UNIVERSITY OF THE EAST

Caloocan City

College of Engineering

Computer Engineering Department

Appointment Management System for Skyline Hospital and Medical Center with Data Analytics

In partial fulfillment of the requirement for NCP 4990 - CpE Practice and Design 2

for the Degree of Bachelor of Science in Computer Engineering

Presented by:

Bajar, Jhusthine G.

Britanico, Albert R.

Gavan, Isabelle Therese L.

Lanzuela, Daniel Mark A.

Samson, Joaquin Paulo N.

Dr Nelson C. Rodelas

Thesis Adviser

ACKNOWLEDGMENTS

A heartfelt thanks first and foremost goes to the **Omnipotent Lord God**, **Almighty**, who has blessed us, guided us consistently, and led us to the successful fruition of this research study.

We would like to express our sincere gratitude to **Dr. Nelson C. Rodelas**, **PCpE,** our thesis adviser, for guiding and mentoring us in shaping our research topic and title. His profound understanding of the field not only honed our skills as researchers but also ignited a genuine passion for our study.

We offer our deepest gratitude to **Skyline Hospital and Medical Center**, for allowing us the opportunity to develop a system specifically for their hospital. We acknowledge and value the collaborative efforts of **Atty. Paulo Antonio B Reyes**, **JD**, **FPCHA**, **MHRM**, the **Human Resource Manager**, for granting us the authorization to conduct this research, **Mr. Eljune Gallo**, the **IT Head,** for providing us with the essential data required for the development of our system, and **Ms. Jamila Bautista**, the **Outpatient Department Supervisor**, who shared valuable insights regarding the hospital's scheduling process. Furthermore, we would like to extend our appreciation to **Mr. Isaiah Gavan**, the **Senior Admission Staff**, for his ideas and assistance in facilitating effective communication with the hospital.

To our esteemed research panelists: **Dr. Joan P. Lazaro**, **Engr. Mark Vernon M. Degala**, **Dr. Alexis John M. Rubio**, and **Engr. Ronnel P. Agulto**, we recognized their valuable feedback which has greatly contributed to the improvement of our research. We

acknowledge that their evaluation will mold us into more proficient professionals, enabling us to apply this knowledge effectively in the future.

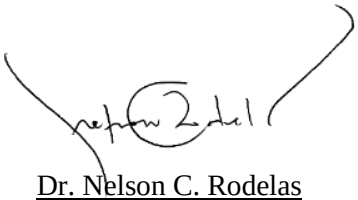
Finally, wholehearted gratitude to our fellow researchers, whose contributions greatly helped the creation process of our research. We are ever thankful to everyone who provided us with support and confidence throughout this journey. Without their encouragement, completing this research paper would not have been possible.

TABLE OF CONTENTS

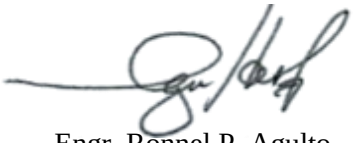
ACCEPTANCE AND APPROVAL FORM

This project study entitled **Appointment Management System for Skyline Hospital and Medical Center with Data Analytics** is prepared and submitted by **Jhusthine G. Bajar**, **Albert R. Britanico**, **Isabelle Therese L. Gavan**, **Daniel Mark A. Lanzuela and Joaquin Paulo N. Samson** in Partial Fulfillment of the Requirements for the Subject **NCP 4990 - CpE Practice and Design 2** is examined and recommended for Acceptance and Approval.

Recommended for Approval:



Dr. Nelson C. Rodelas
Adviser



Engr. Ronnel P. Agulto
Panelist 1

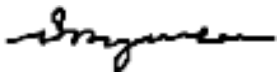
Engr. Mark Vernon M. Degala
Panelist 2

Dr. Joan P. Lazaro
Panelist 3

Dr. Paraluman G. Sim
Department Chair

CERTIFICATE OF EDITING

This is to certify that I have edited this thesis entitled, **"APPOINTMENT MANAGEMENT SYSTEM FOR SKYLINE HOSPITAL AND MEDICAL CENTER WITH DATA ANALYTICS."** prepared by **JHUSTHINE G. BAJAR, ALBERT R. BRITANICO, ISABELLE THERESE L. GAVAN, DANIEL MARK A. LANZUELA,** and **JOAQUIN PAULO N. SAMSON** and have found it to be thorough and acceptable with respect to grammar and composition.



PROF. DARWIN M. GUAINAN

GRAMMARIAN

Date of Completion: May 16, 2023

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Skyline Hospital and Medical Center provides a wide range of medical services, however there are problems that the hospital encounters in appointment booking. The hospital mostly relies on their telephone, messenger or walk-ins. The hospital has made various arrangements to make the appointment process easy and convenient for the patients. This paper aims for the development of the appointment management system with data analytics for Skyline Hospital and Medical Center, allowing the hospital to improve the appointment process. The system has three user types, patient users, doctor users, and the administrator. The web application system is capable of being responsive, so the users will be able to book for appointments in mobile and desktop browsers. Time series forecasting lets the hospital receive future forecasts of patient arrivals in the outpatient department. Machine learning in healthcare benefits both patients and the hospital to allow for tactical decision-making, better resource allocation, and become informed whether there will be an influx of outpatients or not. The system was tested on both mobile and desktop devices, which proved to be functional, and the future forecasts of patient arrivals were done and gave considerable results.

Keywords: Appointment Management System, Time Series Forecasting, Web Application, Outpatient Department, Machine Learning in Healthcare, Hospital Appointment

Chapter I

THE PROBLEM AND ITS BACKGROUND

This chapter features the background of the study, the statement of the problem, the scope and delimitations of the study, and the operational definition of terms.

1.1 Background of the Study

Numerous challenges afflicted the public health sector in various nations worldwide. These must be adequately managed to prevent substandard implementation of services in public hospitals and medical institutions. A number of the concerns comprised insufficient suitable medical equipment, incompetent personnel, congestion, emergency response delays, and inaccurate laboratory tests as a result of traditional medical data-gathering systems (Mandava et al., 2016). Those issues have only been exacerbated due to the COVID-19 (SARS-CoV2) which immensely scattered the infection. The demands of patients continually changed and greatly affected hospitals' capabilities and general functions (Žižović et al., 2021). Despite the numerous setbacks brought by the pandemic and other threats, healthcare and hospitals remain to be augmented even further.

A smart hospital is a healthcare facility with enhanced patient care and administration quality. First-rate technology, optimized network speed, secured data systems, and knowledgeable personnel characterize it. Advanced health data systems aid in medical service, it allows doctors to record essential health information about patients, from dossiers to regimen planning and hospital care (Velibor, 2019). As various kinds of

digital health information and records are being kept in hospitals, big data analytics play a huge role in smart hospitals and healthcare. Big data analytics can examine numerous complicated data and create valuable judgments that would not have been feasible differently. When used for health information, it is capable of uncovering trends, resulting in enhanced healthcare quality, lower expenditures, and facilitating rapid decisions (Costa, 2014; Raghupathi & Raghupathi, 2014; Rumsfeld, 2016; Wang & Hajili, 2018, as cited in Mehta & Pandit, 2018). Data visualization dashboards are created with big data analytics. They are designed to provide effortlessly understandable graphics that enhance user capability to swiftly and correctly digest content so that inferences, judgments, and movements may be decided (Stadler et al., 2016). Through historical data analysis, forecasting future values is possible with time series analysis. When applied in healthcare, it makes it possible for hospitals to have better resource allocation and strategic planning, especially when there is an influx of patient arrivals (Kadri et al., 2014). As the world continues to develop with technological advancements in hospitals and healthcare, the same cannot be said for other nations.

Philippine hospitals recognize that there is a perceived demand for medical facilities (Herrera et al., 2010; Lavado et al., 2010; FronczekMunter, 2013; Santos, 2014; De Dios, 2016, as cited in Dela Cruz & Ortega-Dela Cruz, 2019). Observed needs are information technology (IT) competencies; physical aspects of the hospital such as buildings, furniture, fixtures, utilities, ventilation, safety, security, devices, and machines. Additionally, advancements considered critical to improving hospital infrastructure and overall operations involve innovation and the adoption of appropriate technology (Dela Cruz & Ortega-Dela Cruz, 2019).

The accessibility of hospital beds indicates the receptiveness of health services. Because of the Philippines' tremendous population increase, hospital volume has become a problem that must be handled to ensure that patients receive essential services and have admittance to medical aid. Only four of seventeen areas in the country met the customary regional hospital bed quota. Internationally, only the National Capital Region (NCR) met the criteria set by the World Health Organization. The utilization of systems modeling with system dynamics built a model that determined factors that immensely affected medical services. It turned out that the medical staff, the populace, and financial sources were the causes. A 10-year simulation predicted that the government might be able to meet inpatient bed demand by 2021. But it is only achievable if public and private hospitals give equitable remuneration to medical practitioners and devote themselves to frameworks and facilities. It includes obtaining more inpatient beds to meet the rising populace's medical service insistence (German et al., 2018).

The Philippines' Health Status Index shows that the country lags behind much of Southeast and North Asia in healthcare outcomes as hospitals prefer traditional ways of scheduling appointments. The outpatient department is the main medical facility for non-urgent patients, but it often struggles with the long waiting time. The challenge is usually the lack of a reliable appointment system (Mendoza et al., 2019).

Skyline Hospital and Medical Center (SHMC) aspires to be the finest healthcare contributor north of Metro Manila. The personnel consist of devoted, diligent, board-certified members with several years of experience in diverse specialties and fields that can offer top-tier healthcare services. Their mission is to deliver first-rate healthcare and to gratify health services. SHMC faces a common problem among healthcare facilities, the

extensive waiting time in the outpatient department. The usual delay consumes about half an hour, which places people with deteriorating health conditions in tough situations. The hospital struggles with properly managing patient appointments with the recent surge of the COVID-19 pandemic, the dengue outbreaks, and the usual ailments that get people admitted to the hospital, making scheduling hectic. Properly scheduling and managing appointments is an important factor in reducing waiting time, and having an electronic service will efficiently use the patient's time. The hospital currently relies on telephone, mobile phone, and Facebook Messenger chat conversations to book appointments. To align with SHMC's vision and mission, the researchers propose an appointment management web application system.

The main objective of the researchers was to develop an appointment management web application system for SHMC. Time series analysis with dashboard visualizations will be utilized for forecasting the number of patient arrivals in the outpatient department (OPD) based on SHMC’s historical time series patient arrival records.

1.2 Statement of the Problem

This study aims to develop a system that would help the Skyline Hospital and Medical Center to improve the medical services of the hospital for their clients. A web application with a time series forecasting feature of patient arrivals is developed. Specifically, it seeks to answer the following questions:

1. What are the features of the system that will organize the following:

    1.1. Appointment of patients; and

1.2. Schedules of doctors and specialists?

2. How will the system manage the patients' appointments and schedules of doctors and specialists?

3. How will the time series data be organized and create forecasts of future patient arrivals?

4. How will the system be evaluated using the ISO 25010 software standard criteria in terms of:

    4.1. Functional Suitability;

    4.2. Performance Efficiency;

    4.3. Compatibility;

    4.4. Usability

    4.5. Reliability

    4.6. Security;

    4.7. Maintainability;

    4.8. Portability; and

    4.9. Forecasting Ability?

1.3 Scope and Limitations of the Study

This study focuses on developing a web application to manage appointments for Skyline Hospital and Medical Center. Since the institution does not have an existing web application for appointments, the system will be an excellent and functional online service.

To complement the appointment management system, insights and visualization from time series forecasting analytics are implemented. The features of the system are enumerated:

Scope:

- The appointment management system implemented three user features: patients, admins, and doctor interfaces.
- Patient users can book for appointments, check appointment history, check notifications, and edit or update profiles.
- Doctor users can check for booked appointments, check for notifications, set available time and schedule, and edit or update profile information.
- An in-app notification system is integrated to remind the appointments of both patients and doctors.
- Time series forecasting is implemented to forecast the future number of patient arrivals daily in SHMC's outpatient department with visualization.
- The system allows for booking of appointments for SHMC's outpatient department.
- The administrator has full control of the system and can manage the patients' and doctors' accounts, appointments, schedules, and the forecasting analytics dashboard.
- The system will manage the appointments of the registered patients.

Limitations:

- The system did not include a feature for managing and processing payments for appointments.

