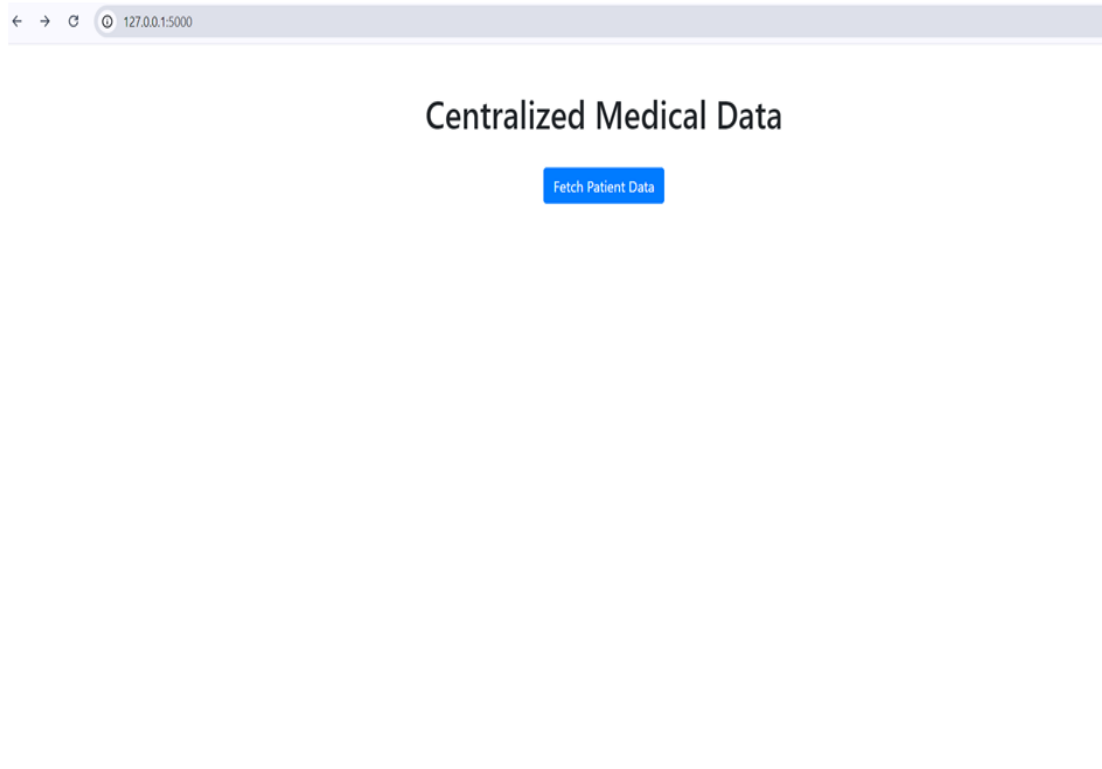
1.Exception Handling: The function employs the use of try-except block to handle exceptions during API calls. Specifically, it uses `requests.RequestException` to deal with any errors that are linked to network issues, wrong responses or any other errors that are associated with the request. In case of a raised exception the function adds a

pre-defined error time (which is 3 seconds in this case) and moves to the next API endpoint in the list.

2. Retry Mechanism: In case when a request to one API endpoint fails, the function does not stop its execution. Rather than that, it waits for 3 seconds before trying to call the next endpoint. This waiting period also reduces problems that may be transient in nature including network hiccups or rate restrictions. This process goes on in a cyclic manner until all the endpoints have been attempted one at a time.

#### *5.5.1.6 Testing User Interface*

To test the UI of the Flask application, the UI of the application has to be tested by manually checking whether it meets the user's expectation and is working as it should. This consists of verifying the existence of the `fetch_patient`. This is done to check that the html page that is being designed has the title of "Centralised Medical Data" and that the "Fetch Patient Data" button is positioned in the centre of the page. After clicking the button, you must verify whether the output presented. The HTML page generated provides exact information of the patient in a neatly organised table format combined with the API information and the time taken to complete the entire process. This manual testing is to check that all the UI elements are shown correctly and the application works as required when the user interacts with the application. The following figures 6, 7 and 8 show the fetching and displaying the patient details from the cloud.



**Figure 6 Dialogue box to fetch patient data**

Patient Details	
Field	Value
Resource Type	Patient
Identifier System	http://hospital.smarthealth.org/patients
Identifier Value	P123456
Active	True
Name	John Doe
Phone	+1-555-123-4567
Email	johndoe@example.com
Gender	male
Birth Date	1985-04-23
Address	123 Main St, Springfield, IL, 62704, USA
Marital Status	Married

**Figure 7 Displaying the patient details**

Birth Date	1985-04-23
Address	123 Main St, Springfield, IL, 62704, USA
Marital Status	Married
Contact Name	Jane Doe
Contact Phone	+1-555-765-4321
Contact Address	456 Oak St, Springfield, IL, 62705, USA
Contact Organization	Springfield General Hospital
Preferred Language	English
General Practitioner	Dr. Emily Smith
Managing Organization	Springfield General Hospital
Link	Jane Doe

**Figure 8 Patient details displayed**

## 5.6 Conclusion

The experiment chapter describes the design and testing of a medical records sharing system on the cloud with the help of AWS Lambda, AWS API Gateway, and Python language. The chapter starts with the API endpoint design across multiple geographical locations and the load balancing of the API requests as well. Some of the key experiments were conducting response time measurements in normal conditions and assessing the system's ability to cope with API failure is described in the following chapter

## 6.Results

### 6.1 Introduction

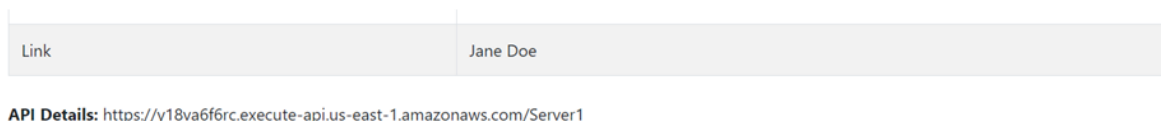
The chapter on Results, Testing and Evaluation gives a detailed account of the effectiveness of the cloud-based medical records sharing system that has been designed using AWS Lambda, AWS API Gateway and Python. This chapter describes the results of several experiments that were performed in an effort to measure the level of success of this system design in relation to load balancing, API commands and response times, and failure handling.

