TELECAREPLUS: AN EXTENSIVE TELEMEDICINE PLATFORM ENHANCING DOCTOR-PATIENT COMMUNICATION THROUGH TELECONSULTATION

SAMIP DAHAL

A project report submitted in conformity with the requirements
for the degree of Master's of Science in Information Technology

Department of Mathematical and Physical Sciences
Faculty of Graduate Studies
Concordia University of Edmonton

TELECAREPLUS: AN EXTENSIVE TELEMEDICINE PLATFORM ENHANCING DOCTOR-PATIENT COMMUNICATION THROUGH TELECONSULTATION

SAMIP DAHAL

Approved:

Rossitza Marinova, Ph.D.

Supervisor Date

Committee Member Name, Ph.D.

Committee Member Date

Patrick Kamau, Ph.D.

Dean of Graduate Studies Date

Abstract

In the recent years, telemedicine has gained significant attention due to its potential to improve access to healthcare services at reduced costs, minimum limitations and accessibility. This study proposes an application named TelecarePLUS, a comprehensive and extensive telemedicine platform that allows healthcare service providers to deliver the best quality care to patients remotely. Especially after COVID-19, where social distancing has been a norm, TelecarePLUS has the capability to provide patients with easy access to healthcare irrespective of their location and physical limitations. It is a one-stop for healthcare which includes features like secure audio/video/text consultations, patient registration, electronic medical records management, prescription management and patient monitoring. The platform is user-friendly and easily accessible, with a primary focus on protecting the privacy of data and maintaining data integrity and confidentiality. With the rise of various diseases and workforce shortages, TelecarePLUS has the potential to revolutionize the digital healthcare industry by providing healthcare services to people who may not have easy access to such privileges, such as the ones living in remote areas, and those with mobility challenges.

Keywords: telemedicine; TelecarePLUS; electronic medical records; data integrity; confidentiality; digital healthcare.

Acknowledgement

I would like to express my sincere gratitude and appreciation to the following individuals who have contributed significantly, both directly and indirectly, towards the successful completion of this project.

- I am immensely grateful to **Dr. Rossitza Marinova** for her invaluable support and mentorship throughout the graduate program and the development of this project. Her guidance and expertise has been crucial for this project and has also shaped my academic journey in better ways. I am equally grateful to all my professors who helped and taught me during my graduate program.
- I extend my heartfelt thanks to **Ms. Parisa Ghanbari** for her exceptional skills and dedication towards this project. Her responsibility to design the frontend part of this application has brought life to the project. Designing a web application with such creativity and commitment while also learning the technology alongside deserves the highest praise and appreciation.
- I acknowledge the unwavering support and encouragement from my **parents, friends and family** both here and back home. Their belief in me and their support during the challenging moments have been pivotal and a constant source of motivation and strength.

Last but not the least, thank you to each and everyone for being an integral part of this journey.

Contents

List of Tables

List of Figures

Listings

1 Introduction

The term telemedicine refers to the use of telecommunications technology to provide healthcare services to patients who have constraints for physical visits.

The formal definition provided by WIHV [3] is given below.

Definition 1.1. *Any interaction between patients and/or members of their circle of care, occurring remotely, using any forms of communication or information technologies, with the aim of facilitating or maximizing the quality and effectiveness of patient care.*

It comprises of various technologies like electronic medical data transmission, secure audio, video and text consultations and prescription management for health assessment, timely diagnosis, intervention and supervision [1]. With advancements in technology, it is now possible for healthcare providers to connect with patients, especially for individuals living in rural or remote areas with mobility restrictions. The primary goal of telemedicine is to improve a patient's health by permitting two-way real time communication between patients and providers distantly [1]. Some pre-existing platforms have also included the aspects of robotics and virtual reality [2]. It is a rapidly evolving service to provide increased access to high quality healthcare with ease and cost effectiveness, especially after the COVID-19 pandemic.

Despite of ground breaking advancements in the field of healthcare, many individual still face challenges in accessing high quality healthcare services. The problems lie at both provider and receiver ends. Receivers(patients) face the problem of long waiting times, travel hindrances and unsystematic diagnosis and care, whereas providers(doctors, nurses, staff, etc.) face the challenges of high traffic of patients at intervals and ineffective resource management and allocation. Moreover, the rise for remote healthcare services delivery has further been highlighted by the global crisis caused by the COVID-19 pandemic.

Telemedicine has stood up as a promising solution to the challenges imposed, but existing similar applications still face the factors like technical difficulties, limitations to certain medical conditions and appointment types, difficulties around user adaptability and trust, and data privacy and security concerns. Despite of these limitations, telemedicine has revolutionized the way healthcare services are delivered as compared to the elementary measures. Due to such reasons, it is critical to develop a new application that has the potential to address the challenges mentioned above and also improve outcomes. The application not only provides potential benefits, but also proves its effectiveness in improving results.

The term telemedicine is interchangeably used with other terms like digital health, telehealth and telepharmacy, there are several differences between them. Figure 1 from [3] shows the difference between them.

Figure 1: Difference between Digital Health, Telehealth, Telemedicine, Telepharmacy

1.1 Problem Statement

Before the introduction of telemedicine, healthcare services mostly involved in-person interactions between patients and service providers. Patients in need of medical attention had to either physically visit health centres, hospitals or clinics, or spend large amount of money to afford such facilities in the comfort of their house, and consultation involved face-to-face interaction between a doctor and a patient.

However, this traditional way of delivering healthcare services came with numerous challenges for both the parties. Patients had to schedule their appointment with a doctor well in advance which resulted in long waiting times. Travel distance also played as a major challenge, especially for patients with mobility constraints and chronic illness. People belonging to remote and underprivileged areas had no or minimum access to specialized healthcare or expert opinions.

On the other hand, service providers faced challenges related to high patient influx during peak periods. This also resulted in inefficient resource management like appointment scheduling as per the doctor's availability and staff allocation. Manual co-ordination and communication between the various bodies of a service provider also led to potential miscommunication and disorganized care. Moreover, the system of managing medical records was also burdensome and susceptible to errors, risking patient's medical data being handled insecurely.

TelecarePLUS as a platform is a one-stop for healthcare which includes features like secure audio/video/text consultations, patient registration, electronic medical records management, prescription management and patient monitoring. The plat-

form is user friendly and easily accessible, with a primary focus on protecting the privacy of data and maintain data integrity and confidentiality.