- The system requires an internet connection in mobile or desktop browsers in order to access and use it.
- Data for time series forecasting is exclusively gathered from SHMC's historical time series data.
- The time series forecast of patient arrivals in SHMC is exclusive for the outpatient department; specific time series forecasts of patient arrivals for every sub-department will not be implemented.
- Appointments made through phone calls and chat conversations will not be managed by the system.

1.4 Significance of the Study

An appointment management system, in the context of healthcare, is essential for managing patients, doctors, specialists, and other hospital staff. A web application developed for this system can substantially solve common management problems in healthcare institutions. This part of the research highlighted the beneficiaries or those who would benefit from the study. The beneficiaries of the research are stated as follows:

Healthcare institutions. The appointment management system has been specifically developed for Skyline Hospital and Medical Center to improve the institution's current system. But various other hospitals, clinics, and other healthcare institutions without a digitized system like this may also benefit from having one. With the integration of forecasting analytics, it can help institutions prepare and properly allocate resources especially if the number of predicted patients to come are high.

Patients. The study can help patients experience lesser trouble in making appointments with doctors and specialists. Using the web application can eliminate the inconvenience of extensive and unnecessary waiting periods.

Doctors and specialists. Having access to the scheduled appointments of patients through the appointment management system application can help doctors and specialists execute better time management and service quality. This allows them to establish improved connections and relationships with patients.

Receptionists. Being the ones responsible for the initial accommodation of patients, the appointment management system can help them quickly verify the scheduled appointments of patients. They may also aid patients in booking appointments through the web application to ensure that all patients can correctly do them.

Researchers. Acting also as developers of the appointment management system application, the researchers can further enhance their intellect and experience with developing software with data analytics. Future researchers can also add additional upgrades and integrations into the system for better healthcare quality overall.

1.5 Operational Definition of Terms

Bootstrap - A front-end framework used to help the system be responsive, wherein the UI display becomes adaptive in both desktop and mobile devices.

Cascading Style Sheets (CSS) - A frontend markup language that is used to make styles for the web application system.

Data Analytics - This refers to the analysis, interpretation, and visualization of data in order to produce useful information, insights.

Database - A data repository that is responsible for the collection, storing, and organization of data that is used in the system such as usernames, passwords, and the dataset for data analysis.

Dataset - It refers to the gathered data for analysis. The datasets in this study would be gathered from Skyline Hospital and Medical Center's historical patient arrival records.

Django - A Python-based web framework. It allows for construction of websites from the ground up rather than beginning from scratch.

Django Rest Framework - It is an API that helps in pagination and authentication of the web application.

HyperText Markup Language (HTML) - It is used to structure and organize the contents of the web application.

JavaScript - It is used as the scripting language in the client side. It helps add functionality in the application.

JupyterLab - This is used as the environment for building the time series forecasting model using the Python programming language.

PostgreSql - This is a database management system that can support a wide range of applications and it can store and retrieve data.

Python - A general purpose programming language that is used to create and build the time series forecasting model, as well as the backend side of the system with Django.

Vue.js - A JavaScript framework to design the web application system, creation of front-end UI, and the other components of the system.

Chapter II

REVIEW OF RELATED LITERATURE AND STUDIES

This chapter includes an examination of related literature and studies, both international and domestic, as well as a synthesis and evaluation of the cited findings.

2.1 Hospital Appointment System Features

Tamayo (2018) developed a multi-platform outpatient medical system to address the challenges of the existing manual procedure in the Philippines, which relies heavily on paper-based records and results in the accumulation of significant patient files and no-shows that disrupt hospital workers' time and resources. The system features four components: administrator, secretary, doctor, and outpatient. The administrator component provides prime control over system users, outpatients, medical services, and providers, while the secretary component is intended for clinic secretaries to coordinate doctor appointments. The doctor component enables doctors to manage their scheduling, and the outpatient component is for patients to schedule appointments online using a browser and a live Internet connection. The system was developed using online tools such as CSS, HTML, JavaScript, and other platforms for the front end, and Lavarel PHP framework for the back end. The system's data is stored in a MySQL database, and SMS notifications are sent using the Twilio cloud communication platform.

Additionally, a study on the development of a smart health doctor appointment system using Python (Django) as the front end and Postgres-SQL as the back end was conducted by Shelwante et al. (2019). The system provides an online booking framework

for patients to conveniently and securely make reservations through a website. It consists of three modules: a patient module, a doctor module, and a message notification feature. Patients need to register and login to search for doctors by location, specialty, and travel time. Once they find a suitable doctor, they can schedule an appointment with a dentist, eye doctor, or neurologist. Doctors, on the other hand, need to register and provide their phone number and specialty before accessing the Dashboard. In the Dashboard, they can view scheduled appointments and accept appointment requests. The system also includes Google Firebase notifications for reliable and cost-free message delivery.

Similarly, Malik et al. (2017) created a hospital management application called "Mr. Doc" for Android that has modules for patients and administrators. The application was created using Android Studio, HTML, and PHP, with data transfer and access through APIs between the website and the Android application. Upon installation, the patient module requires registration and presents a menu screen displaying hospitals, doctors, and their availability schedules. Patients can view the list of hospitals, choose a doctor, and schedule appointments via phone or email for a specific date and time. The admin portal can add doctors, access their details, and view the meeting schedules of registered patients.

According to Alghamedi (2011), the Mother and Child Health Department of Klinik Kesihatan Changlun currently relies on manual methods for administration. Patient registration and record-keeping are paper-based, with appointments recorded on cards and in physical files. To improve the clinic's efficiency and flexibility, a study on appointment management systems was developed. This system allows clients to book and manage appointments online, providing access to services anytime and anywhere. Clients can check available time slots, choose a suitable time, and eliminate the need for multiple visits or

waiting in queues. The system includes an online alert feature to notify clients of any changes to their appointments, and it helps mothers track their children's immunization schedules and pregnant women manage their clinical appointments. Overall, the system is expected to improve the clinic's services and efficiency.

A system for booking doctor appointments and receiving medical advice and prescriptions from preferred doctors called the Online E-Healthcare Online Doctor Appointment Booking System, was developed by Agrawal (2022). Users can search for specialists based on their disease and sign in or out of the website. The doctor's information, including contact number and specialty, is available on the website. Patients can view a doctor's schedule and select an appointment based on availability. The doctor's dashboard displays the number of patients seen, appointments booked, and daily earnings. They can also manage appointments, view upcoming meetings, and check the schedule. The admin manages patient records and tracks the number of patients, appointments, and earnings for all doctors. They can also view which doctors are present in the hospital and manage their schedules and availability. Furthermore, the admin can see information like which doctor is treating which patient and whether the patient has paid their fees. The system has system security to protect hardware, software, data, procedures, and people from unauthorized access and natural disasters. System security is categorized into four areas: Security, which involves protecting against unauthorized access and attacks, Integrity, which refers to maintaining the accuracy and completeness of data, Privacy, which involves protecting personal information from being disclosed, and Confidentiality, which refers to protecting sensitive information from being accessed by unauthorized individuals.

Furthermore, Guhma (2018) created an online appointment booking system called "zer0Clock" to solve the problem of long queues in Ghana's hospitals and public-sector offices. The system allows individuals and businesses to register and schedule appointments to avoid long queues. It generates an event calendar for registered users to schedule events, and administrative users can customize their company's website and view statistical analysis of bookings and feedback. The system has a video and live chat function for virtual appointments, and a notification system that reminds clients of their appointments. Clients can schedule, cancel, or reschedule appointments on the calendar, and receive reminders 30 minutes before their appointment. The application also includes Google Map functionality to help clients navigate venues. With the growing use of the internet in Africa over the past decade, the application has the potential to significantly reduce long queues or eliminate them.

2.2 Management of Hospital Appointment Systems

The current healthcare system is laborious as it involves the manual recording of data on paper which needs to be transmitted to multiple departments, making it susceptible to human error (Sharma, 2020). To address these issues, a system was designed to manage doctor-patient interactions, which can assist physicians in their work and allow patients to schedule appointments and keep track of their health. The system is compatible with various user devices, including smartphones, tablets, laptops, and computers. Doctors can use the system to manage their online appointment scheduling, and patients can reserve appointments online using their names. After each appointment, doctors enter the patient's medical information into the database, and patients can view their entire medical history as

required. To develop the system, various tools such as Javascript, HTML, JSP, and Microsoft SQL Server Database 2014 were utilized.

Moreover, Idowu et al. (2014) developed an online system for booking medical appointments, which allows patients to book their appointments. The system was built using Dreamweaver and PHP as a scripting language to manipulate the data in the database, which was stored using MySQL on the WampServer. Patients must create an account first to access the activity menu, which includes the option to schedule an appointment based on the availability of doctors on the weekly calendar. The system also has a report option that allows patients to view the progress of their appointments and medical records. The administrator is responsible for editing, inserting, deleting, and reviewing the results for all enrolled outpatients. The system simplifies the task of generating reports for medical professionals, saving them time and effort.

The research conducted by Khan and Karim (2020) presented a smart e-health system for the Covid-19 pandemic, consisting of several features such as video calling, a diagnosis system, an appointment system, a blog section, and online prescriptions. Patients can make online appointments using the web app appointment system and search for the appropriate specialist to view the available doctors at a given time. Once the patient has made an appointment, an appointment table on the doctor's dashboard displays an accept button. When the doctor approves the appointment, a notification is sent to the patient's notification bar. The front end of the project was constructed using HTML5, CSS3, and the bootstrap framework, while the back end was built using the Django rest API, which uses the Python-based framework's simple coding, low coupling, and speedy development to facilitate building intricate, database-driven websites. The system not only streamlines

the process of making medical appointments for patients but also minimizes contact during the pandemic.

A web-based project was created to improve patient appointments, registration, and diagnosis at a dermatology clinic in a teaching hospital in Ragama, according to Hewasinghe (2019). The system allows patients to schedule appointments via SMS or email and informs them if the clinic will not be held on a particular day. The system provides efficient patient registration and enables clinical staff to schedule patients easily. The system also provides patient prescriptions to the pharmacist to avoid rushing into the hospital pharmacy. The system is authorized for different user categories and can be accessed remotely via the internet or mobile app. It is portable and user-friendly and can be easily implemented in other hospitals. The system uses a WAMP server (which includes Windows, Apache, MySQL, and PHP) and includes a template for dermatologists to enter diagnostic information. It has a user-friendly interface, is portable, and includes email and basic message notification services provided by AWS and ASN thus allowing reports to be sent to other hospitals for patient referrals and be utilized in dermatology clinics in other hospitals with identical diagnostic templates.

Also, in a study conducted by Akshay et al. (2019), Bookazor is a web-based tool that enables users to book appointments with salons, hospitals, and architects in a certain geographic area. The system is developed using an ionic framework, an open-source SDK for hybrid mobile app development, and utilizes CSS, HTML, and JavaScript. Firebase is integrated into the application to provide functionalities such as analytics, messaging, crash reporting, and database management to enhance the development process. The NodeJS technology is used to store and manage appointment requests, which include a sequence of

regions to be visited by the user. The NodeJS server is responsible for scheduling appointments at specific times, checking operative availability in designated regions, and updating the routes to reflect booked appointments. Additionally, the scheduler updates the routes periodically to prevent duplication and generate a new set of appointments.

In another study by Osunade et al. (2014), a mobile application was created for Android platform users to manage their time. The application was developed using two Google Application Programming Interfaces (APIs): the Google Maps API and the Google Calendar API, with additional programming in Java. The application allows users to add appointments and events, which are stored in the phone's calendar and synced with the Google calendar. Users are alerted at a preset time before their appointment. It also includes a Google Map, which helps users view the location of the appointment beforehand, reducing the time it takes to get there. The user enters appointment details through the user interface, which are then stored by the Appointment Manager App. The application sends the date and time of the appointment to the Calendar API and the location of the appointment to the Map API. The Latitude API obtains the user's coordinates, which are then used to load the corresponding map on the user interface.

2.3 Application of Time Series Analysis and Forecasting

According to Raghupathi & Raghupathi (2014), big data in healthcare could arrive from both inward (e.g., electronic health records, clinical decision support systems) and outward (state authorities, laboratories, pharmacies, insurance companies, health maintenance organizations) sources. They are often in various formats (flat files,.csv, relational tables, ASCII/text) and lodged in numerous places.