Following are the tests conducted:

- Testing load Balancing
- Testing API Failures
- Testing Latency

#### 6.1.1 Testing Load Balancing

Load balancing functionality testing is done by checking that the number of API requests made when the “Fetch Patient Data” button is clicked is balanced correctly. When the output page is rendered, patient information is displayed on the page alongside the API endpoint which was used to make the request. This output enables the validation of efficient load balancing because the APIs are expected to be called in a serial, round-robin manner. Thus, it is possible to control each API separately to check that all of them are called, which confirms the proper load balancing. This testing establishes that the system delivers requests to the APIs in equal measure as planned hence balanced load distribution. The figure 9 and 10 shows the screenshot of execution of the first and second time.



**Figure 9 Executing 1st Time**

Link	Jane Doe
------	----------

API Details: <https://unbf0byjdc.execute-api.us-east-2.amazonaws.com/Server2>

**Figure 10 Executing 2nd time**

## 6.1.2 Testing with API Failures

To test the handling of API failure the Ohio API was deliberately stopped. This was done by navigating to API settings and modifying the default endpoint to “Inactive” then saving these changes. The API deployment was then updated to reflect this deactivation thus adopting the new state. The figure 11 shows the screen shot of API failures testing, followed by which the Figure 12 and 13 shows the execution of the first and second time.

Regional APIs are deployed in the current AWS Region. Edge-optimized APIs route requests to the nearest CloudFront Point of Presence. Private APIs are only accessible from VPCs.

Regional ▼

**Default endpoint**  
To allow clients to access your API only by using a custom domain name, deactivate the default endpoint. This is an API-level setting and affects all stages of your API. After you change this setting, deploy your API for the update to take effect.

☐ Active

☒ Inactive

**API key source**  
Select the source of API keys from incoming requests. APIs can receive keys from either the x-api-key header or a Lambda authorizer.[Learn more](#)

**Figure 11 Testing with API failures**

The Ohio API was unavailable at this time but the application remained available and operational. It was able to pull data from the Virginia API at least. This is something that shows the application can manage an API outage and seamlessly move to another available endpoint so as to continue providing service.

Link	Jane Doe
------	----------

API Details: <https://y18va6f6rc.execute-api.us-east-1.amazonaws.com/Server1>

**Figure 12 Executing 1st Time**

Link	Jane Doe
------	----------

API Details: <https://y18va6f6rc.execute-api.us-east-1.amazonaws.com/Server1>

**Figure 13 Executing 2nd time**

### 6.1.3 Testing Latency

Latency testing was conducted to evaluate the response times of two API endpoints: In the first server, Virginia and in the second server, Ohio. From the results obtained it can be seen that the response times of both the servers are almost the same, for instance the response time for Virginia was an average of 0. Further, the time taken by the total number of vehicles was estimated to be 82 seconds and Ohio averaged approximately 0. 83 seconds. The tests show a low level of difference in latency between the two servers which shows that both the locations are equal in as far as the time taken to respond to requests is concerned. Table 4 shows the results of both the endpoints of Virginia and Ohio.

**Table 4 Response times of two API endpoints Virginia and Ohio**

Test	Virginia (Server 1)	Ohio (Server 2)
Test 1	0.84	0.81
Test 2	0.82	0.83
Test 3	0.77	0.85
Test 4	0.81	0.86
Test 5	0.82	0.82
Test 6	0.9	0.89
Test 7	0.89	0.82



Test 8	0.87	0.8
Test 9	0.86	0.85
Test 10	0.86	0.84

All in all, the latency data show that both the Virginia and Ohio servers are operating effectively with comparable performances. The results of the tests are also similar across the different scenarios with both servers' response time averaging within the same range. This means that the load balancing should be in a position to distribute the requests in a way that will not lead to a negative impact on the performance.

#### 6.1.4 Testing Latency After API Failure (ohio server 2)

Following the API failure of Ohio (Server 2), a latency test was conducted to measure the effect that this had on the other working server, Virginia (Server 1). The latency results for Virginia are as follows: minimum latency is 4.1 and 4.82 ms in the ten trials that were conducted. It was also seen that the average latency was a little higher than the baseline level which means that the server took slightly longer time to respond due to the load shifting from the failed API. Table 5 shows the server details of Ohio and their failures.

**Table 5 API failure of Ohio Server**

Test	Virginia (Server 1) in milliseconds
Test 1	4.72
Test 2	4.1
Test 3	4.2
Test 4	4.7
Test 5	4.66
Test 6	4.6
Test 7	4.6
Test 8	4.82
Test 9	4.1
Test 10	4.1

#### 6.1.5 Testing Latency After API Failure (Virginia - server 1)

However, when Virginia (Server 1) was the server under test after experiencing the Ohio API failure, the latency values were between 4. 3 to 4. 92 milliseconds. These values are similar to the latency measurements when both servers were active with a slight raise in the latency. This implies that although there was a clear trend of increase in latency, it did not escalate beyond the normal limits thus establishing that the system was capable of coping with the increased demand without a considerable decline in efficiency. Table 6 shows the Testing Latency after API failure in Virginia server 1.

**Table 6 Latency of API failure of Virginia – Server 1**

Test	Ohio (Server 1) in milliseconds
Test 1	4.92
Test 2	4.67
Test 3	4.4
Test 4	4.7
Test 5	4.6
Test 6	4.3
Test 7	4.6
Test 8	4.9
Test 9	4.8
Test 10	4.4

In essence, the load balancing mechanism proved useful in coping with the high traffic load on the Virginia server in the absence of the Ohio server. Although the latency had increased to some extent the overall performance of the system was still good. This testing is beneficial because it proves that the load balancing strategy can switch between different servers during the fall of one of them and still guarantee effective service to the users.

## 6.2 Analysis of Results

The study aimed at assessing the efficiency of the system developed utilising the AWS Lambda and API Gateway for data retrieval of patients. The system used two Lambda functions in North Virginia and Ohio, and to balance the API request load the round-robin load balancing was used. This load balancing was effective in ensuring that no endpoint was overloaded while at the same time maintaining stability through testing. The integration of AWS API Gateway helped in making the Lambda functions implementable as RESTful webs services so that HTTP requests and responses could be easily made and received.