1.2 Organization of this report

The project report is organized as follows:

- Chapter 2 presents an overview of the objectives of TelecarePLUS, the research questions associated to achieve those objectives and how those objectives contribute to the larger picture.
- Chapter 3 talks about the literature review and presents the popularity of telemedicine that has grown over the years, especially after the pandemic.
- Chapter 4 depicts about the project design methodologies and the strategies used to achieve the objectives along with the development timeline and technological stack.
- Chapter 5 addresses the code implementation of the project and presents some insights to the code snippets. The snippets are presented in the Appendix A section.
- Chapter 6 describes the conclusion of this project, the goals and objectives achieved and the future endeavours for this project.
- Chapter 7 highlights the contributions of this project and report.

2 Objectives / Research Questions

The main objective of TelecarePLUS is to promote overall user satisfaction by providing easily accessible healthcare services virtually. Keeping such factors in mind, below are the list of research questions for TelecarePLUS:

- Compared to traditional measures, how effective is telemedicine in contributing to improve the overall outcome?
- What are the hindrances that might affect in the adoption of telemedicine by both ends?
- What are the key factors that contribute to the success of telemedicine?
- What are the ethical and legal issues associated with telemedicine, and how can they be addressed?
- What are the optimized technological and infrastructural requirements for a successful telemedicine implementation?
- How can telemedicine be used to improve access to healthcare services, particularly for rural individuals?
- What are the long-term results associated with the adoption of telemedicine, and how can they be ensured for continuous improvement and optimization of the existing model?

3 Literature review (and theoretical framework)

To understand the state of the art of TelecarePLUS and the relationship between healthcare, telemedicine, data and patients, a comprehensive review of the popularity of existing telemedicine scenarios was conducted. The references were also added from relevant articles and pages from reputable sources and other healthcare and telemedicine websites.

According to the article published by Fierce Healthcare, in January 2021 titled "Telehealth adoption rates continue to soar, with 38% of consumers using virtual care in 2020" [4], it has been found that there is a significant increase in telemedicine usage among users during the pandemic. Consumers have welcomed virtual care at a faster pace, where around 40% say that they believe they will continue to use telemedicine services going forward—up from 11% of consumers using it prior to COVID-19. The report also states that globally, telemedicine companies raised $4.2 billion USD in the first half of 2021 according to a global communications and research firm.

An analysis conducted by McKinsey and released in July reveals that Telehealth adoption has stabilized at a significantly higher level compared to the pre-COVID-19 era, with usage ranging from 13% to 17% of visits across all medical specialties. This represents a remarkable 38-fold increase in Telehealth utilization compared to the pre-pandemic period.

Another article by McKinsey & Company [5] in July 2021 states that this steep change in healthcare practices, borne out of demands, was enabled by these factors - increased willingness to use remote healthcare services by patients and providers and changes enabling easy access and reimbursement facilities. During the global crisis of the pandemic, telemedicine acted as a bridge to personal care, and now stands as a strong contender to redefine virtual and hybrid care models, with a goal of improved healthcare access, patient outcomes and affordability with minimum limitations. The research also shows 40 to 60 percent of telemedicine consumers expressed interest in trying out broader virtual care solutions. Moreover, going to the provider side, 58 percent of physicians continue to view digital health services more positively now than they did before COVID-19. In April 2021, 84 percent of physicians had started using virtual consultation into their practices, and 57 percent expressed an interest to continue to provide virtual care in the future. Figure 2 shows how COVID-19 changed the outlook for telehealth.

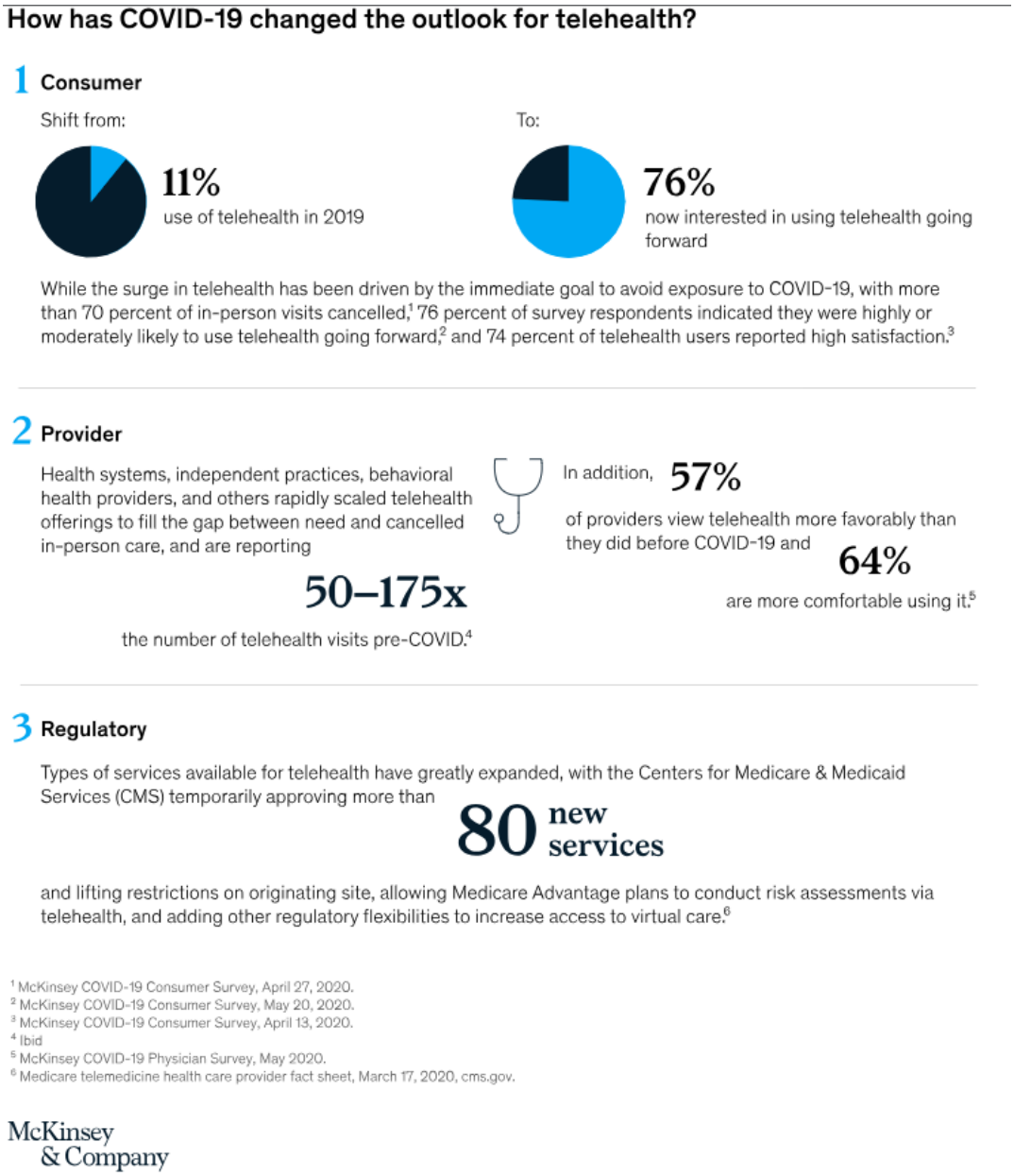

Figure 2: How has COVID-19 changed the outlook for telehealth? [5]