Verma et al. (2020) noted that forecasting is one of the duties in the field of data science that aids companies to effectively distribute resources for maximum gains and to reach demands in their highest points. It assists in setting objectives and measurement of performance over time to offer insight for additional enhancements. Progressive developments in the field of machine learning and AI has allowed forecasting to be less laborious and time intensive.

In research from Fang et al. (2022), the eXtreme Gradient Boosting (XGBoost) model is a scalable machine learning algorithm that is based on decision trees. It is able to handle time series data that are not linear due to its powerful self-learning feature. Accurate predictions are done with an internal algorithm that merges results from several trees. XGBoost is capable of indicating feature importances, preventing overfitting, handling missing values, and parallel computing for decreased computing duration. The XGBoost model was utilized for predicting COVID-19 in the USA, and was compared with ARIMA on daily COVID-19 data from December 13, 2020, to June 30, 2021. The training set started from December 13, 2020, to June 16, 2021, while the testing set started from June 17, 2021, to June 30, 2021. The study used 3 performance metrics, namely, mean absolute error (MAE), root mean squared error (RMSE), and mean absolute percentage error (MAPE). MAE indicated the mean absolute errors between the actual and predicted values. RMSE indicates the square root of error on average, and is used to check the difference between the predicted values and the actual values. Finally, MAPE shows the average error between the actual and predicted values in percentage form. The forecasting results revealed that the fitted XGBoost model was better than the ARIMA model. The ARIMA model's training set metric scores were 7061.536 MAE, 13517.664 RMSE, and 7.996%

MAPE. For the test set metric scores, it yielded 2083.571 MAE, 2633.424 RMSE, and 15.884% MAPE. The XGBoost model's training set metric scores are as follows: 2331.134 MAE, 3500.331 RMSE, 4.096% MAPE. Finally, the test set metric scores for the latter model yielded 962.357 MAE, 1209.984 RMSE, and 7.892% MAPE. It can clearly be concluded that the XGBoost model significantly performed better across all metric scores than the traditional ARIMA model.

A similar study was conducted by Alim et al. (2020) as they attempted to predict human brucellosis cases in mainland China. The data came from the National Health and Family Planning Commission of China, wherein the monthly time series data spanned from January 2008 to June 2019. January 2008 to June 2018 was assigned as the training set, while July 2018 to June 2019 was assigned as the testing set. MAE, RMSE, and MAPE were the performance metrics utilized to evaluate and compare the ARIMA and XGBoost model. The ARIMA model's training set metric scores are as follows: 338.867 MAE, 450.223 RMSE, and 10.323% MAPE. The testing set metric scores yielded 529.406 MAE, 586.059 RMSE, and 17.676% MAPE. The XGBoost model's training set metric scores are the following: 189.332 MAE, 262.458 RMSE, and 4.475% MAPE. As for the testing set metric scores, it displayed 249.307 MAE, 280.645 RMSE, and 7.643% MAPE. Across all metric scores, the XGBoost model outperformed the traditional ARIMA model.

Cheng (2020) stated that outliers usually occur in real-life applications and in time series data. One way to solve this is through trimming, where some values are set aside in one tail, or in both tails. However, omitting values causes some information to be lost. Winsorization is a way to assign extreme values to stay within a certain threshold/cutoff value without omitting data; it prevents loss of information as the sample size does not

change. The challenge with winsorization is specifying the cutoff value, which is usually found by reducing the value of a utilized loss function. If the distribution of data has symmetric tails, symmetric winsorization may be performed. However, if the data is skewed (has asymmetric tails), a one-sided winsorization can be conducted.

2.4 System Evaluation Criteria

'ISAKAY' which is an android-based booking system for TriBike operators and drivers association was developed by Balba et al. (2019) with Cloud-Based Data Analytics. The system's acceptability was evaluated using ISO 25010:2011 standard starting from the use of the system on the administration side (the operator), the drivers, the users and the IT practitioner's side. The evaluation involved testing the system's performance efficiency, compatibility, usability, reliability, security, maintainability, and portability. The results of the performance efficiency test showed a score of 4.58, indicating that the system is highly acceptable in terms of time behavior, resource behavior, and capacity. The compatibility test yielded a score of 4.64, demonstrating that the system can communicate with other systems while sharing the same hardware or software environment. The usability test resulted in a score of 4.73, indicating that the system is user-friendly and easy to use. The reliability test produced a score of 4.68, meaning that the system is mature and can recover or resume after failure. The security test resulted in a score of 4.66, indicating that the system is secure and can maintain integrity, confidentiality, nonrepudiation, accountability, and authenticity. The maintainability test yielded a score of 4.62, demonstrating that the system is stable and easy to modify and test. Finally, the portability test resulted in a score of 4.69, showing that the system can be easily moved and replaced by other software. The

overall mean score of all the tests was 4.64, indicating that the system is highly acceptable and has been well-received by users as a new booking system in Odiongan, Romblon.

In a study by Soini (2021), a framework was presented for implementing new appointment scheduling and virtual meeting systems with various features including agent and customer flows, access and channel roles, user interfaces, core functionalities, and architectures. The study employed the ISO 25010 model for ensuring high-quality non-functional requirements of the system, which was created by the International Organization for Standardization (ISO). The ISO 25010 model consists of eight quality characteristics, namely Functional Suitability, Performance Efficiency, Compatibility, Usability, Reliability, Security, Maintainability, and Portability. The Functional Suitability requirements involve integrating with O365 calendars and managing user access, while Performance Efficiency requires a fast response time without slowing down other systems. Compatibility entails handling many users and events, and Usability demands that the system should always be available and easily restored from backups. Reliability requires disallowing erroneous input, proving events and activities, and regularly backing up data. Security needs to confirm modified solution functionality, have access control, and test automation. Maintainability involves keeping logs of logins, data changes, and ID browsing, and being easy to modify and develop. Lastly, Portability should enable moving the solution to another environment. The study confirmed the security features at the end of the implementation phase, while Maintainability was tested during production use, and Portability was necessary for transferring the systems to another vendor.

In addition, Batoon et al. (2022) developed a web-based system aimed at enhancing the quality of maternal healthcare services and working conditions in a maternity clinic.

The quality of the developed system was evaluated using ISO 25010, which comprised six criteria: functional suitability, performance efficiency, usability, reliability, security, and maintainability. Functional suitability is concerned with the system's ability to assist in the completion of tasks and objectives. Performance efficiency evaluates resource utilization in terms of the system fulfilling requirements. Usability measures how easily and efficiently users can complete tasks. Reliability evaluates the system's ability to perform functions successfully in a specific environment. Security measures the system's ability to secure important data, adhere to user access levels, and prevent unauthorized access. Maintainability measures how well the system can adapt to environmental and requirement changes. Sub-criteria for each criterion include functional appropriateness, resource utilization, learnability, maturity and availability, confidentiality, integrity and authenticity, and reusability and testability.

2.5 Synthesis

The related studies have provided a variety of results and shall become the basis of creating the features and process of the appointment management system. These pieces of literature used many kinds of tools for system development and implemented different methodologies in building their structures. Online scheduling has been deemed important to solve healthcare organizations' difficulties such as the stacking of records on paper and patients who are not attending appointments in the hospital. The related research has also shown that time series analysis is applicable in healthcare to forecast patient arrivals. Integrating these studies will guide the development of an appointment management system with the application of data analytics.

The researchers will use numerous tools to build the overall composition of the web application. JavaScript, HTML, CSS, and the Bootstrap framework will be used for the system's front end to create a responsive appointment management web design for patients, doctors, and administrators. MySQL will serve as the database management of the system that will store the users' information. Django and Python will also be used for the system's back end to build the website. Data analytics will be applied using time series forecasting with XGBoost. Once the XGBoost model has produced considerable results, it will be used to forecast daily patient arrivals which will help in the hospital's resource allocation, strategic planning, and to offer information and insights.

2.6 Conceptual Framework

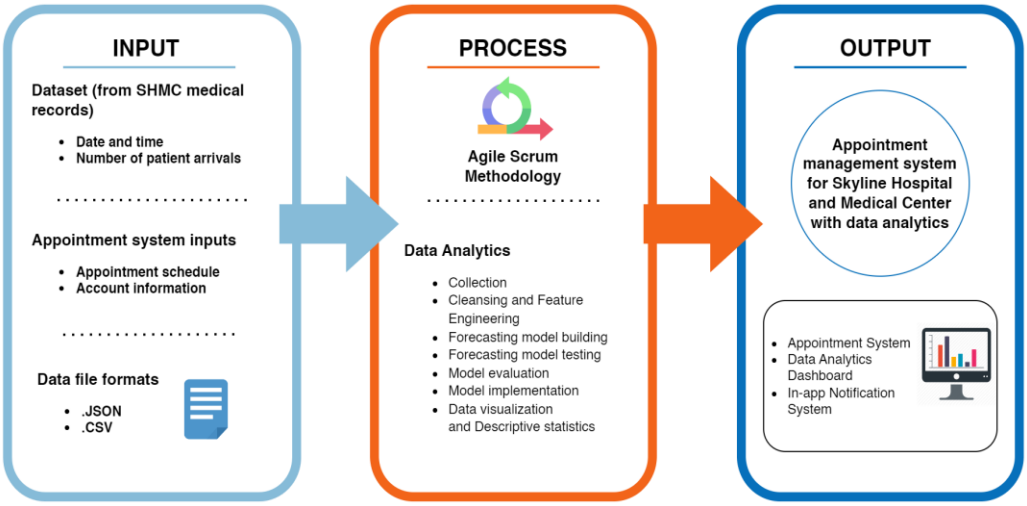


Figure 2.1: Conceptual framework of the study

Figure 2.1 is an input, process and output model, or IPO model that represents the conceptual framework of the study. The IPO model is the appropriate model for visualizing the development processes as it states the concept inputs, process and output. The inputs for the system are: the dataset from SHMC patient arrival records; and the user inputs such as booking schedule and basic account information. For the process, the researchers will

use the agile scrum methodology in developing the system. The development process will start with identifying all the features that need to be implemented in the system and choosing the tasks that have the highest priority. The developers will only focus on these tasks during the sprint. The process will iterate over and over until the final deliverable is achieved. The data analytics process consists of data collection that comes from the SHMC database, which is then cleansed and feature engineered when it is transformed into a time series dataset. Feature selection is also important as when dealing with time series data needs time series related features (e.g. day of the year, day of the week, the year in which the day belongs, etc.) Data visualization is implemented as the process of the data analytics goes on. Python modules for visualization such as `matplotlib`, combined with `seaborn`, would be essential in enhancing the aesthetics and interpretability of the generated graphs. The dataset is split into training and test sets wherein the model will be trained for forecasting in the training set, then testing and assessment with the test set. After that, the model undergoes evaluation with error metrics such as mean absolute percentage error (MAPE), root mean squared error (RMSE), and symmetric mean absolute percentage error (sMAPE). It is then implemented and published in the data analytics dashboard (viewable only by administrator users) in the appointment management system for insights and information. The output for the study is the Appointment Management System with data analytics for Skyline Hospital and Medical Center. The system will have an appointment system for the doctors and patients. The system will also have a data analytics dashboard that can be viewed by the administrators, and an in-app notification system.

Chapter III

RESEARCH DESIGN AND METHODOLOGY

This chapter specifies the research design, the software development method, the various algorithms utilized for the system, and the statistical tools applied for data gathering.

3.1 Research Design

In this study, the researchers utilized a descriptive research design to gather information about a particular issue and address questions such as what, who, where, when, and how. This type of research design is typically used to investigate the current state of a phenomenon (Akhtar, 2016). The study employed a descriptive research design to evaluate the appointment management system's performance in terms of scheduling and appointments, with the incorporation of time series forecasting.

3.2 Project Development

The project development methodology used by the researchers is the Scrum methodology. It is a method in Agile development that is based on iterative and incremental processes. It implements the “pull” system which controls the flow of the work. In Scrum methodology, the project is split into manageable tasks that can be finished within the time period set called “sprint” (Hidalgo, 2019). Scrum sprint usually takes around 1 to 2 weeks or up to a month in duration. The Scrum methodology helps complete and deliver outputs quickly and effectively every increment to the stakeholders at the end of every scrum cycle.