In the testing phase, the system's performance was very effective in managing API failure. When the Ohio API was deliberately shut down, the system was able to perform well by forwarding the requests to the Virginia API. All these were managed well through failure handling since the system was able to cope with the outage with minimal disturbance of service. This is an indication of the system's flexibility; the system is capable of continuing with the data retrieval operations even if some endpoints become unavailable or fail.

Latency testing results revealed that the two endpoints had nearly the same performance with the only noticeable difference being the response time of Virginia and Ohio servers when both were active. Following the Ohio API failure, the latency of the remaining Virginia server had a slight rise but still stayed optimal, therefore, supporting the load balancing mechanism. Therefore, the study shows that the implemented system is capable of managing API requests and handling failure if any, while maintaining optimal performance to ensure that it is a suitable system for managing patient data.

## 6.3 Discussion of results

The load balancing technique used in the application which is round-robin was able to spread the API requests across several endpoints. It was observed that the APIs were being called one after the other thus providing equal distribution of requests. This approach was useful in Avoiding overloading any given endpoint hence improving on the probability of data retrieval and maintaining uniformity in the performance of the API endpoints.

When the Ohio API endpoint was deliberately stopped, the application worked smoothly as the requests were forwarded to the Virginia API. This goes to show the reliability of the system's failure management, which was able to ensure that the service remains uninterrupted even in the case of one of the API endpoints being out of service. The utilisation of an operational endpoint is possible in the system, thus making the system reliable and robust in its operation.

The latency test showed that response time for the Virginia and Ohio servers was relatively similar in the normal scenario with slight fluctuations between the two. This implies that both the servers offered efficient performance with similar response time; hence, the load balancing procedure would not introduce a high amount of latency.

When Ohio (Server 2) became unresponsive, the latency for the Virginia (Server 1) endpoint saw a slight rise from 4.1 to 4.82 milliseconds. Likewise, when Virginia was tested after its failure, latency results were between 4.3 to 4.92 milliseconds. Thus, although there was some deterioration in the latency owing to the augmented load on the operational server, the response times did not cross the tolerance limits, indicating that the system had coped with the augmenting traffic load efficiently.

The User Interface (UI) to get and present patient records was effective in the testing phase. The `fetch_patient`. This is due to the fact that the output can be seen as an HTML page and the output. This html page was loaded and the information of the patient was presented appropriately and in a well-organised manner on the page. Load balancing and API failure handling was tested through UI, to ensure that users would not encounter any issues during the interaction with the system and would receive proper feedback on the data and the system performance.

## 6.4 Evaluation Against Requirements / Research Goals

### 6.4.1 Evaluation Against Research Aim

The researcher was able to design a cloud based medical record sharing system that was aimed at improving data transfer within the UK healthcare industry. The system used HL7 standards for the structured medical data, and the two completely separate API endpoints

were developed for data acquisition. This was done while testing the system and it was discovered that the system could process API requests with little to no latency variance between the endpoints. The implementation of HL7 in JSON format made patient data to be in a standard format and easily retrievable from one application to another in the cloud-based system.

A load balancing method was used to make sure that the API requests were distributed in as many endpoints as possible in order to avoid overloading one server. Moreover, the error handling model was proved to be reliable since it still worked while one of the API endpoints was non-operational. Although the latency grew slightly when switching to the backup endpoint the system well controlled the load and did not cause a major impact on the performance. In general, the study supports the hypothesis that load balancing and error handling reduce latency effects and increase the robustness of medical data sharing systems.

#### 6.4.2 Evaluation Against Research Objectives

**To design and implement a cloud-based infrastructure capable of securely storing and managing medical records in HL7 format.**

This goal was met through establishment of a cloud infrastructure that is capable of effectively managing medical records that are in HL7 format. It included configuration of cloud storage environments and ensuring compliance with the security best practices when handling data. It was developed in order to prevent access of the patient's private health information, which is in compliance with the current standard on data protection and healthcare privacy.

**To integrate HL7 as a standardised format for a sample medical data in JSON format for easy access through cloud API endpoints.**

The implementation of HL7 standards required the translation of the existing medical data in the form of JSON format, which is easily utilised in cloud APIs. This step was taken to make sure that medical data could be easily transferred as well as retrieved through API

endpoints. Through the exploitation of JSON, the system made it easy to work with the cloud services, that is, to manage and obtain data.

**To develop two API endpoints in different geographical locations and to test the response time difference in fetching medical data in a standardised format.**

Two API endpoints were developed in two different geographical regions with a view of comparing the response time. This objective was to assess the effectiveness of data access in different locations and to determine how geography impacts on latency. Based on the outcomes of this test, recommendations were made on the best ways of enhancing efficiency of data retrieval and the performance of the cloud-based system.

**To enable load balancing techniques to handle API endpoints that effectively call APIs one by one.**

The following measures were employed to ensure that the API traffic is well managed to avoid overloading some nodes; load balancing techniques. This was done in order to avoid overloading one endpoint and thus the API requests were split across the available endpoints. This approach enhanced system dependability and capacity as it did not congest one API endpoint with multiple requests and hence, made it easy to manage multiple requests.

**To apply a model that can handle API failures more effectively and test the latency difference before and after API failures.**

An ER model was also developed for the management of failure of APIs and its effects in the system. This model was evaluated in order to determine its ability to sustain system performance during and following failures. To determine the performance of the system in as much as it can recover and still provide services even when there are API failures the research assessed the latency before and after the API failure.

**To create a user-friendly front-end interface using Python for easy access and interaction with the system.**

In this work, Python was used to design a user-friendly front-end interface for easy interaction with the cloud-based system. This interface provided the users with the ability to access and manipulate medical records with ease thus improving the user experience. The front-end design was kept simple and practical in order to enable the users to be able to easily use the different parts of the system.

### 6.4.3 Evaluation Against Research Question

Yes, the experimental results reveal that there is a direct effect on latency variation when implementing load balancing techniques and when APIs fail.