Similar article by HealthTech Magazine in May 2020 [6] titled "6 Reasons Telehealth Is Now More Important Than Ever" states that telemedicine has offered a lifeline during the most difficult times of the pandemic and could have permanent implications on the market. It is likely to become a permanent fixture in healthcare delivery in the near future. A survey by IT vendor Sykes states that more than 60 percent of the patients are willing to try telemedicine due to the pandemic. Kaiser Permanente, Oakland California based, which nearly operates 40 hospitals started conducting more than 90 percent of mental health visits virtually starting the pandemic period, which would have taken years to implement under normal circumstances. Figure 3

shows how telehealth gained popularity during the pandemic for mental healthcare.

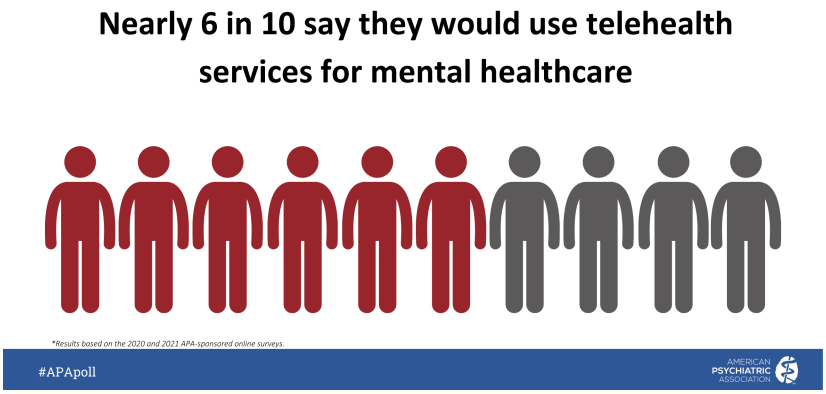

Figure 3: Popularity of telehealth for mental healthcare in USA [8]

The graph shown in Figure 4 explains the actual growth for years 2020-2022 and estimated growth for years 2023-2030, of telemedicine market in the North America region. The telemedicine market size was valued at USD 70.4 billion in 2021 and is estimated to grow at a compound annual growth rate (CAGR) of 19.5% from 2022 to 2030. The increase in demand to reduce the cost of healthcare, various companies merging or coming together within the industry, and other key initiatives by major companies are some of the main drivers resulting in the popularity of this market. The market holds many growth opportunities owing to the rising adoption of telemedicine [7].

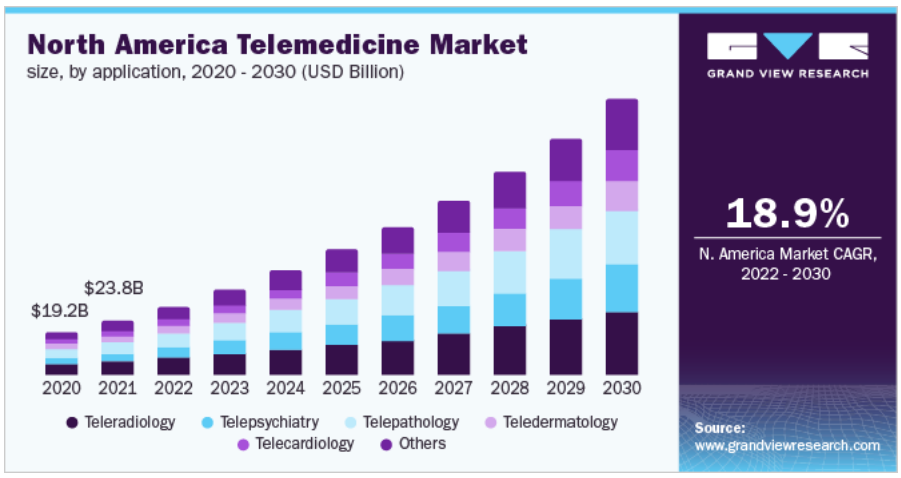

Figure 4: North America Telemedicine Market Size by Application, 2020-2030 (in USD billion) [7]

4 Project Design

The primary objective of TelecarePLUS is to establish itself as a healthcare platform incorporated with teleconsultation features, patient registration, appointment scheduling, electronic medical records management and overall promote efficient healthcare delivery. The aim of this platform is to address the challenges imposed by the traditional measures and improve accessibility to medical services despite of limitations and constraints.

The above mentioned goals can only be achieved by identifying the correct target population, building user friendly and intuitive designs, implement best data collection and storage techniques and accurately analyse and interpret the received data. These measures would help this proposed system fill the gaps of the underlying system. The methodologies can be conducting questionnaires, surveys and interviews, distributed through various electronic means like email, social media platforms or text messages, to understand the exact needs and preference of the users. Factors like user's physical location, familiarity with technologies, ease of access to technology and preferred mode for consultation should be highly considered in this scenario. Moreover, the project design of this proposed application includes the following:

- **Scope and Features:** The proposed application will focus on a range of features, both implemented and hypothetical. The features include simulated and secure audio/video/text consultations between patients and providers, patient registration, appointment scheduling, conceptualized electronic medical records management and prescription management system and remote patient monitoring.

  The whole application is conceptualized to be divided into three parts:

  - **Patient Portal (Implemented):** The application features a patient portal, where a patient can register themselves as new or existing patient. The patient is asked to verify themselves via OTP received in their email address. The patient then chooses from the list of doctors they want to consult with. On proceeding further, the patient chooses a desired time slot for consultation and proceed to book the appointment. On successful booking, the patient receives a link to the virtual consultation room in their email.

  - **Doctor Portal (Conceptualized):** On the other end, doctors will have a dedicated portal where they can login using their email or mobile and a password. The login system will also support Multi Factor Authentication (MFA) or One Time Password (OTP) for security and blocking unauthorized access. The portal's dashboard has the list of consultations for the current date. Moreover, the dashboard will also have a feature to show data by date, week, month or filter by appointment types and status.