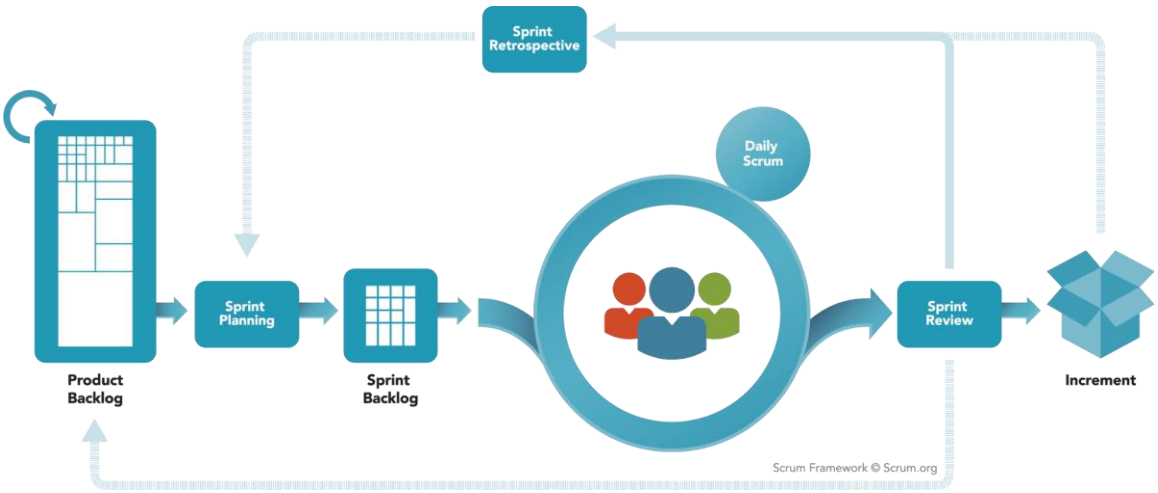


Figure 3.1: The Scrum Framework (Scrum.org, 2020)

Figure 3.1 shows the general process of each sprint. For each sprint, high-priority tasks are selected from the product backlog during the sprint planning meeting that the developers can do and deliver to the customers. These selected tasks are called the "sprint backlog," which the developers will only work on for the duration of the sprint. The daily scrum is a short meeting that is held every day. It usually takes about 10 to 15 minutes to discuss the current progress of the sprint, identify and resolve development challenges and adjustments, and plan for the next 24 hours. At the end of each sprint, a sprint review is held to discuss what the development team accomplished, deliver the sprint output, and have the stakeholders examine the output and give feedback on the product.

Application Language and Framework. The Python programming language will serve as the main scripting language in developing the application logic and using the Django web framework for handling tasks such as handling the response and request from and to the client and web server, and connecting databases. Python is one of the most popular programming languages along with Ruby, Pearl, and many others (McKinney,

2017), and has a large number of libraries for creating applications and models, for machine learning, data science, data analysis, and many more.

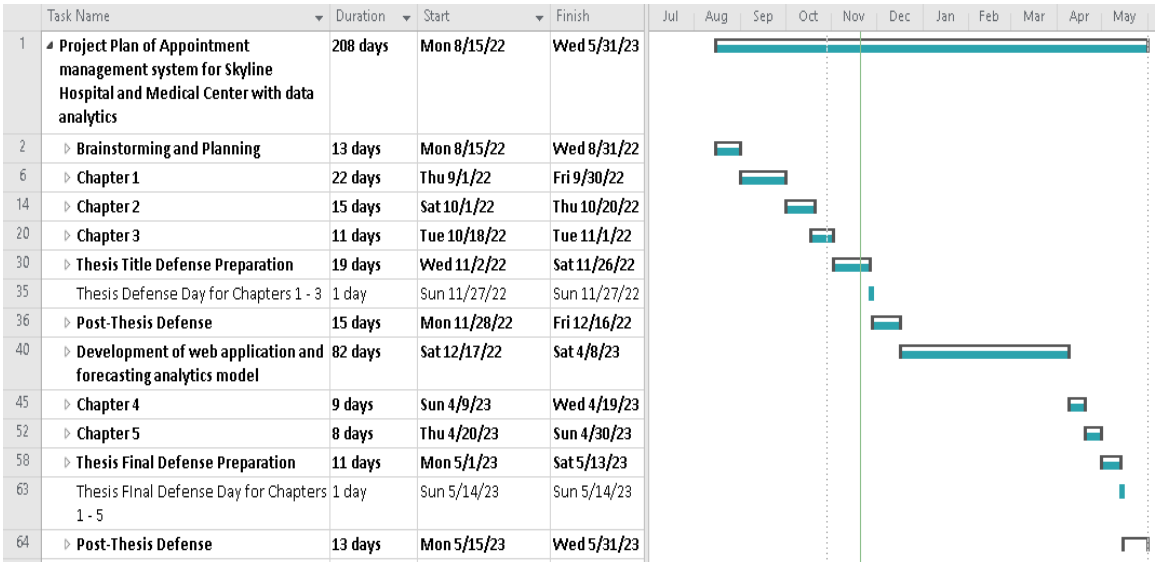
User Interface. The program will have three different interfaces based on the type of account. These are the admin's, doctor's, and patient's interfaces. The administrator will have the majority of the application's control. The admins can view the dashboard and reports, manage the doctors and patients accounts, and accept or reject an appointment request. The doctor can view his or her scheduled appointments that have already been accepted by the admin and set his or her available schedule for appointments. The patient can only book an appointment through the app and view all of his or her scheduled appointments.

Appointment System. The appointment system will be responsible for handling the appointments that were made by the patient. The system will dynamically generate the form based on the inputs from the patient, and it will be the one who picks a doctor for the patient if they do not have a preference. And it allows the patient to book appointments at a convenient time and ensures that they receive timely medical care.

Database Management System. The database will be used for storing and managing all the data that was already saved and any incoming data that needs to be saved in the system. PostgreSQL will be used as the database management for the appointment management system. It is a high-performance relational database system for managing databases that is highly compatible with the Django web framework.

3.3 Project Planning and Management

A Gantt chart was utilized to represent the project planning schedule. It is excellent for planning task deadlines and monitoring.



	Task Name	Duration	Start	Finish
1	Project Plan of Appointment management system for Skyline Hospital and Medical Center with data analytics	208 days	Mon 8/15/22	Wed 5/31/23
2	Brainstorming and Planning	13 days	Mon 8/15/22	Wed 8/31/22
6	Chapter 1	22 days	Thu 9/1/22	Fri 9/30/22
14	Chapter 2	15 days	Sat 10/1/22	Thu 10/20/22
20	Chapter 3	11 days	Tue 10/18/22	Tue 11/1/22
30	Thesis Title Defense Preparation	19 days	Wed 11/2/22	Sat 11/26/22
35	Thesis Defense Day for Chapters 1 - 3	1 day	Sun 11/27/22	Sun 11/27/22
36	Post-Thesis Defense	15 days	Mon 11/28/22	Fri 12/16/22
40	Development of web application and forecasting analytics model	82 days	Sat 12/17/22	Sat 4/8/23
45	Chapter 4	9 days	Sun 4/9/23	Wed 4/19/23
52	Chapter 5	8 days	Thu 4/20/23	Sun 4/30/23
58	Thesis Final Defense Preparation	11 days	Mon 5/1/23	Sat 5/13/23
63	Thesis FInal Defense Day for Chapters 1 - 5	1 day	Sun 5/14/23	Sun 5/14/23
64	Post-Thesis Defense	13 days	Mon 5/15/23	Wed 5/31/23

Figure 3.2: Gantt chart of the project plan

Figure 3.2 displays the task outline and the visualization for the Gantt chart prepared by the researchers. The task outline reveals the task names, the duration for each task, the start times for every task, and their end times. The Gantt chart visualization displays the date and time at the top of the chart, whereas the highlighted horizontal bars indicate the duration of each task based on the task outline. The Gantt chart in this research is utilized to track and monitor each and every task in the project. The appointment management system itself is a rigorous task, and applying data analytics with time series forecasting can make it even more complex and time consuming. It is essential that the Gantt chart would be strictly followed in order to avoid chasing deadlines.

3.4 Software Requirements

Figure 3.3: Bootstrap

The researchers used Bootstrap, which is a JavaScript library, to integrate a responsive web site in the appointment system that also includes HTML and CSS. Bootstrap can also be used in JavaScript Plugins.

Figure 3.4: Django

Django is a high level web framework to build secure and maintainable websites. The researchers used this component to focus on developing the complete source code and database of the python.

Figure 3.5: PostgreSQL

PostgreSQL is a powerful, open source object-relational database system that supports both SQL and JSON querying. The researchers used this as the primary data

storage or data warehouse that safely stores and scales the most complicated data workloads.

Figure 3.6: Django REST Framework

Django is a flexible toolkit for building Web APIs. The researchers used this to transfer information between an interface and a database which makes serialization much easier.

Figure 3.7: Vue.js

Vue.js is a JavaScript framework that provides built-in directives and user defined directives. The researchers used this programming language for building user interfaces that provides a declarative and component-based programming model that helps effectively develop user interfaces.

Figure 3.8: JupyterLab

JupyterLab is a programming environment used for conducting data analytics, machine learning, model building, and other data science tasks. The researchers used this programming environment to build the time series forecasting model for future patient arrivals.

Figure 3.9: HTML, CSS, JavaScript

HTML is a markup language to design web pages that will display in the webpages. CSS is responsible for creating the style and layout of the webpages. JavaScript is used to create a block of code to design the system's web page and to perform particular tasks on the system. These were the core languages used in making the frontend or the client side of the system that is utilized by doctor users and patient users.

Figure 3.10: Python

Python is a high level programming language that helps the researcher to integrate the components of the system's functionality. It is also used to develop the time series forecasting model to be used into the system.

Figure 3.11: Visual Studio Code

Visual Studio Code is the code or text editor to be used for coding and building the appointment management web application system with the languages mentioned above such as Python, HTML, and JavaScript. Code that is used with the Vue.js framework for the frontend or client side, and the Django REST framework for the backend side is also developed using this editor.

3.5 System Flowchart

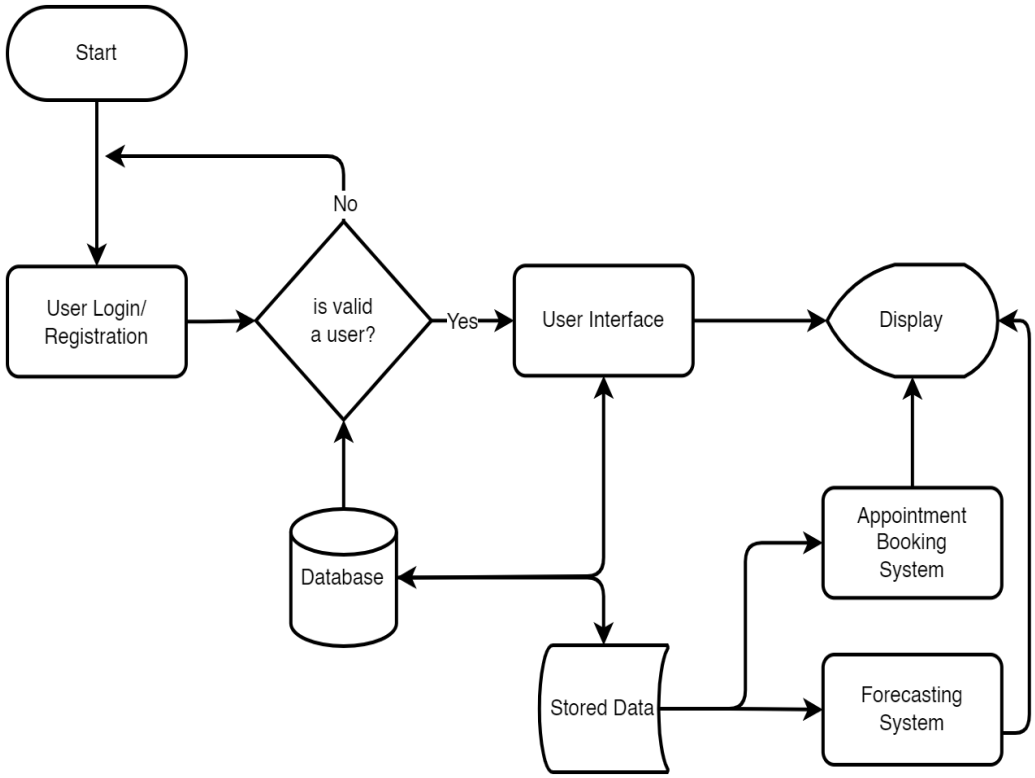


Figure 3.12: High-level Flowchart

Figure 3.12 depicts the high-level layout of the Web-Based Application System. The flowchart depicts the primary systems, which begin with the user registration and login page, where users sign in or establish accounts. The verification is performed by determining whether the inputs exist in the database. All information entered through the user interface is kept in the database and validated by the administrators. After administrators have validated this information, the recorded data will be entered into the systems and processed accordingly. The systems are subject to data analytics, in which each dataset is examined and analyzed. It comprises a forecasting system that employs machine learning and notification and scheduling tools.