From the results shown above, it can be seen that load balancing techniques have been useful in minimising latency variance. With a round-robin load balancing algorithm, the system balanced the requests between the Virginia and Ohio endpoints. This way, the response time remained almost the same for both the servers and none of the servers was overloaded with requests.

When the Ohio server crumbled, the system was redirected to the Virginia server. This led to latency increasing slightly with the response times going up by 4. 1 to 4. 82 milliseconds. It can also be seen that the Virginia server had a somewhat higher latency than the other server when it was the only active server. However, the latency did not exceed the norm, proving that the system coped with increased levels of load and ensured functionality even in conditions of failures.

### 6.4.4 Technical Challenges

This project presents the technical challenges faced during the development of the medical records sharing system. One of the challenges encountered was the inability of the API to call data from the Lambda function at the beginning. This issue was resolved through checking on the Lambda function settings and permissions, and making sure that it has the ability to read from the sources. The challenge brought out the need to have proper function setup and debugging when using server less architecture.

Another problem was related to the usage of EC2 instances initially, they were employed, but then Lambda functions were introduced because of the possibility to manage them better



and since they are more scalable. Integration of the Lambda functions facilitated the deployment process and enhanced the systems' efficiency. Furthermore, although the integration of load balancing was easy, managing the API failure was quite challenging. The first problem faced by the system was errors in API failures and they were solved by adding if-else conditions to handle the failover to ensure that the system remains stable. These experiences pointed towards the importance of effective error control and flexible load balancing in cloud systems.

## 6.5 Summary

The last chapter of the project is the Results, which focuses on the evaluation of the cloud-based medical records system in terms of load balancing as well as the handling of API failures. This segment also shows how the round-robin load balancing mechanism ensured that each endpoint received an equal share of requests and did not overload any endpoint while at the same time maintaining services in the event of an API endpoint failure. Latency testing demonstrated that there was little fluctuation in delays during normal operations while moderate fluctuations were noted during API failures but the system was able to handle the increased load. In conclusion, it can be stated that the evaluation proves the system is effective and reliable in the context of data retrieval.

## 7. Conclusions

### 7.1 Summary of conclusions

The purpose of this research was to design a medical records sharing system on a cloud platform in the UK healthcare sector. These included handling API endpoints in as much as it tried to address the issue of load balancing and effective error management. To this end, the following tasks were identified: the creation of a secure cloud infrastructure; the integration of HL7 standards; the development of geographically dispersed APIs; the testing of load balancing and failure recovery solutions.

The experiment was to design a cloud-based platform to securely store medical records in HL7 format and to transform it into JSON for API consumption. To test the response time and to check how well load balancing works, two API endpoints were created in two different geographical locations, Virginia and Ohio. To handle the API requests, the round-robin load balancing technique was used and to check the performance of the system under API failure conditions, endpoint shutdown was simulated and the effect on latency was noted.

From the research, it was realised that the round-robin load balancing method was effective in the equal distribution of API requests so that no single endpoint was overloaded and performance was constant. The response times for the Virginia and Ohio endpoints were alike in normal case which indicates that load balancing has not impacted the latency in a noticeable way. In the simulated failure of APIs, the system was able to redirect the requests to the working endpoint and the latency was slightly high but reasonable.

Therefore, the study showed that proper load balancing and failure management strategies are necessary for ensuring dependable and high-performance of a cloud-based medical records sharing system. The findings established that the system was capable of accommodating growth and functionality in spite of losses at one of the endpoints. The user interface was also satisfactory as it made it easy to access the patients' information. Altogether, the study has the objectives set and has confirmed the efficiency of the applied measures aimed at increasing the reliability of the system.

## 7.2 Recommendations

To make the cloud-based medical records sharing system even better, it would be advisable to consider some of the more sophisticated load balancing algorithms like the Least Connections or the IP Hash to help in balanced distribution of the request and to avoid latency. Such methods may perform more efficiently than round-robin techniques under various load conditions. Also, it is possible to employ and check the horizontal scalability approaches and failover possibilities for different regions or cloud providers, which can enhance the capability of the system to handle increased traffic and guarantee high availability of the solution, thus providing a more robust and scalable option.

In addition, there are various ways of enhancing the API performance including caching, query optimization, and data compression which could greatly enhance the response times and general efficiency. It is also important for the health data that a thorough security assessment should be performed; performing security assessments such as vulnerability testing and checking the compliance with healthcare data protection laws will guarantee the system's resistance to threats and breaches.

## 7.3 Future work

The future work for further improvement of the medical records sharing system is to extend the system to accommodate multiple locations. Due to the distribution of the API endpoints in different geographical locations the system is capable of providing faster responses and backup, so that latencies and outages will not be critical issues and will not interrupt the service in any part of the world. It also means that the geographic distribution of data will be better managed and the solution will be more suitable for the exchange of global health data.

Another important consideration is the Management of Large Database Medical Records where there will be the need to enhance the data storage and Database Management Systems to cater for the large volumes of data. Further, EC2 instances can be used for cloud computing for scalability to increase the processing capabilities and flexibility. Lastly, integrating AWS Cloud Watch for monitoring and performance metrics gives a very important

factor in the system's performance, which helps in determining the areas that need to be worked on further in the research and development of the system.

## References

- Abbas, A., Alroobaea, R., Krichen, M., Rubaiee, S., Vimal, S. and Almansour, F.M., 2024. Blockchain-assisted secured data management framework for health information analysis based on Internet of Medical Things. *Personal and ubiquitous computing*, 28(1), pp.59-72.
- Abouali, M., Sharma, K., Ajayi, O. and Saadawi, T., 2021, December. Blockchain framework for secured on-demand patient health records sharing. In *2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)* (pp. 0035-0040). IEEE.
- Benil, T. and Jasper, J.J.C.N., 2020. Cloud based security on outsourcing using blockchain in E-health systems. *Computer Networks*, 178, p.107344.
- Chang, W., Zhang, Q., Fu, C., Liu, W., Zhang, G. and Lu, J., 2021. A cross-domain recommender system through information transfer for medical diagnosis. *Decision Support Systems*, 143, p.113489.
- Chen, R.J., Lu, M.Y., Chen, T.Y., Williamson, D.F. and Mahmood, F., 2021. Synthetic data in machine learning for medicine and healthcare. *Nature Biomedical Engineering*, 5(6), pp.493-497.
- Christo, M.S., Jesi, V.E., Priyadarsini, U., Anbarasu, V., Venugopal, H. and Karuppiah, M., 2021. Ensuring improved security in medical data using ecc and blockchain technology with edge devices. *Security and Communication Networks*, 2021(1), p.6966206.
- Mahajan, H.B., 2022. Emergence of healthcare 4.0 and blockchain into secure cloud-based electronic health records systems: solutions, challenges, and future roadmap. *Wireless Personal Communications*, 126(3), pp.2425-2446.
- Makhdoom, I., Zhou, I., Abolhasan, M., Lipman, J. and Ni, W., 2020. PrivySharing: A blockchain-based framework for privacy-preserving and secure data sharing in smart cities. *Computers & Security*, 88, p.101653.