    On successful appointment booking from the patient side, the video call

option will be available to the doctor who joins the same virtual consultation room, which was earlier received by the patient. Both the parties will only be allowed to join the room 15 minutes prior to the consultation time to restrict unwanted check-ins. The patient and the doctor talk over the video, which will also have a chat feature alongside to share any relevant information or file upload features to upload reports and lab tests results.

- **Electronic Medical Records (EMR) Management System (Conceptualized):** The doctor will meanwhile keep the track of the symptoms and diagnosis through a medical note. The note is finally shared to the patient through email and text message as an e-prescription. A copy of e-prescription is also shared with the pharmacy and the laboratory with patient's consent, that will further contact the patient for the suggested drugs and tests. Finally, the drugs will also be sent to the patient's address through online payment and delivery. Moreover, they will also have an option to pickup their drugs from a pharmacy nearby.

  Meanwhile, doctors will also have an option to generate the prescription template in the web application. It is the same template that they use to write prescriptions on paper. This ensures that no manual intervention occurs and all the records are entered and managed by the system itself.

The whole process of consultation with the above proposed application model is automated without the need of physically visiting the place by both the parties. This application indeed has what it takes to redefine telemedicine.

- **Target Users:** This proposed solution targets both providers and patients. Healthcare providers are represented by doctors, nurses and specialists in the healthcare field. They will be utilizing TelecarePLUS to provide services to the patients. On the other hand, the patients will be using the application to receive those services virtually. Patients, especially with mobility constraints and location hindrances, are primarily focused.

- **User Experience Design:** TelecarePLUS will emphasize on a user friendly interface with proper navigation and user interactions. The focus will be mainly on the essential features either implemented or conceptualized.

- **Data Privacy and Security:** The platform will focus to ensure the privacy, integrity and confidentiality of patient's data. These factors will be a top priority to consider and the application will explore conceptual measures to prevent sensitive data from unwanted access and breach. The system should have strong encryption techniques, access control mechanism and role based permission measures right in place. These measures will limit data access to authorized users only.

- **Technology Stack:** Table 1 lists the technological stack that has been proposed for TelecarePLUS.

IDE for Development	PyCharm Professional Edition [10]
DBMS	AWS DynamoDB (NoSQL) [11]
Programming Language and Framework	Python 3 with Serverless [12] [13]
Frontend Technology	ReactJS [14]
Cloud Infrastructure	AWS [15]
Library	Python boto3 [9]
Version Control	Git [16]
Authentication and Authorization	AWS Cognito [17]
API Management	AWS API Gateway [18]
Storage	AWS S3 [19]
Notification	AWS SNS [20], AWS SQS [21], AWS SES [22]
Logs Monitoring	AWS CloudWatch [23]
Secondary Storage	AWS Athena [24], AWS Elasticsearch [25]

Table 1: Tech stack for TelecarePLUS

- **Integration with Existing Systems:** TelecarePLUS will theoretically discuss of integrating with existing solutions to provide seamless data exchange between old and new systems. It encompasses theoretical methods for data synchronization ensuring that up-to-date patient data availability for healthcare service providers. There are many standards like HL7 [26] and FHIR [27] that promote data consistency, compatibility and seamless data exchange. Overall, the application should be a useful extension of the existing application rather than being a standalone application.

- **Development Timeline:** Table 2 explains the development timeline that has been followed for TelecarePLUS:

Phase	**Duration (Weeks)**	**Milestones/Deliverables**
1: Planning	2	– Project initiation. – Defining project objectives. – Literature review. – Determining project methodologies. – Approval of the project.
2: Feature Implementation	4	– Backend architecture planning. – Development of patient registration features. – Implementation of REST APIs.
3: Testing and Quality Assurance	1	– Unit testing. – Usability testing. – Implement optimization measures.
4: Deployment	2	– AWS deployment.
5: Evaluation and Finalization	1	– Evaluation of the proposed application. – Finalizing project documentation. – Presentation of TelecarePLUS.

Table 2: Development Timeline for TelecarePLUS

- **Risk Assessment:** Like every other application, TelecarePLUS is susceptible to risks and threats. The risks may include data breach, technical vulnerabilities, user side issues, regulatory issues and major challenges related to data integrity and security. These risks and threats have the potential to disrupt system functionality, compromise sensitive data and trust. To address these risks, there are certain mitigation measures like access controls, authentication, authorization, encryption and security audits.
- **Future Considerations:** TelecarePLUS tries to incorporate a lot of features while also leaving room for future enhancements and improvements. Usability testing helps in evaluating the ease of use of the application among the target users. These data will help to identify any areas for enhancements or improvements in the application.

5 Project Implementation and Results

The implementation of the project is explained in below subsections, with each subsections giving examples in the form of code snippets:

5.1 Project Setup

The project is built using Python programming language and Serverless framework. The operating system environment is Windows 11. The pre-requisite to install Serverless is that NPM is already installed on the computer. The commands mentioned below in Listing 1 installs serverless and sets up the project.

```
$ npm install -g serverless
$ mkdir ~/telecare-plus
$ cd ~/telecare-plus
$ sls create -n telecare-plus -t aws-python3
```

Listing 1: Install and setup a Serverless Python project

5.2 Classes

TelecarePLUS comprises of various classes as separate entities. These classes server as a model to perform operations on the database. The list below mentions all the classes used in TelecarePLUS. The class examples in Listing 2, Listing 3 and Listing 4 of the appendix section A.1.1 give some code snippets for the base, patient and appointment class respectively.

- Base Class
- Patient
- Appointment
- Prescription
- Doctor
- User
- Settings
- Video room
- Payment

All the classes are inherited from the parent class, BaseModel. The BaseModel is shown in Listing 2. The constructor of the class has the default values initialized for all the attributes. The class also contains some static methods to build and return the value of primary and secondary index keys for the AWS DynamoDB NoSQL database. The partition key (pk) basically retrieves the partition of the data to be fetched and the sort key (sk) sorts the data within that partition. We can see how

Patient and Appointment model inherit the BaseModel in Listing 3 and Listing 4 respectively.

The combination of both pk and sk, mentioned as id in the class examples, uniquely identifies the data. For example, in appointment class, the value of pk is account_id and appointment_date, which are the values of hospital id and the date of appointment respectively. Both the values are separated by a colon (:) sign. This key helps us retrieve all the appointments for a particular hospital and a particular date.

The value of sk is the text *appt* and doctor_id, which is the unique id for a doctor separated by a colon (:). On successfully retrieving a partition of data using pk, the sk sorts the data in the partition and returns the list of all appointments for a particular doctor and date for a specific hospital id.