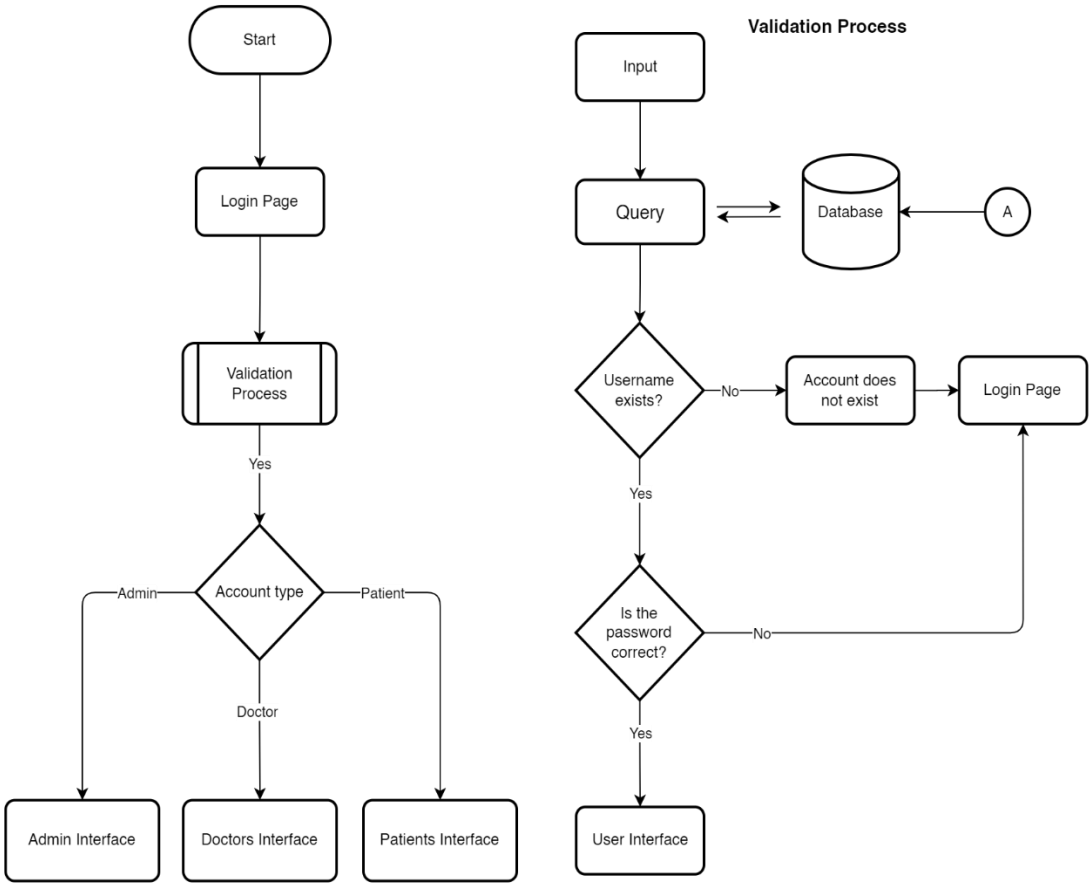


Figure 3.13: Low-level Flowchart - Login Interface

Figure 3.13 shows the low level system flow for the login page. The user needs to enter their username and password. The input will be validated using a simple validation process where the entered username will be searched first in the database, and if the username exists, the password entered will be compared to the one saved in the database. If the username does not exist, it will display an error message and select the username input field. If login is successful, the app will dynamically generate the user interface depending on their account type.

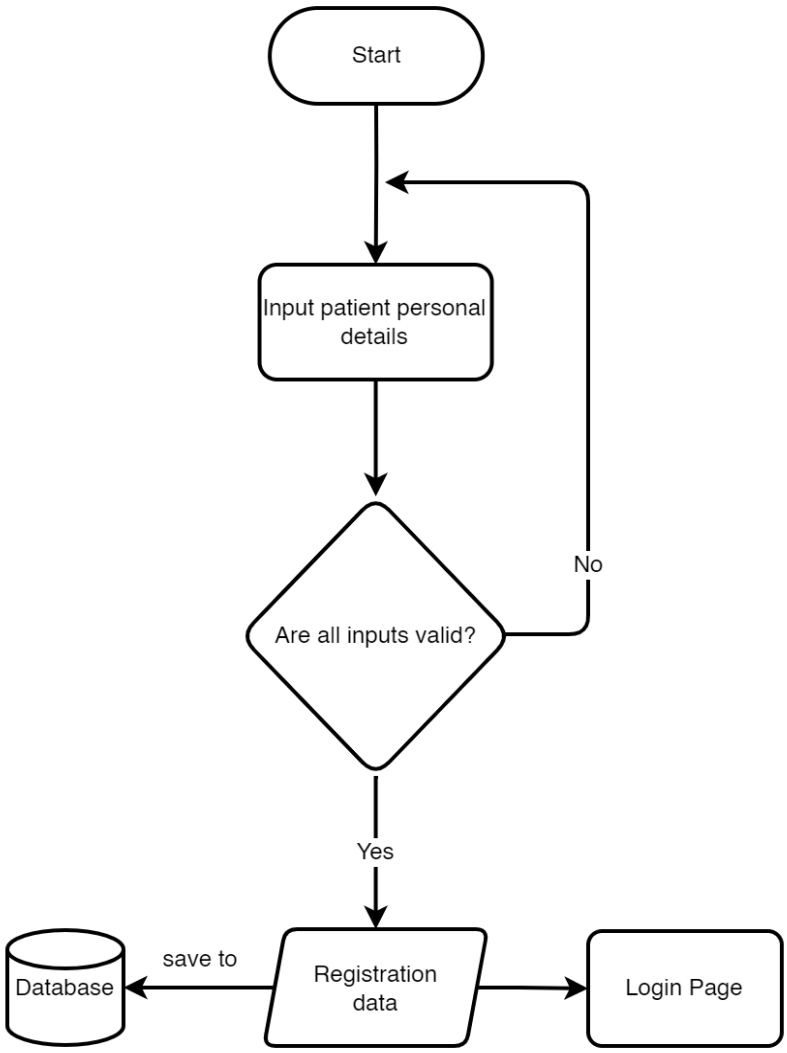


Figure 3.14: Low-level flowchart - Registration Interface

Figure 3.14 shows the low level flowchart of the registration interface. The start of the process will be the user must input the details needed to complete the registration. The system will verify if the user information is valid. After validation the inputs of the user will be stored in the database. The user will be redirected back to the login page after a successful registration.

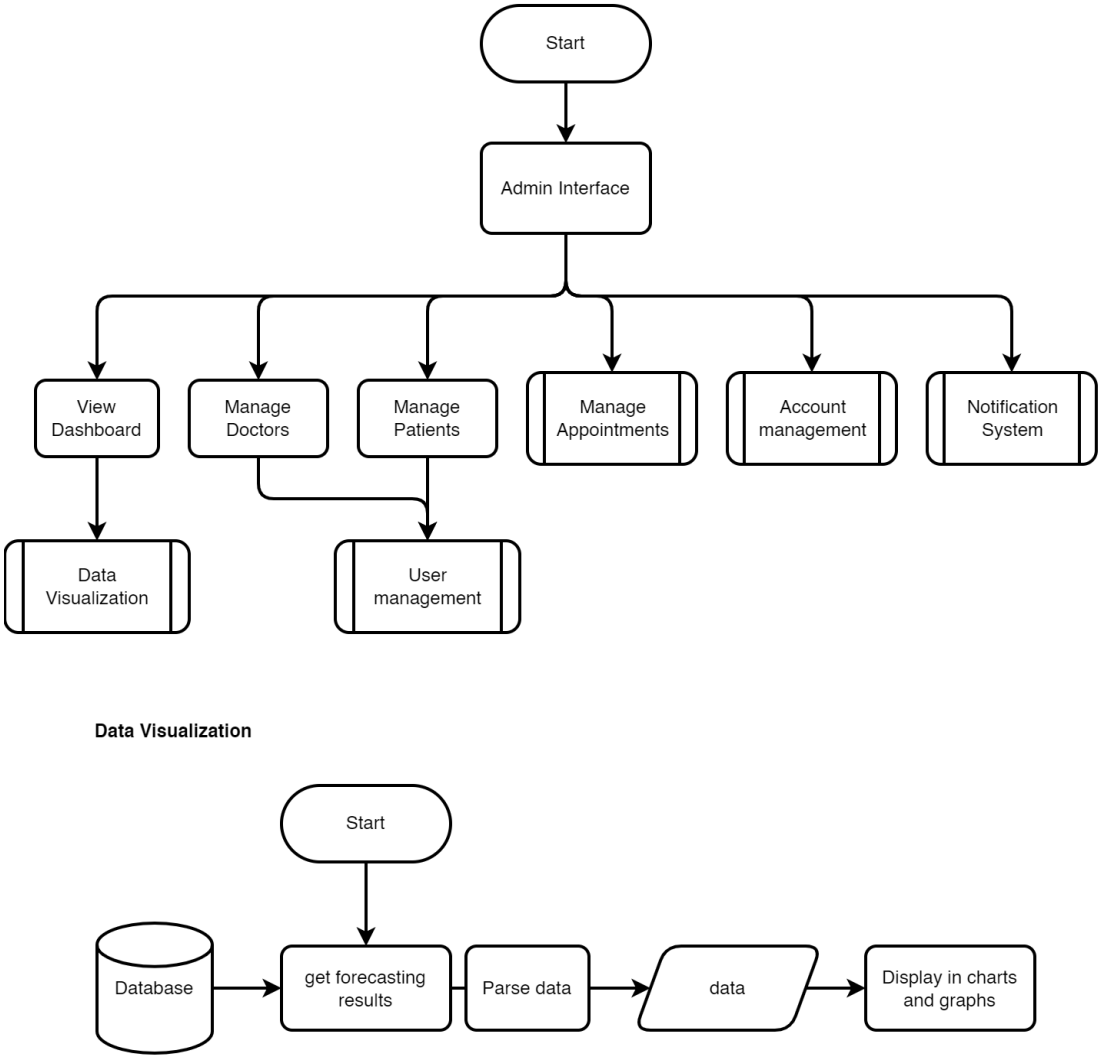


Figure 3.15: Low-level Flowchart - Admin Interface

Figure 3.15 shows the low-level flowchart for the administrator interface. In the admin interface, the administrator can view the dashboard, manage doctor and patient accounts, manage all appointments, change the administrator's account settings, and check notifications. In data visualization, the forecast result is collected from the database and sent to the client side. The result is then parsed so that it can be used and visualized in the administrator dashboard.