Maxi, K. and Morocho, V., 2021, December. Integrating medical information software using health level seven and FHIR: a case study. In the International Conference on Smart Technologies, Systems and Applications (pp. 84-98). Cham: Springer International Publishing.

Morid, M.A., Borjali, A. and Del Fiol, G., 2021. A scoping review of transfer learning research on medical image analysis using ImageNet. *Computers in biology and medicine*, 128, p.104115.

Pradhan, B., Bhattacharyya, S. and Pal, K., 2021. IoT- based applications in healthcare devices. *Journal of healthcare engineering*, 2021(1), p.6632599.

Romero, M., Interian, Y., Solberg, T. and Valdes, G., 2020. Targeted transfer learning to improve performance in small medical physics datasets. *Medical physics*, 47(12), pp.6246-6256.

Salim, M.M. and Park, J.H., 2022. Federated learning-based secure electronic health record sharing scheme in medical informatics. *IEEE Journal of Biomedical and Health Informatics*, 27(2), pp.617-624.

Scheibner, J., Raisaro, J.L., Troncoso-Pastoriza, J.R., Ienca, M., Fellay, J., Vayena, E. and Hubaux, J.P., 2021. Revolutionising medical data sharing using advanced privacy-enhancing technologies: technical, legal, and ethical synthesis. *Journal of medical Internet research*, 23(2), p.e25120.

Setyawan, R., Hidayanto, A.N., Sensuse, D.I., Suryono, R.R. and Abilowo, K., 2021, November. Data integration and interoperability problems of HL7 FHIR implementation and potential solutions: a systematic literature review. In 2021 5th International Conference on Informatics and Computational Sciences (ICICoS) (pp. 293-298). IEEE.

Shuaib, K., Abdella, J., Sallabi, F. and Serhani, M.A., 2022. Secure decentralised electronic health records sharing system based on blockchains. *Journal of King Saud University-Computer and Information Sciences*, 34(8), pp.5045-5058.

- Sivan, R. and Zukarnain, Z.A., 2021. Security and privacy in cloud-based e-health systems. *Symmetry*, 13(5), p.742.
- Sultana, M., Hossain, A., Laila, F., Taher, K.A. and Islam, M.N., 2020. Towards developing a secure medical image sharing system based on zero trust principles and blockchain technology. *BMC Medical Informatics and Decision Making*, 20, pp.1-10.
- Sun, J., Ren, L., Wang, S. and Yao, X., 2020. A blockchain-based framework for electronic medical records sharing with fine-grained access control. *Plos one*, 15(10), p.e0239946.
- Thabit, F., Alhomdy, S., Al-Ahdal, A.H. and Jagtap, S., 2021. A new lightweight cryptographic algorithm for enhancing data security in cloud computing. *Global Transitions Proceedings*, 2(1), pp.91-99.
- Trigui, A., Ali, M., Hached, S., David, J.P., Ammari, A.C., Savaria, Y. and Sawan, M., 2020. Generic wireless power transfer and data communication system based on a novel modulation technique. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(11), pp.3978-3990.
- Vanmathi, C., Mangayarkarasi, R., Hari Haran, V. and Karthikeyan, S., 2021. A secure data transfer in a cloud environment using double-layer security for internet of medical things. In *Soft Computing for Problem Solving: Proceedings of SocProS 2020, Volume 2* (pp. 211-229). Springer Singapore.
- Yaqoob, I., Salah, K., Jayaraman, R. and Al-Hammadi, Y., 2022. Blockchain for healthcare data management: opportunities, challenges, and future recommendations. *Neural Computing and Applications*, pp.1-16.
- Zhou, L., Fu, A., Mu, Y., Wang, H., Yu, S. and Sun, Y., 2021. Multicopy provable data possession scheme supporting data dynamics for cloud-based electronic medical record system. *Information Sciences*, 545, pp.254-276.

# Appendix A: Ethics certificate



## Certificate of Ethics Review

Project title: Centralized Sharing Of Medical Records Using Cloud

Name:	Hareesh Kakarla	User ID:	UP2228499	Application date:	01/06/2024 18:44:47	ER Number:	TETHIC-2024-108861
-------	-----------------	----------	-----------	-------------------	---------------------	------------	--------------------

You must download your referral certificate, print a copy and keep it as a record of this review.

The FEC representative(s) for the **School of Computing** is/are [Elisavet Andrikopoulou, Kirsten Smith](#)

It is your responsibility to follow the University Code of Practice on Ethical Standards and any Department/School or professional guidelines in the conduct of your study including relevant guidelines regarding health and safety of researchers including the following:

- [University Policy](#)
- [Safety on Geological Fieldwork](#)

It is also your responsibility to follow University guidance on Data Protection Policy:

- [General guidance for all data protection issues](#)
- [University Data Protection Policy](#)

Which school/department do you belong to?: **School of Computing**

What is your primary role at the University?: **Postgraduate Student**

What is the name of the member of staff who is responsible for supervising your project?: **Vasileios Adamos**

Is the study likely to involve human subjects (observation) or participants?: No

Will financial inducements (other than reasonable expenses and compensation for time) be offered to participants?: No

Are there risks of significant damage to physical and/or ecological environmental features?: No

Are there risks of significant damage to features of historical or cultural heritage (e.g. impacts of study techniques, taking of samples)?: No

Does the project involve animals in any way?: No

Could the research outputs potentially be harmful to third parties?: No

Could your research/artefact be adapted and be misused?: No

Will your project or project deliverables be relevant to defence, the military, police or other security organisations and/or in addition, could it be used by others to threaten UK security?: No

Please read and confirm that you agree with the following statements: I confirm that I have considered the implications for data collection and use, taking into consideration legal requirements (UK GDPR, Data Protection Act 2018 etc.), I confirm that I have considered the impact of this work and and taken any reasonable action to mitigate potential misuse of the project outputs, I confirm that I will act ethically and honestly throughout this project

### Supervisor Review

As supervisor, I will ensure that this work will be conducted in an ethical manner in line with the University Ethics Policy.