There is a separate secondary index called Global Secondary Index(gsi) which is a type of index that contains both pk and sk, but different from the table's pk and sk. They are called global because they help to retrieve data at once from multiple partitions [32].

The data for all the classes are stored in a single table with each data distinguished on the basis of the values of pk, sk and gsi keys.

5.3 Helper Code

These helper code are general functions that are used multiple times over the application for various purposes. The functions mentioned in appendix section A.1.2 are examples of code that help the system interact with AWS services.

As seen in the appendix section A.1.2, the code imports and uses boto3 to initialize AWS services. Boto3 is a package that has the implementation of Python SDK for AWS CLI [9]. The package helps the python code access and interact with AWS services.

Listing 5 explains how a s3 client is created using boto3 and the file is uploaded to the bucket based on the filename and bucketname mentioned in lines 7 and 8 of the code snippet in Listing 5.

Listing 6 using SES client and sends email from the receiver to the sender. The email subject is mentioned and the email body is constructed in the code itself.

The code snippet in Listing 7 is a decorator function around every AWS lambda function that gets the request from the event and assigns it to the payload variable.

The code in Listing 8 describes how a SNS client, initialized using boto3, can be used to publish a message as notification to different devices. The topic ARN and the message are sent in the publish method and the message id after successful publish is returned back to the function.

DynamoDB can handle events and trigger functions based on the events. The code in Listing 9 explains how INSERT, MODIFY and REMOVE events in DynamoDB can trigger functions to add, update and remove data from ElasticSearch index respectively. Data in ElasticSearch index is stored as documents under a unique document ID for every data.

5.4 Lambda functions

AWS Lambda functions are basically small units of code executed in the AWS environment without the need to manage servers. The first lambda function mentioned in Listing 10 of the appendix section A.1.3, takes payload from the event and creates a patient in the DynamoDB table. The task to get payload from the event is done by the api_handler decorator mentioned in Listing 7.

The code gets the values from the payload and performs the necessary validations. On successful validation, the code initializes an object for the patient class and inserts a put item operation on the DynamoDB table to insert the patient data. The function then returns a response and a status code of 200 to the frontend for rendering it to the user. In case of error, the try-except block handles the error and returns the appropriate message to the front end and inform the end user.

The second lambda function in Listing 11 of the appendix section A.1.3, generates an OTP and sends it to the user via email. The function generates a random six digit OTP through a helper function and then uses boto3 to access AWS SES (Listing 6) and send the OTP to the end user's email address as received in the request.

6 Conclusion and Future Works

Traditional healthcare measures imposes some hindrances in delivering services to the end users. Likewise, it also puts up some challenges for the healthcare providers. These challenges were especially highlighted after the outburst of COVID-19 pandemic in 2020 [28]. Looking at the positive side, the pandemic also gave exponential rise to the prospects of virtual care and telemedicine [29].

A comprehensive one-stop healthcare platform can automate and enrich the entire process of serving patients and prove equally effective for the providers. This is where telecarePLUS comes into the picture. The user-friendly and ease of access nature of the application facilitates in providing the best quality services efficiently and at low cost. Moreover, the broader scope of this application also keeps data integrity, confidentiality and privacy as its topmost priority as medical data are sensitive and should be handled carefully.

Forthcoming initiatives for this project involves implementing the conceptualized ideas defined in the scope and features sub-section of the project design section. Moreover, there are numerous exciting possibilities for the future development and enhancement of telecarePLUS. Additional efforts to strengthen data integrity and privacy will also be a crucial factor that will be considered to battle the continuous evolution of data related threats [30].

The capstone project, as a part of an academic requirement, lacks all the third party integration required for payment, video call, email, text message and calls. These expansion will result in a more holistic application and contribute to provide better user experience results. The integration with existing EMR systems will also be equally prioritized as this new application should be able to take over from the existing application without any issues and keep providing seamless experience to the end users. Continuous collaboration with healthcare bodies and individuals will also play a crucial role in refining the application to meet and adjust to the dynamic needs to the healthcare paradigm [31].

In conclusion, telecarePLUS has laid a strong foundation to transform how healthcare services are delivered to the end users, especially those in utter need of it, efficiently and effectively.

7 Contributions

This project report aims to provide a case for TelecarePLUS as a healthcare platform with diverse set of features, primarily focused to overcome existing traditional challenges, while also prioritizing data privacy, confidentiality and ease of access. Below is a list of its contributions:

- **Core Features:** This project report presents the core features of TelecarePLUS as a one-stop healthcare platform includes secure audio/video/text consultations, patient registration, electronic medical records management system, prescription and reports management and patient monitoring. These features contribute to improve healthcare accessibility and efficiency.
- **User Friendly Design:** This project emphasizes the user-friendly design of TelecarePLUS as a healthcare platform ensuring seamless and user friendly experience for both service providers and patients. Patients, especially those in remote and undeserved areas and with mobility restrictions, can access healthcare services conveniently.
- **Data Privacy and Security:** This research project highlights the importance of data integrity, data privacy and confidentiality for a patient's sensitive medical data and the security measures needed to safeguard this data.
- **Addressing Challenges Through Innovative Approach:** This project report demonstrates the innovative approaches taken by TelecarePLUS as a cohesive system to tackle the challenges imposed by the traditional healthcare services as mentioned in the problem statement section.
- **Future Work:** Finally, this project report also discusses the future prospects of TelecarePLUS and how its adoption and success could carve a path for advanced technologies, contributing to the transformation of global delivery of healthcare services.