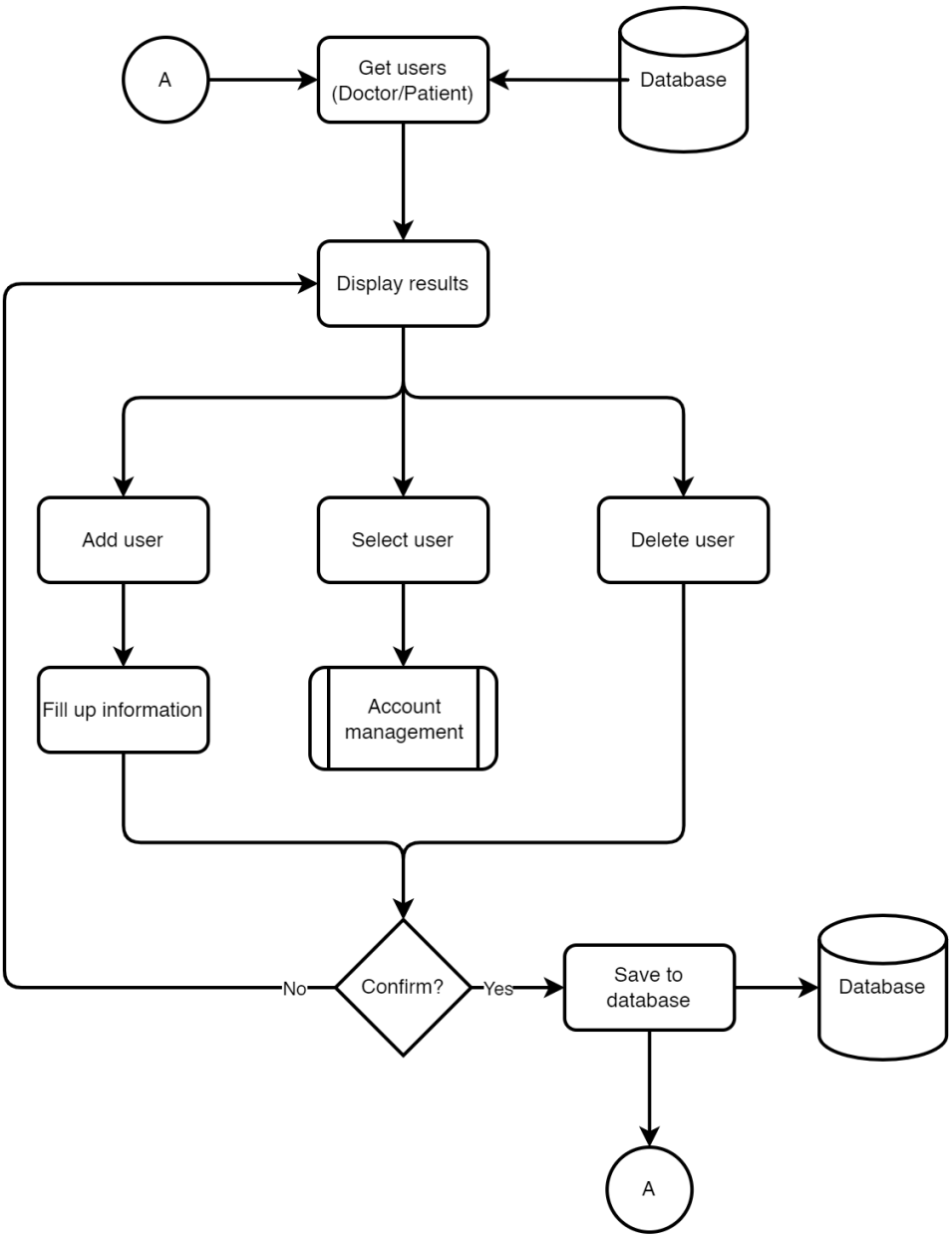


Figure 3.16: Low-level Flowchart - User Management

In the user management, figure 3.16, the list of doctors or patients is collected first from the database and then displayed in a table. From there the administrator can add a user, either a doctor or patient, select a user for managing their account, and delete a user account.

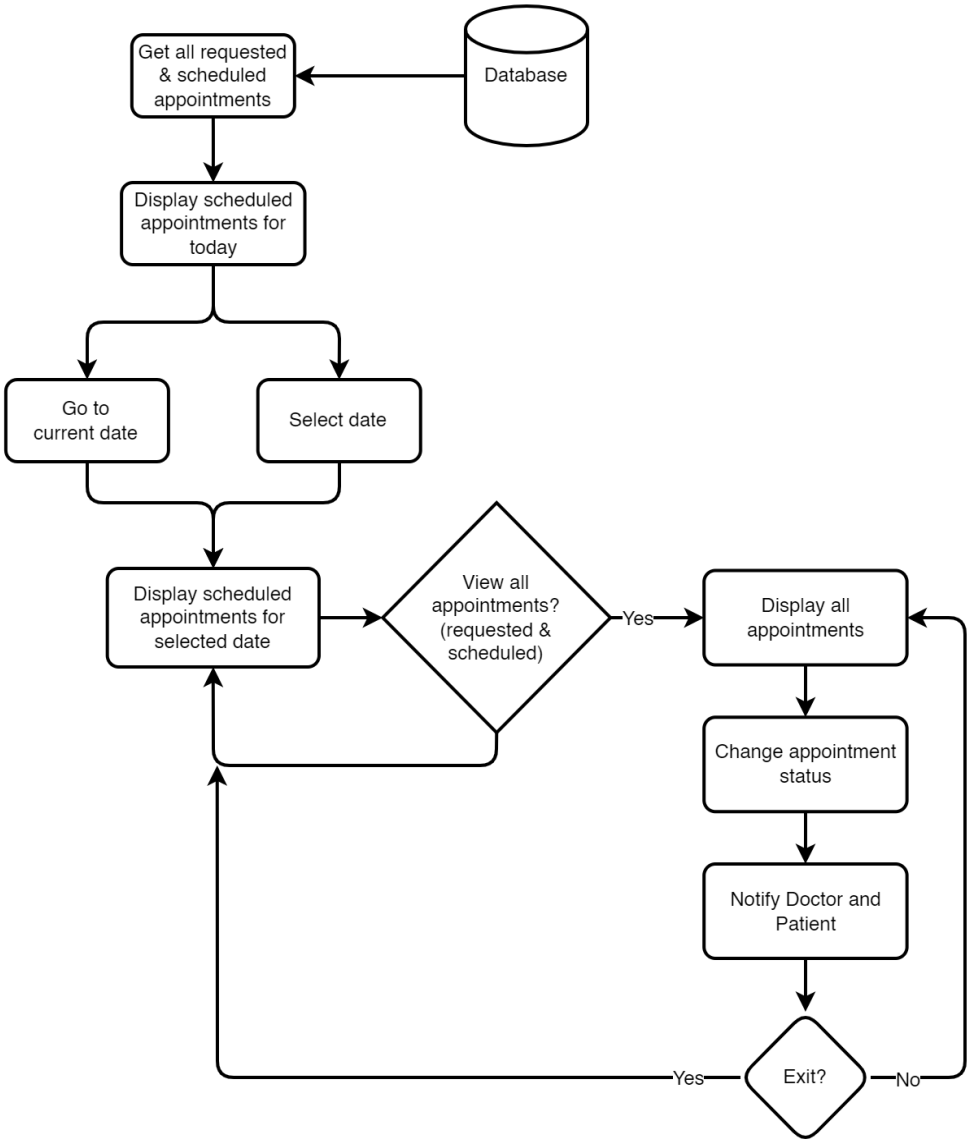


Figure 3.17: Low-level Flowchart - Appointment Management

Figure 3.17 is the flowchart for appointment management in the admin interface. Here, all of the scheduled and requested appointments are collected from the database and displayed by date. The administrator can select a date (by default, today's date is selected) and display all appointments for that day. In the dialog, the administrator can accept or reject requests for appointments. The system will notify the doctor and/or the patient after the admin accepts or rejects the appointment.

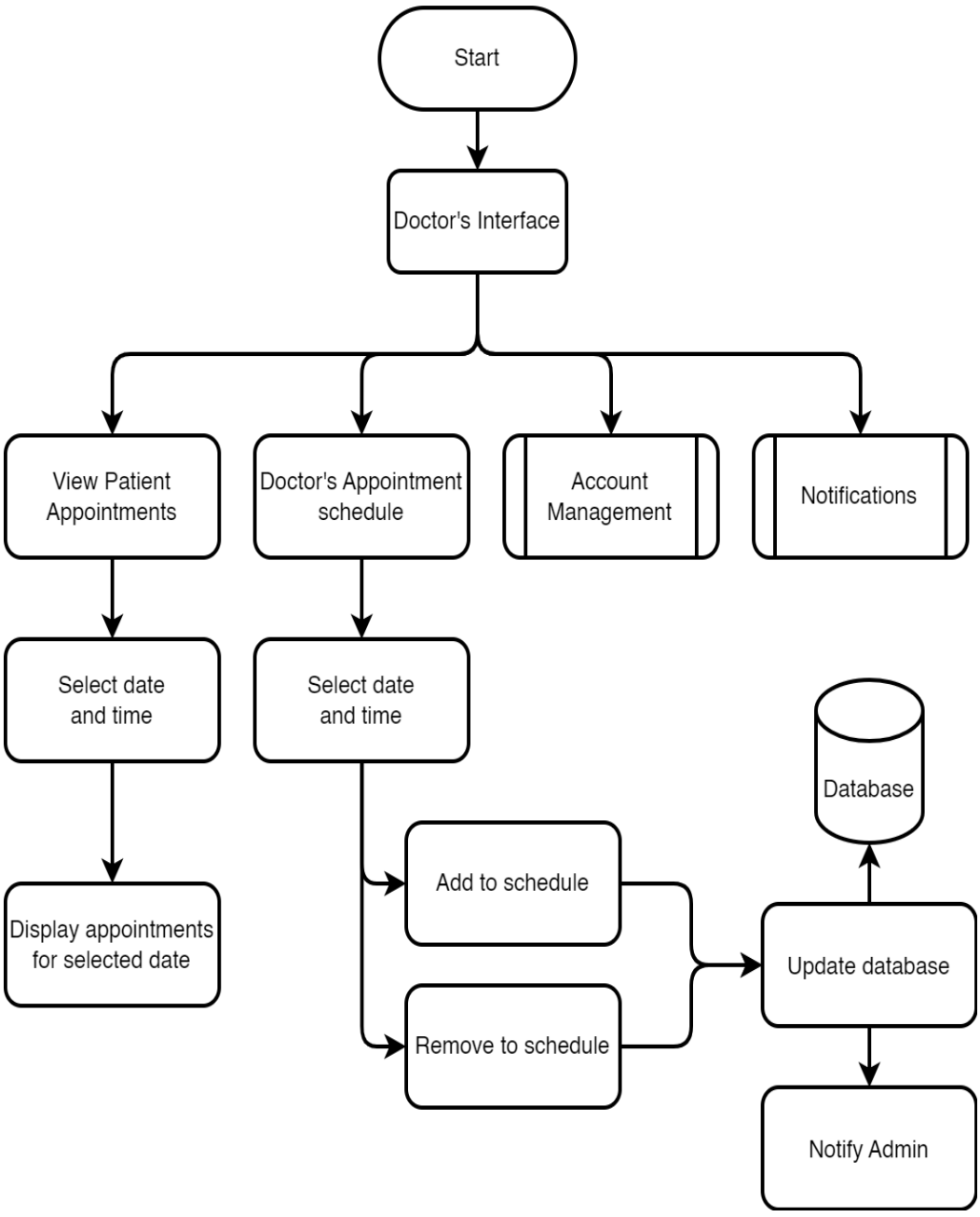


Figure 3.18: Low-level Flowchart - Doctor's Interface

Figure 3.18 presents the low-level flowchart of the doctor's interface and indicates the flow of the appointment system. The doctor can see all of his or her scheduled appointments and also set his or her schedule for appointments. The system will trigger the notification system and notify the admin of any updates from the doctor.