Supervisor comments:

Supervisor's Digital Signature: **vasileios.adamos@port.ac.uk** Date: **02/06/2024**



## Appendix B: Project Specification



# **School of Computing Postgraduate Programme**

**MSc in Computer Network Administration  
and Management**

## **Project Specification**

**Hareesh Kakarla**

# Project Specification

## 1. Basic details

Student Name:	Hareesh Kakarla
Project Title:	Centralised Sharing of Medical Records Using Cloud Platform
Course and year:	Msc Computer Network Administration and Management and 2023-2024
Client organisation:	University Of Portsmouth
Client contact name:	Vasileios Adamos
Project supervisor:	Vasileios Adamos

## 2. Outline of the project environment

The relevant aspect of current healthcare situation is the decline of centralization of medical records in the hospitals and healthcare organisations. This leads to lower quality and ineffective system of healthcare data sharing among the facilities. UK here misses centralised system hence the situation where the patient information is spread across the different private servers and software platforms is develop that affect effective access to relevant information. Moreover, there is a genuine problem of patient care coordination turning into a complex process which may expose data security and privacy (Allam, et al, 2020). It, therefore, necessary to have one generalized medical record connecting system adopting cloud dedicated for such processes. The cloud service platforms offer scalability, accessibility, and secure data transfer hence it will be convenient to have such a system as it will streamline data transfer processes and provide seamless access to patient records across different healthcare entities. The centralized platform which governs all data using

---

structured formats like HL7 further strengthens interoperability among healthcare channels and eases the way of communication between medical providers (Sheller, et al, 2020). The proposed project addresses the challenges with the purpose to reform at the medical data management in the UK healthcare sector, namely: for more envisaged patients care outcomes and more efficient health care provision (delivery).

### **3. The problem to be solved**

In healthcare system in the UK, the absence of a unified system for exchanging medical records brings about major difficulties which hamper the delivery of efficient patient care and communication between the healthcare providers. Actually, there are many different hospital IT systems in use with a variety of cloud servers and data repositories, and their decentralized approach to data hampers the effective transfer of essential patient data when and where it is required (Makhdoom, et al, 2020). Such dispersion not only entails instrumental factors such as longer time of diagnosis and treatment but adds up to the danger of medical mistakes and the possibilities of redundant healthcare services. In addition, the lack of a single data format represents another hurdle. Not all systems can understand the information effectively which is provided by the rival methods. Therefore, patients may be suffering the form of excessive discontinuity of care, unnecessary diagnostic tests and poor treatment results. Besides that, healthcare providers encounter gaps in accessing the necessary details of a patient which hinders the decision-making process and ultimately the patient safety (Chen, et al, 2021). This problem can be addressed by changing the healthcare professionals' current practices through different specific methods that can ensure the secure and seamless data exchange, standardize identified data formats, and give healthcare providers timely access to comprehensive patient records, therefore significantly increasing the quality and efficiency of healthcare across the United Kingdom.

### **Aim**

To develop a centralized medical records sharing system using a cloud platform in the UK healthcare sector, facilitating seamless data transfer and access across different hospitals and healthcare organisations.

---

## Research question

How can Health Level Seven(HL7) be effectively implemented in a cloud-based environment for medical data standardisation?

## Objectives

- To design and implement a cloud-based infrastructure capable of securely storing and managing medical records.
- To integrate HL7 as a standardized format for medical data exchange within the system.
- To develop robust database management mechanisms ensuring data integrity, security, and compliance with regulatory standards.
- To enable efficient data upload and retrieval functionalities for healthcare organizations using the system.
- To create a user-friendly front-end interface using Python for easy access and interaction with the system.

## 4. Breakdown of tasks

### Approach

- Design a cloud system of high scalability and security for a centralized database of medical records, which guarantees its accessibility and reliability for medical professionals.
  - Introduce HL7 as the emerging industry protocol for sharing medical data between multiple healthcare providers by this way data exchange is easier and more accurate.
  - Use reliable data management practice to keep data non-manipulatable, secure, and compliance with data protection laws and regulations with sensitive patient information protected from unauthorized access or data leaks.
  - Implement protocols for healthcare organizations to upload and pull up their medical data on the internet quickly and securely reducing the delays which commonly occur in accessing patients' critical information and at the same time boosting the operational efficiency.
-

- Develop using Python an accessible and user-friendly front-end interface which will allow the healthcare professionals to have a trouble-free access to medical records and communication channel to the system's central node.

## Background Research

(Abouali et al,2021) work aspires to a block chain framework for storage and sharing legitimate patient health information and granting the whole control power and access to patients. Using this novel approach, we implement Ethereum blockchain smart contracts, the inter-planet file system (IPFS) as an off-chain storage system, and the Cypher protocol as a key management and blockchain-based proxy re-encryption to practically advance a decentralised, secured, on demand patient health record sharing system. This result indicates that our plan is more secure than other similar schemes. Non-authorized healthcare providers or patients would be able to access the PHRs. Lastly, the data encrypted by the patient can only be accessed and read by entities which are verified and adopted by the patient only.

Here, (Wang et al, 2021) paper outlines MedShare a decentralised framework for EHRs that is safe for sharing and can be utilized by parties involved. The technological approach we have employed is the deployment of smart contracts to build a trusted network derived from the blockchain technology that would encrypt EHR of the healthcare centers for sharing. Since the fine-grained access control is one of the fundamental tiers required in practical EHR sharing service, a family member of ABE, constant size, whose access policy is the part of the search result on the blockchain is found here. Besides, a cunning algorithm is put forward which provides a feature of boolean multi-keyword search to permissible MedShare participants for the cases where EHR is encrypted. Once the test had been done the outcome showed that MedShare was indeed very significant for sharing of EHRs.