A Appendix

A.1 Code Examples

A.1.1 Classes

```
    class BaseModel:
        def __init__(self, updated_by = '', created_by = '',
   is_active = True, updated = '', created = ''):
        self.updated_by = updated_by
        self.created_by = created_by
        self.is_active = is_active
        self.updated = updated
        self.created = created

```

Listing 2: Base class

```
from model.base_model import BaseModel


class Patient(BaseModel):
    def __init__(
        self,
        first_name="",
        middle_name="",
        last_name="",
        country_code="",
        mobile="",
        email="",
        dob="",
        age={},
        gender="",
        patient_hid="",
        address={},
        year_of_birth=0,
        display_address="",
        joining_datetime="",
        guardian_name="",
        guardian_phone="",
        updated_by="",
        created_by="",
        is_active=True,
        updated="",
        created="",
        account_id="",
        patient_id="",
        pk="",
        sk="",
        id="",
        gsipk1="",
        gsisk1="",
        gsipk2="",
```

```
        gsisk2="",
        gsipk3="",
        gsisk3="",
    ):
        super().__init__(
            updated_by=updated_by,
            created_by=created_by,
            is_active=is_active,
            updated=updated,
            created=created,
        )
        self.first_name = first_name
        self.middle_name = middle_name
        self.last_name = last_name
        self.country_code = country_code
        self.mobile = mobile
        self.email = email
        self.dob = dob
        self.age = age
        self.gender = gender
        self.patient_hid = patient_hid
        self.address = address
        self.year_of_birth = year_of_birth
        self.display_address = display_address
        self.joining_datetime = joining_datetime
        self.guardian_name = guardian_name
        self.guardian_phone = guardian_phone
        self.account_id = account_id
        self.patient_id = patient_id
        self.pk = pk  # account_id:patient_id
        self.sk = sk  # pt
        self.id = id  # pk$sk
        self.gsipk1 = gsipk1  # account_id:patient_hid
        self.gsisk1 = gsisk1  # sk:patient_id
        self.gsipk2 = gsipk2  # account_id
        self.gsisk2 = gsisk2  # sk:joining_datetime
        self.gsipk3 = gsipk3  # account_id:mobile
        self.gsisk3 = gsisk3  # sk

    @staticmethod
    def get_pk(account_id, patient_id):
        return "{}:{}".format(account_id, patient_id)

    @staticmethod
    def get_sk():
        return "{}".format("pt")

    @staticmethod
    def get_id(pk, sk):
        return "{}${}".format(pk, sk)

    @staticmethod
    def get_gsipk1(account_id, patient_hid):
```

```
        return "{}:{}".format(account_id, patient_hid)

    @staticmethod
    def get_gsisk1(sk, patient_id):
        return "{}:{}".format(sk, patient_id)

    @staticmethod
    def get_gsipk2(account_id):
        return "{}".format(account_id)

    @staticmethod
    def get_gsisk2(sk, joining_datetime):
        return "{}:{}".format(sk, joining_datetime)

    @staticmethod
    def get_gsipk3(account_id, mobile):
        return "{}:{}".format(account_id, mobile)

    @staticmethod
    def get_gsisk3(sk):
        return "{}".format(sk)

```

Listing 3: Patient class

```
from model.base_model import BaseModel

class Appointment(BaseModel):
    def __init__(
        self,
        account_id="",
        allow_checkin=False,
        appointment_date="",
        appointment_slot="",
        appointment_id="",
        appointment_type="",
        appointment_mode="",
        appointment_month="",
        appointment_slot_timetsamp="",
        appointment_status="",
        doctor_id="",
        doctor_name="",
        enable_video=False,
        invoice_amount="",
        invoice_status="",
        is_patient_online=False,
        is_rescheduled=False,
        lsi1="",
        patient={},
        patient_id="",
        payment_order_details={},
        payment_order_id="",
        payment_status="Pending",
```

```
        room_id="",
        start_consultation=False,
        updated_by="",
        created_by="",
        is_active=True,
        updated="",
        created="",
        pk="",
        sk="",
        id="",
        gsipk1="",
        gsisk1="",
        gsipk2="",
        gsisk2="",
        gsipk3="",
        gsisk3="",
        gsipk4="",
        gsisk4="",
        appointment_hid="",
    ):
        super().__init__(
            updated_by=updated_by,
            created_by=created_by,
            is_active=is_active,
            updated=updated,
            created=created,
        )
        self.allow_checkin = allow_checkin
        self.appointment_date = appointment_date
        self.appointment_slot = appointment_slot
        self.appointment_id = appointment_id
        self.appointment_type = appointment_type
        self.appointment_mode = appointment_mode
        self.appointment_month = appointment_month
        self.appointment_slot_timetsamp = appointment_slot_timetsamp
        self.appointment_status = appointment_status
        self.appointment_hid = (appointment_hid,)
        self.doctor_id = doctor_id
        self.doctor_name = doctor_name
        self.enable_video = enable_video
        self.invoice_amount = invoice_amount
        self.invoice_status = invoice_status
        self.is_patient_online = is_patient_online
        self.is_rescheduled = is_rescheduled
        self.lsi1 = lsi1  # appt:appointment_hid
        self.patient = patient
        self.payment_order_details = payment_order_details
        self.payment_order_id = payment_order_id
        self.payment_status = payment_status
        self.room_id = room_id
        self.start_consultation = start_consultation
        self.account_id = account_id
        self.patient_id = patient_id
```

```
        self.pk = pk  # account_id:appt_date
        self.sk = sk  # appt:doctor_id
        self.id = id  # pk$sk
        self.gsipk1 = gsipk1  # account_id:patient_id
        self.gsisk1 = gsisk1  # appt:appointment_date
        self.gsipk2 = gsipk2  # account_id:doctor_id
        self.gsisk2 = gsisk2  # appt:appointment_date
        self.gsipk3 = gsipk3  # appt:appt_month
        self.gsisk3 = gsisk3  # appt:appointment_date
        self.gsipk4 = gsipk4  # appt:account_id
        self.gsisk4 = gsisk4  # appt:appt_date:doctor_id

    @staticmethod
    def get_pk(account_id, appointment_date):
        return "{}:{}".format(account_id, appointment_date)

    @staticmethod
    def get_sk(doctor_id):
        return "{}:{}".format("appt", doctor_id)

    @staticmethod
    def get_id(pk, sk):
        return "{}${}".format(pk, sk)

    @staticmethod
    def get_gsipk1(account_id, patient_id):
        return "{}:{}".format(account_id, patient_id)

    @staticmethod
    def get_gsisk1(appointment_date):
        return "{}:{}".format("appt", appointment_date)

    @staticmethod
    def get_gsipk2(account_id, doctor_id):
        return "{}:{}".format(account_id, doctor_id)

    @staticmethod
    def get_gsisk2(appointment_date):
        return "{}:{}".format("appt", appointment_date)

    @staticmethod
    def get_gsipk3(appointment_month):
        return "{}:{}".format("appt", appointment_month)

    @staticmethod
    def get_gsisk3(appointment_date):
        return "{}:{}".format("appt", appointment_date)

    @staticmethod
    def get_gsipk4(account_id):
        return "{}:{}".format("appt", account_id)

    @staticmethod
```

```
    def get_gsisk4(appointment_date, doctor_id):
        return "{}:{}:{}".format("appt", appointment_date, doctor_id
    )

    @staticmethod
    def get_lsi1(appointment_hid):
        return "{}:{}".format("appt", appointment_hid)