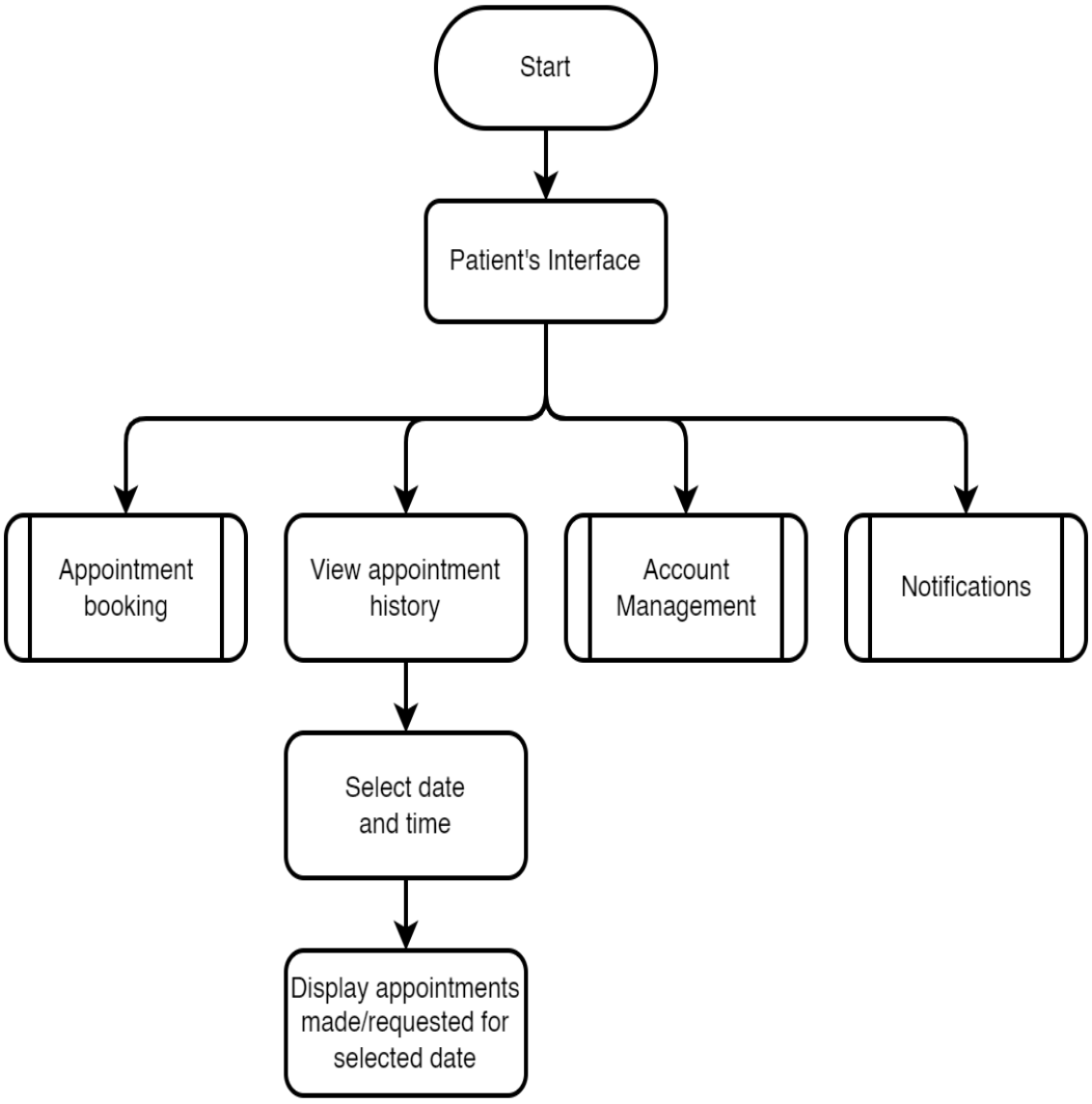


Figure 3.19: Low-level Flowchart - Patient's Interface

Figure 3.19 shows the base flowchart of the patient's interface. Patient users can schedule or book for appointments through the appointment booking feature. They can also view and check their history of appointments made using the system, and select a specific date and time in which they have scheduled for an appointment. For account management, patient users can edit and update profile information when necessary. They can also receive notifications for important announcements.

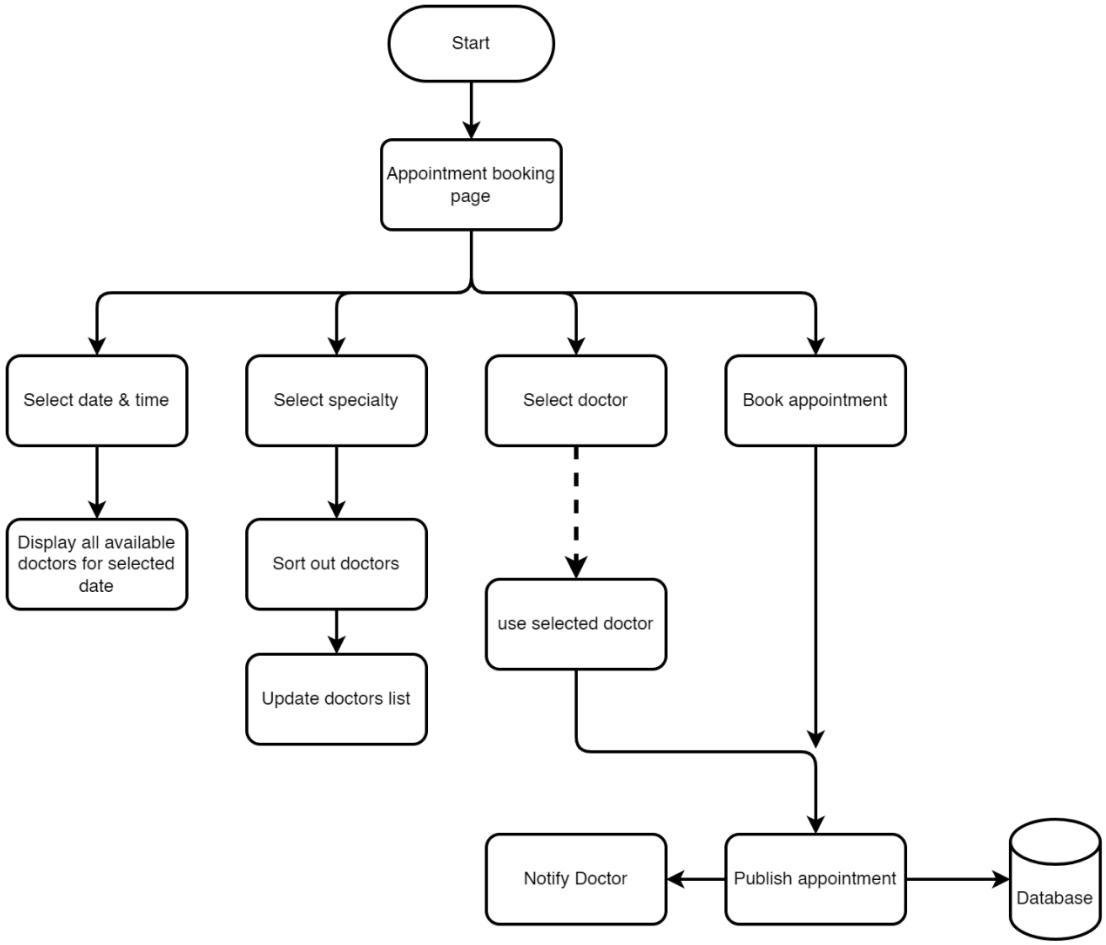


Figure 3.20: Low-level Flowchart - Appointment Scheduling

Figure 3.20 Figure 3.20 shows the system flow when patients book for an appointment. Patient users can select a date and time for the appointment schedule. When the date and time is selected, the available doctors would appear. Patient users can also select the specific service or the specialty to narrow down the list of doctors available. After that, the patient can now book an appointment and fill in additional form details. The appointment will be scheduled with the selected doctor, saves it to the database, and notify the admin for the appointment request; if there are none, the system will choose from available doctors.

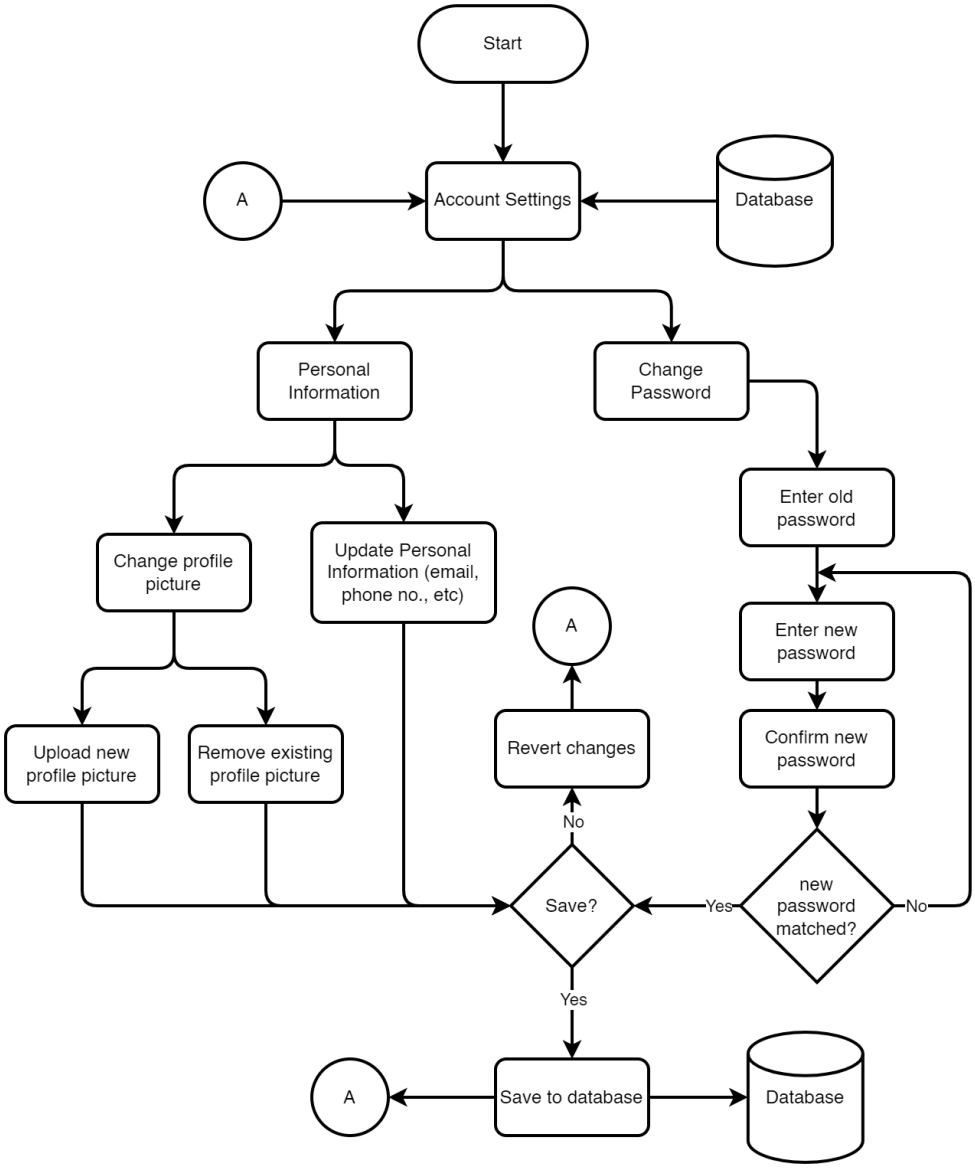


Figure 3.21: Low-level Flowchart - Account Management

Figure 3.21 shows the low level flowchart of the account management interface the system provides each user in the system. Admin, patients and doctors can use the account interface in updating their personal information and passwords. Every user's information will be stored in the database.

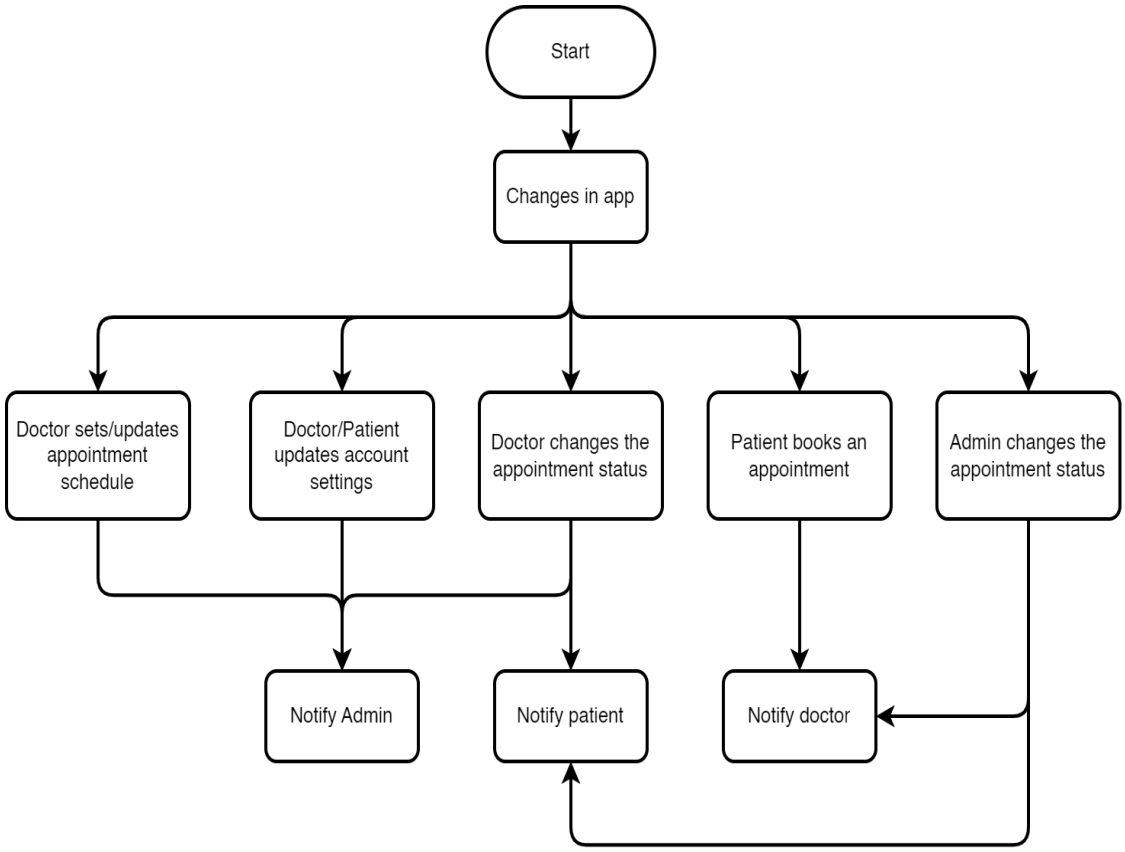


Figure 3.22: Low-level Flowchart - In-App Notifications

Figure 3.22 shows how the in-app notification system will notify the users of any changes or actions in the system. The administrator will receive the notifications: when the doctor sets or updates his or her available schedule; when the doctor or patient updates their account settings; and when a registered patient books an appointment through the application. The doctor will only be notified if the requested appointment is accepted by the administrator, whereas the patient will be notified whether the administrator accepted or rejected their appointment request.