The articles by Abouali and Wang (2021) both present the decentralized structure of blockchain for the secure and controlled application of electronic health records (EHR) controlling function. The research team of Abouali et al. (2021) boots all the Ethereum's smart contract, IPFS to save the documents off-chain, and Cypher protocol to deliver a non-demand and secured access to the patients' health records. Wang et al. propose managing MedShare with smart contracts for trusted nodes with blockchain that also has advanced access control

---

mechanisms like attribute-based encryption and boolean multi-keyword search. Nevertheless, an examination of scalability, interoperability, and regulatory standardization concerning decentralized systems in a real-world healthcare setting is not an investigative area yet.

## **Skills**

- **Cloud Infrastructure Management:** Mastering the art of designing and putting into action cloud-based infrastructure, e.g. AWS, Azure, or Google Cloud, is the main point to ensure proper scalability, security, and reliability of the storing and managing of medical records.
- **Software Development and Integration:** Ability of being associated with expertise in the integration of healthcare standards like HL7 into software systems; competency in the field of database management that enables data integrity, security, and compliance with regulatory standards.

## **The main deliverables**

The essential outputs from the centralized medical records systems based on cloud platform would be a highly scalable and robust capacity designed to securely store and process data. The system should be integrated into HL7 organisation to exchange data in a standard manner. This infrastructure is designed and executed to make sure it is accessible, reliable and complies with regulatory standards to offer necessary health organizations and to store and obtain patients record.

Moreover, the system shall entail appropriate database management mechanisms to preserve data security and integrity, so medical data will be prevented from falling into the hands of unauthorised persons or becoming compromised in various kinds of breaches.

Moreover, the system is going to be designed in such a way that it provides efficient data upload and retrieval processes so that healthcare organisations can easily transfer and get medical records in minimum time by avoiding delayed access to critical patient information. User-friendly front-end interface will be developed using Python which will favor the process of chains of interactions and navigation for healthcare professionals.

## **5. Project deliverables**

The project deliverables for the centralized medical records system using a cloud platform are as follows:

The low-cost secure cloud-based infrastructure built for individual medical record storage and management where any convenient, efficient, reliable access to providers as well as shared access by several providers is possible. HL7 standards' implementation is necessary for medical data exchange of the organization, providing opportunity to use uniform data formats during interaction of information systems that within other healthcare systems and decreasing the possibility of inaccuracy in data transmission. Applying solid data management systems to provide data integrity, security, and adherence to regulatory standards guarding from unlawful external access. Making sure that private patient information is properly protected. Establishment of optimised procedures of smooth data transfer and retrieve as well as permitting the health care organizations to easily and securely upload and gain access to patients' info thus resulting to quickened accessing of important information. Development of Python powered front-end user interface where healthcare staff are able to easily use the software and effortlessly interact with their medical records, thereby making it convenient and accessible. Drawing up the instructive documents and training bulletins which will help physicians to learn to use the system right and ease the implementation of the system in the healthcare organization and make it wide spread.

---

## **The overall deliverables include:**

- Cloud-based storage and management system for health records.
- Incorporation of HL7 standards for the purpose of medical data exchange.
- Efficient database management system of the highest level with data integrity, security, and compliance in mind.
- Simplified data load and retrieve functionality.
- Developed user-friendly front-end interface based on Python.
- Detailed instruction guide and training facilities.

## **6. Requirements**

The prerequisites for functioning of the centralised medical records system include development of robust cloud-based infrastructure that is imminently scalable and secure for storing and managing healthcare information. A system that has integrated HL7 standard

Into its foundation will be able to guarantee efficient and standardized data exchange, while at the same time it must have in place robust database management techniques to provide data integrity, security, and comply with the regulations. Fast upload of data and retrieve functions are the basis of the application which are facilitated by a user-friendly frontend developed with Python. Detailed documentation and training materials, intensive testing and a roll-out plan are the ingredients that can bring about successful implementation and the acceptance, of the healthcare organisation, backed by a constant technical support and maintenance framework which helps the system to remain reliable.

## **7. Legal, ethical, professional, social issues**

When constructing a centralized medical system, we need to find answers to several legal, ethical, professional, or social issues. On the legal side of data privacy, staying in compliance with the regulations such as the GDPR of the EU and the Data Protection Act of the UK is an important step towards protection of patient privacy and confidentiality. Morality wise, protecting the patient's autonomy and ensuring that they give their consent to the collection, storage, and sharing of their medical data must be taken care of along with the transparency and responsibility in the handling of such data. Professionally, health professionals must comply with ethical guidelines and the code of practice for accessing

---



and employing patient's information aiming to respect this trust and confidentiality. With regards to the social dimension, close examination of the healthcare disparity and digital literacy accessibility is imperative for a fair access to the centralised system. The system implications and changing dynamics of the healthcare delivery and patient-doctor relationships should not be ignored in the process of digitalization of a healthcare system. Striking a balance among these principles is the main focus of a system that is aimed at satisfying patients' needs, nourishing individual rights, and promoting professional and general values.

## **8. Facilities and resources**

- Computer/Laptop
- Internet Access
- Software
- Cloud Services
- Library Access
- Training Resources
- Internet Reliability
- Library Resources
- Training Resources

## 9. Project Plan

Centralized Sharing of Medical Records Using Cloud Platform	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12
Literature survey												
Framing title and aim												
Framing methods, RQ and objectives												
Preparing project plan												
Dataset collection												
Dataset preprocessing												
Data visualization												
Label encoding												
Implementing												
Cloud-Based Infrastructure												
HL7 Integration												
Database Management System												
Data Upload and Retrieval Functionalities												
User-Friendly Front-End Interface												
Training Materials												
Evaluation of results												
Conclusions drawn												
Final results												
Findings												
Future enhancement												
Documentation												

## 10. Supervisor Meetings

(Very important) What general plan has been agreed with your supervisor regarding supervision meetings over the period of the project? A detailed schedule of meetings is not needed here, but you should have agreed on the frequency of meetings and the mode of meeting (face-to-face where possible, video conferencing, telephone, email, etc).