```

Listing 4: Appointment class

A.1.2 Helper Code

```
    import boto3

    # create a boto3 S3 client
    s3_client = boto3.client('s3')

    # define the S3 bucket name and file name
    bucket_name = 'e-prescriptions'
    file_name = 'prescription.jpg'

    # open the image file to upload
    with open(file_name, 'rb') as file:
        # upload the image file to S3 bucket
        s3_client.upload_fileobj(file, bucket_name, file_name)

```

Listing 5: Upload prescription images to AWS S3 bucket using Python and boto3

```
    import boto3
    from botocore.exceptions import ClientError

    # create a boto3 SES client
    ses_client = boto3.client('ses')

    # define the email variables
    sender = 'sender@senderemail.com'
    recipient = 'recipient@receiveremail.com'
    subject = 'Payment Link for Dr. XYZ - 17 Apr 2023'
    body = 'Dear Patient, Thank you for booking an appointment with
    Dr. XYZ for 17 Apr 2023 at 01:00 PM. Please pay for the
    consultation here: https://www.xyz.com/payhere'

    # send the email using SES
    try:
        response = ses_client.send_email(
            Source=sender,
            Destination={
                'ToAddresses': [
                    recipient,
                ],
            },
```

```
            Message={
                'Subject': {
                    'Data': subject,
                },
                'Body': {
                    'Text': {
                        'Data': body,
                    },
                },
            },
        )
        print("Email sent successfully.")
    except ClientError as e:
        print(e.response['Error']['Message'])
```

Listing 6: Send an email notification about payment

```
import json

def api_handler(handler_function):
    def wrapper(event, context):
        payload = {}
        if event['body']:
            payload = json.loads(event['body'])
        return handler_function(payload, context)

    return wrapper


```

Listing 7: Decorator to extract event payload from a REST API request

```
import boto3
import json

# Create a boto3 SNS client
sns_client = boto3.client('sns')

# Publish the message
response = sns_client.publish(
    TopicArn='arn:aws:sns:us-east-2:121212121212:telecareplus-sns-topic',
    Message=json.dumps({'message': 'Appointment Reminder with Dr. XYZ at 01:00 PM'}),
    MessageStructure='json'
)

# return the message ID of the message published to SNS
return response['MessageId']

```

Listing 8: Publish a message as notification to devices through SNS

```
import boto3
import json
from elasticsearch import Elasticsearch, RequestsHttpConnection
from requests_aws4auth import AWS4Auth

region = 'us-west-2'
service = 'es'
index_name = 'patient-index'
host = 'telecareplus-index-1234567890.us-west-2.es.amazonaws.com'
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service)

es = Elasticsearch(
hosts=[{'host': host, 'port': 443}],
http_auth=awsauth,
use_ssl=True,
verify_certs=True,
connection_class=RequestsHttpConnection
)

def dynamodb_trigger(event, context):
    for record in event['Records']:
        if record['eventName'] == 'INSERT':
            add_to_elasticsearch(record['dynamodb']['NewImage'])
        elif record['eventName'] == 'MODIFY':
            update_elasticsearch(record['dynamodb']['NewImage'])
        elif record['eventName'] == 'REMOVE':
            delete_from_elasticsearch(record['dynamodb']['Keys'])
        else:
            print(f'Event not supported: {record["eventName"]}')

def add_to_elasticsearch(item):
    item_id = item['id']['S']
    item_data = {
        'patient_name': item['patient_name']['S'],
        'patient_age': int(item['patient_age']['N']),
        'patient_mobile': item['patient_mobile']['S'],
        'patient_email': item['patient_email']['S'],
        'patient_health_card_number': item['patient_health_card_number']['S'],
        'patient_address': item['patient_address']['S']
    }
    es.index(index=index_name, id=item_id, body=item_data)

def update_elasticsearch(item):
    item_id = item['id']['S']
    item_data = {
        'doc': {
            'patient_name': item['patient_name']['S'],
            'patient_age': int(item['patient_age']['N']),
```

```
                'patient_mobile': item['patient_mobile']['S'],
                'patient_email': item['patient_email']['S'],
                'patient_health_card_number': item['
    patient_health_card_number']['S'],
                'patient_address': item['patient_address']['S']
            }
        }
        es.update(index=index_name, id=item_id, body=item_data)

    def delete_from_elasticsearch(keys):
        item_id = keys['id']['S']
        es.delete(index=index_name, id=item_id)


```

Listing 9: AWS Lambda function to listen to DynamoDB Stream events and upload or update data to Elasticsearch index

A.1.3 Lambda Functions

```
import json
import boto3

# Import the Patient class from the model
from model.patient import Patient

# Define the AWS Lambda function
@api_handler
def create_patient(payload, context):
    try:
        # Initialize DynamoDB client
        dynamodb = boto3.resource('dynamodb')
        table_name = 'emr_master'
        table = dynamodb.Table(table_name)

        # Extract patient data from the payload
        first_name = payload.get('first_name', '')
        middle_name = payload.get('middle_name', '')
        last_name = payload.get('last_name', '')
        country_code = payload.get('country_code', '')
        mobile = payload.get('mobile', '')
        email = payload.get('email', '')
        dob = payload.get('dob', '')
        age = payload.get('age', {})
        gender = payload.get('gender', '')

        # Create a new patient object
        patient = Patient()

        # Save the patient data to the DynamoDB table
        response = table.put_item(
            Item=patient.__dict__
        )
```

```
34
35          # For example, you can return the patient data as a response
36          response = {
37              'statusCode': 200,
38              'body': json.dumps({'message': 'Patient created
    successfully', 'patient_data': patient.__dict__})
39          }
40          return response
41
42      except Exception as e:
43          # Handle any exceptions that may occur during the process
44          response = {
45              'statusCode': 500,
46              'body': json.dumps({'error': 'An error occurred while
    creating the patient', 'details': str(e)})
47          }
48          return response
49
50
```