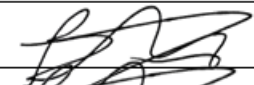

How will your project plan accommodate any extended absences of your supervisor (annual leave, etc)? Every Wednesday at 12:00 there will be a zoom project meeting for my supervisor's project students which will be the regular meeting and any urgent requests will be dealt with via email or emergency zoom meetings to be arranged.

## 11. Project Mode

If there are two possibilities for your project mode, after negotiation, please record your planned duration and submission date. It is also helpful to record your initial registration mode (i.e. Are you a fulltime or a part time student). Remember, the exact dates will be announced through Moodle—to represent a generic guideline.

	Please delete as appropriate	
Registration mode	Full Time	
Project mode	Full Time	
Planned submission deadline	16/09/2024	

## 12. Signatures

	Signature:	Date:
Student		29/05/2024
Client		30/05/2024
Project supervisor		30/05/2024

---

## Appendix C: Project Code

1.fetch\_patient.html. This code will provide the UI page for the application.

```
templates > <> fetch_patient.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Centralized Medical Data</title>
7      <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
8  </head>
9  <body>
10     <div class="container mt-5 text-center">
11         <h1>Centralized Medical Data</h1>
12         <a href="{{ url_for('fetch_patient') }}" class="btn btn-primary mt-4">Fetch Patient Data</a>
13     </div>
14 </body>
15 </html>
16
17
```

2.app.py: This is the code for the functionality of the application.

```
app.py > ...
1  from flask import Flask, render_template, jsonify
2  import requests
3  import time
4  import json
5
6  app = Flask(__name__)
7
8  # API Endpoints
9  APIS = [
10     'https://y18va6f6rc.execute-api.us-east-1.amazonaws.com/Server1',
11     'https://unbf0byjdc.execute-api.us-east-2.amazonaws.com/Server2'
12 ]
13
14 # Initialize the API index
15 api_index = 0
16
17 def fetch_patient_data():
18     global api_index
19     error_time = 0
20     # Try each API endpoint in round-robin fashion
21     for _ in range(len(APIS)):
22         api = APIS[api_index]
23         try:
24             response = requests.get(api)
25             response.raise_for_status()
26             data = response.json()
27             body = data.get('body')
```

```

body = data.get('body')
if body:
    patient_data = json.loads(body) # Use json.loads to parse JSON string safely
    # Update the index to the next API for subsequent calls
    api_index = (api_index + 1) % len(APIs)
    return patient_data, api, error_time # Return patient data, API URL, and time taken
else:
    patient_data = {}
    return patient_data, api, error_time
except requests.RequestException:
    error_time += 3 # Accumulate error time in seconds
    # Move to the next API in the list
    api_index = (api_index + 1) % len(APIs)
except Exception as e:
    print(f"Error processing API response: {e}")
    # Update the index to the next API even if an exception occurs
    api_index = (api_index + 1) % len(APIs)
    return None, None, error_time

# Return None if all APIs fail
return None, None, error_time

```

```

@app.route('/')
def index():
    return render_template('fetch_patient.html')

@app.route('/fetch')
def fetch_patient():
    start_time = time.time()
    data, api_used, error_time = fetch_patient_data()
    total_time = round(time.time() - start_time + error_time, 2) # Ensure total_time is a float

    if data:
        return render_template('output.html', patient_data=data, api_details=api_used, total_time=total_time)
    else:
        return render_template('output.html', error_message="Failed to fetch data from all endpoints", total_time=total_time)

if __name__ == '__main__':
    app.run(debug=True)

```

### 3.Output.html: This code provides the output page for the application.

```
ates > <> output.html > html > body > div.container.mt-5 > table.table.table-bordered.table-striped > tbody > tr > td
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Patient Data Output</title>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <style>
    .table td, .table th {
      vertical-align: middle;
    }
    .table img {
      max-width: 150px;
      height: auto;
    }
  </style>
</head>
<body>
  <div class="container mt-5">
    <h1 class="text-center mb-4">Patient Details</h1>

    {% if error_message %}
      <div class="alert alert-danger" role="alert">
        {{ error_message }}
      </div>
    {% else %}
      <table class="table table-bordered table-striped">
        <thead>
          <tr>
```

```

<html lang="en">
<body>
  <div class="container mt-5">
    <table class="table table-bordered table-striped">
      <thead>
        <tr>
          <th>Field</th>
          <th>Value</th>
        </tr>
      </thead>
      <tbody>
        <!-- Displaying Patient Details -->
        <tr><td>Resource Type</td><td>{{ patient_data.resourceType }}</td></tr>
        <tr><td>Identifier System</td><td>{{ patient_data.identifier[0].system if patient_data
        <tr><td>Identifier Value</td><td>{{ patient_data.identifier[0].value if patient_data.i
        <tr><td>Active</td><td>{{ patient_data.active }}</td></tr>
        <tr><td>Name</td><td>{{ patient_data.name[0].given[0] if patient_data.name else 'N/A'
        <tr><td>Phone</td><td>{{ patient_data.telecom[0].value if patient_data.telecom else 'N
        <tr><td>Email</td><td>{{ patient_data.telecom[1].value if patient_data.telecom and pat
        <tr><td>Gender</td><td>{{ patient_data.gender }}</td></tr>
        <tr><td>Birth Date</td><td>{{ patient_data.birthDate }}</td></tr>
        <tr><td>Address</td><td>{{ patient_data.address[0].line[0] if patient_data.address els
        <tr><td>Marital Status</td><td>{{ patient_data.maritalStatus.coding[0].display if pati
        <tr><td>Contact Name</td><td>{{ patient_data.contact[0].name.given[0] if patient_data.
        <tr><td>Contact Phone</td><td>{{ patient_data.contact[0].telecom[0].value if patient_d
        <tr><td>Contact Address</td><td>{{ patient_data.contact[0].address.line[0] if patient_
        <tr><td>Contact Organization</td><td>{{ patient_data.contact[0].organization.display i
        <tr><td>Preferred Language</td><td>{{ patient_data.communication[0].language.coding[0]
        <tr><td>General Practitioner</td><td>{{ patient_data.generalPractitioner[0].display if
      </tbody>
    </table>
  </div>
</body>
</html>

```

Blank Page