Listing 10: Create Patient

```
1  import json
2  import boto3
3  import hashlib
4  import random
5
6  # create a boto3 SES client
7  ses_client = boto3.client('ses')
8
9  # Define the AWS Lambda function
10 @api_handler
11 def send_otp(payload, context):
12     try:
13         # Extract mobile and email from the payload
14         mobile = payload.get('mobile', '')
15         email = payload.get('email', '')
16
17         # Generate a random 6-digit OTP based on the hash value of
    mobile and email
18         otp = generate_otp(mobile, email)
19
20         # Send the OTP to the user's email address using SES
21         sender = 'sender@senderemail.com'
22         subject = 'Your OTP for Verification'
23         body = f'Your OTP is: {otp}. Please use this OTP for
    verification.'
24
25         response = ses_client.send_email(
26             Source=sender,
27             Destination={
28                 'ToAddresses': [email],
29             },
```

```
            Message={
                'Subject': {
                    'Data': subject,
                },
                'Body': {
                    'Text': {
                        'Data': body,
                    },
                },
            },
        )

        response = {
            'statusCode': 200,
            'body': json.dumps({'message': 'OTP sent successfully to your email.'})
        }
        return response

    except Exception as e:
        response = {
            'statusCode': 500,
            'body': json.dumps({'error': 'An error occurred while sending OTP', 'details': str(e)})
        }
        return response

def generate_otp(mobile, email):
    # Generate a random 6-digit number as OTP
    otp = ''.join([str(random.randint(0, 9)) for _ in range(6)])

    # Create a hash value based on mobile and email
    hash_string = f"{mobile}{email}".encode('utf-8')
    hash_value = hashlib.sha256(hash_string).hexdigest()

    # Use the hash value to modify the OTP
    modified_otp = ''
    for i in range(len(otp)):
        modified_otp += otp[i] + hash_value[i]

    return modified_otp
```

Listing 11: Generate OTP

References

[1] medicaid.gov, *Telehealth.* [Online]. Available: https://www.medicaid.gov/medicaid/benefits/telehealth/index.html

[2] Grigsby, J., Kaehny, M. M., Sandberg, E. J., Schlenker, R. E., & Shaughnessy, P. W. (1995). Effects and effectiveness of telemedicine. *Health care financing review*, 17(1), 115–131. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4193577/.

[3] medmehealth.com, *What is Virtual Care?* [Online]. Available: https://www.medmehealth.com/blog/what-is-virtual-care

[4] fiercehealthcare.com, *As the telehealth market shakes out, Teladoc, Amwell feeling pressure from new entrants, more specialization.* [Online]. Available: https://www.fiercehealthcare.com/tech/as-telehealth-market-shakes-out-teladoc-amwell-feeling-pressure-from-new-entrants-more.

[5] mckinsey.com, *Telehealth: A quarter-trillion-dollar post-COVID-19 reality?* [Online]. Available: https://www.mckinsey.com/industries/healthcare/our-insights/telehealth-a-quarter-trillion-dollar-post-covid-19-reality.

[6] healthtechmagazine.net, *6 Reasons Telehealth Is Now More Important Than Ever.* [Online]. Available: https://healthtechmagazine.net/article/2020/05/6-reasons-telehealth-now-more-important-ever.

[7] grandviewresearch.com, *Telemedicine Market Size, Share & Trends Analysis Report By Component (Products, Services), By End User (Patients, Providers), By Application, By Modality, By Delivery Mode, By Facility, And By Segment Forecasts, 2022 - 2030* [Online]. Available: https://www.grandviewresearch.com/industry-analysis/telemedicine-industry.

[8] psychiatry.org, *New Nationwide Poll Shows an Increased Popularity for Telehealth Services* [Online]. Available: https://www.psychiatry.org/news-room/news-releases/new-nationwide-poll-shows-an-increased-popularity.

[9] boto3.amazonaws.com, *Boto3 documentation* [Online]. Available: https://boto3.amazonaws.com/v1/documentation/api/latest/index.html

[10] jetbrains.com, *PyCharm* [Online]. Available: https://www.jetbrains.com/pycharm/

[11] aws.amazon.com, *Amazon DynamoDB* [Online]. Available: https://aws.amazon.com/dynamodb/

[12] python.org, *Python* [Online]. Available: https://www.python.org/

[13] serverless.com, *Serverless Framework* [Online]. Available: https://www.serverless.com/

[14] react.dev, *React - The library for web and native user interfaces* [Online]. Available: https://react.dev/

[15] aws.amazon.com, *Start Building on AWS Today* [Online]. Available: https://aws.amazon.com/

[16] git-scm.com, *git –distributed-is-the-new-centralized* [Online]. Available: https://git-scm.com/

[17] aws.amazon.com, *Amazon Cognito* [Online]. Available: https://aws.amazon.com/cognito/

[18] aws.amazon.com, *Amazon API Gateway* [Online]. Available: https://aws.amazon.com/api-gateway/

[19] aws.amazon.com, *Amazon S3* [Online]. Available: https://aws.amazon.com/s3/

[20] aws.amazon.com, *Amazon Simple Notification Service* [Online]. Available: https://aws.amazon.com/sns/

[21] aws.amazon.com, *Amazon Simple Queue Service* [Online]. Available: https://aws.amazon.com/sqs/

[22] aws.amazon.com, *Amazon Simple Email Service* [Online]. Available: https://aws.amazon.com/ses/

[23] aws.amazon.com, *Amazon CloudWatch* [Online]. Available: https://aws.amazon.com/cloudwatch/

[24] aws.amazon.com, *Amazon Athena* [Online]. Available: https://aws.amazon.com/athena/

[25] aws.amazon.com, *Amazon OpenSearch Service* [Online]. Available: https://aws.amazon.com/opensearch-service/

[26] hl7.org, *HL7 International* [Online]. Available: https://www.hl7.org/

[27] hl7.org, *HL7 FHIR* [Online]. Available: https://www.hl7.org/fhir/overview.html

[28] Filip, R., Gheorghita Puscaselu, R., Anchidin-Norocel, L., Dimian, M., & Savage, W. K. (2022). Global Challenges to Public Health Care Systems during the COVID-19 Pandemic: A Review of Pandemic Measures and Problems. *Journal of personalized medicine*, 12(8), 1295. DOI: https://doi.org/10.3390/jpm12081295

[29] mercomcapital.com, *Telehealth Companies Raised a Record $5 Billion in the First Half of 2021* [Online]. Available: https://mercomcapital.com/telehealth-companies-raised-a-record-5-billion-in-the-first-half-of-2021/.

[30] currentware.com, *The Impact of Cyberattacks on Healthcare* [Online]. Available: https://www.currentware.com/blog/the-impact-of-cyberattacks-on-healthcare/.

[31] Haraldseid-Driftland, C., Billett, S., Guise, V. et al. The role of collaborative learning in resilience in healthcare—a thematic qualitative meta-synthesis of resilience narratives. *BMC Health Serv Res* 22, 1091 (2022). DOI: https://doi.org/10.1186/s12913-022-08451-y

[32] dynobase.dev, *DynamoDB Global Secondary Index (GSI) - The Ultimate Guide. (n.d.)* [Online]. Available: https://dynobase.dev/dynamodb-gsi/.