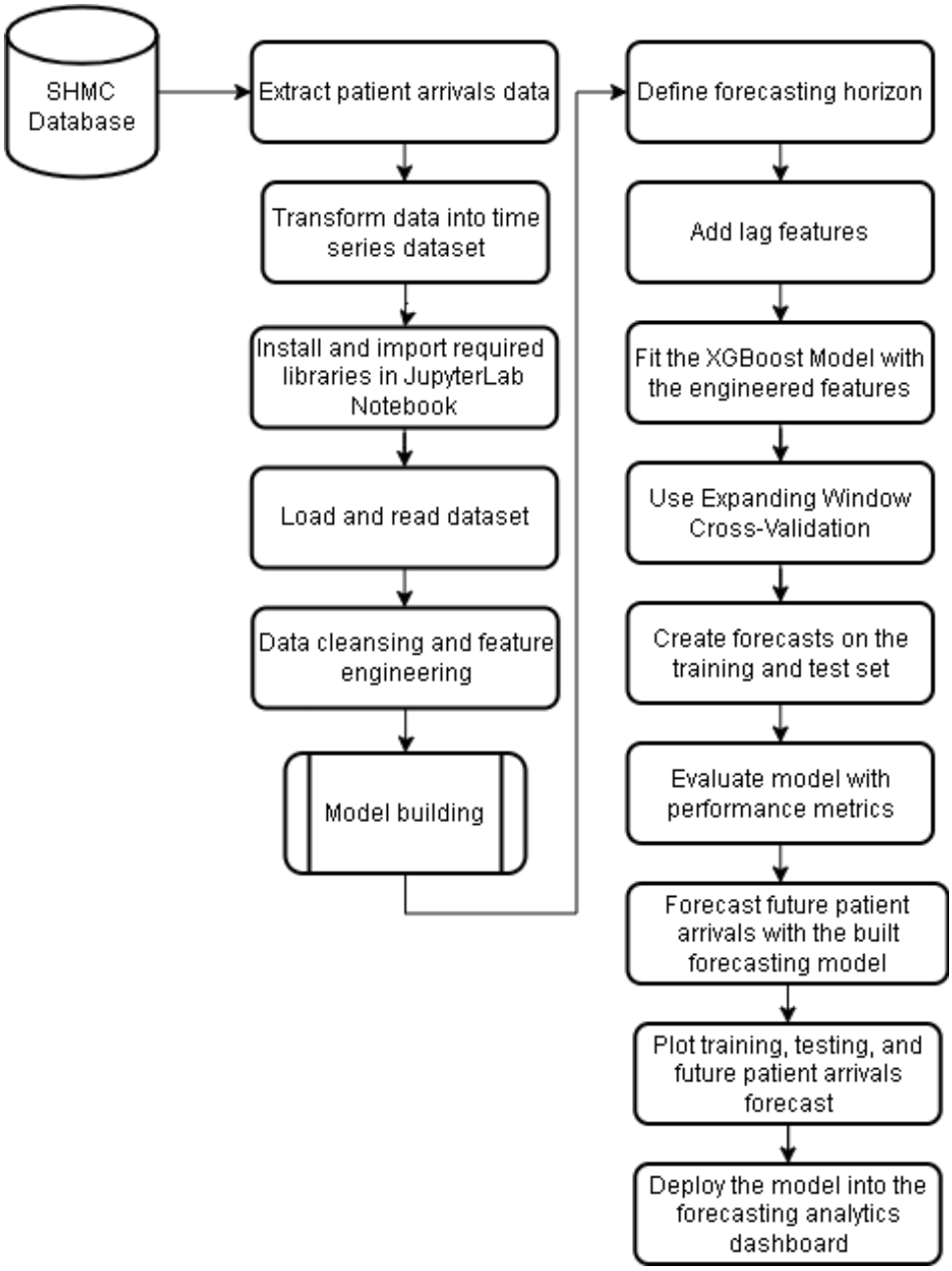


Figure 3.23: Low-level Flowchart - Time series forecasting

Figure 3.23 illustrates the process of time series forecasting in the system. First, data is extracted from SHMC's database, transformed into a time series dataset, and is loaded into a JupyterLab Notebook, which serves as the data analytics workspace. The libraries necessary for data analytics are also installed and imported into the notebook, these include Numpy, Pandas, Scikit-learn, Matplotlib, Seaborn, and XGBoost,. The

dataset is cleansed for attributes that will not be needed and feature engineering occurs, where time series features are derived from the time series data such as the day of the week, the day of the month, the day of the year etc., which will then be used as input features for the model. After the time series features are created, the forecasting horizon is defined with the lag features, which will allow the XGBoost model to be suitable for time series forecasting. An expanding window time series cross-validation is utilized so that the model may be able to learn the temporal structure of the data across the folds. The XGBoost model is then fit into the data with the engineered time series features and lag features. After this, forecasts on the training and test set occur and will then be evaluated using performance metrics. When reasonable results have been acquired in the model's evaluation, forecasting of future patient arrivals is done with visualization. Finally, the forecasting model is deployed in the appointment management system's forecasting analytics dashboard.

3.6 Expected Output

The expected output segment shows the wireframes of the system in the desktop and mobile browser perspective.

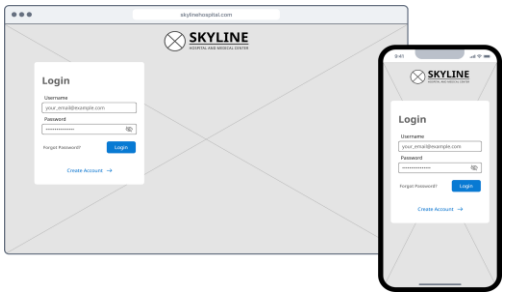


Figure 3.24: Login page

Figure 3.24 shows the login interface for logging in an account. It asks for the account's username or the email address, and the password associated with it to allow

access to the system's main page. The forgot password feature allows users to change their forgotten password. If an account is not yet created, the registration feature will allow the user to register for the system.

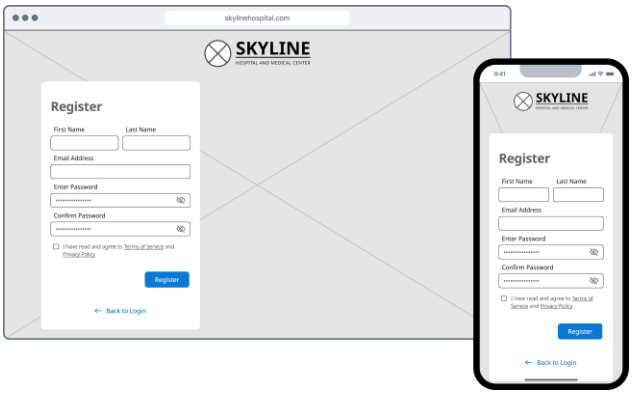


Figure 3.25: Registration page

Figure 3.25 shows the registration page interface. It will ask for the user's first name, last name, e-mail address, and the password to be entered two times to prevent forgetting it at the time of registration. Once registration is complete, the user is redirected back to the login page.

Figure 3.26: Mobile drawer functionality interface

Figure 3.26 illustrates the mobile drawer functionality interface for the users who would access the system through mobile device browsers. It is used to easily navigate through the different pages of the application to compensate for the small screen displays of mobile devices.

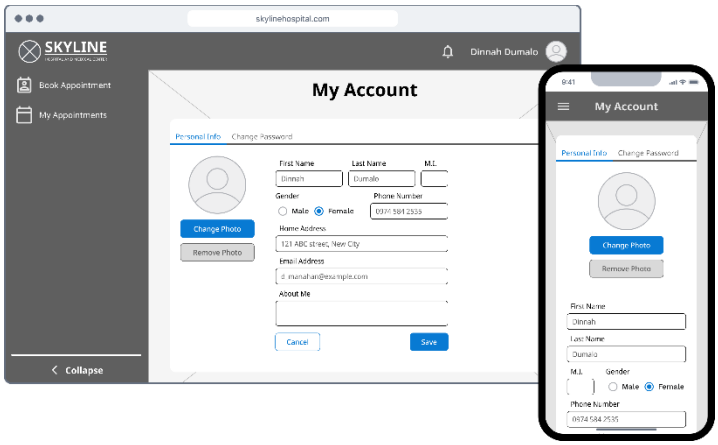


Figure 3.27: Patient profile interface

Figure 3.27 represents a patient user's profile page interface where the patient's personal information is displayed. Patients may also edit or update the displayed information anytime.

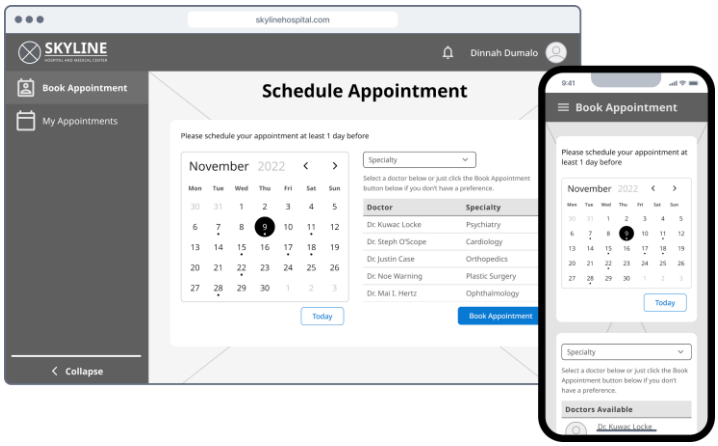


Figure 3.28: Patient book appointment interface

Figure 3.28 portrays the interface when a patient books for an appointment. The patient can select a date in the calendar and a list of doctors/specialists available for the selected date will appear on the table beside the calendar. The specialty dropdown will allow for sorting of the list of doctors based on the specialty selected.

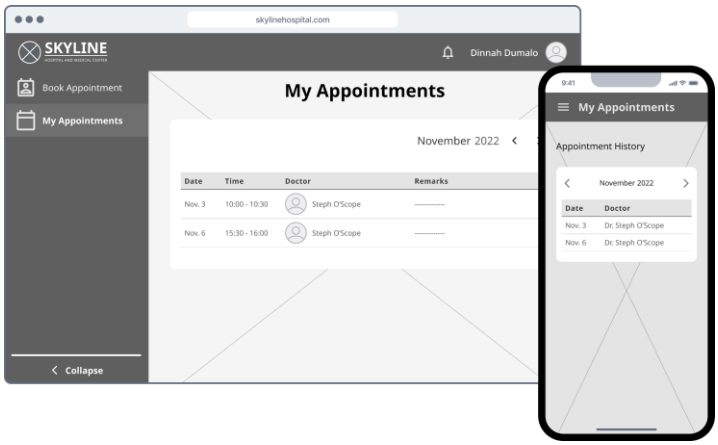


Figure 3.29: Patient appointments history interface

Figure 3.29 illustrates the patient user's appointments page. It displays their booked appointments, the date, the time, and the doctor assigned to the scheduled appointment. The user can also sort out the appointments per month by selecting a specific month using the arrow keys in the page.

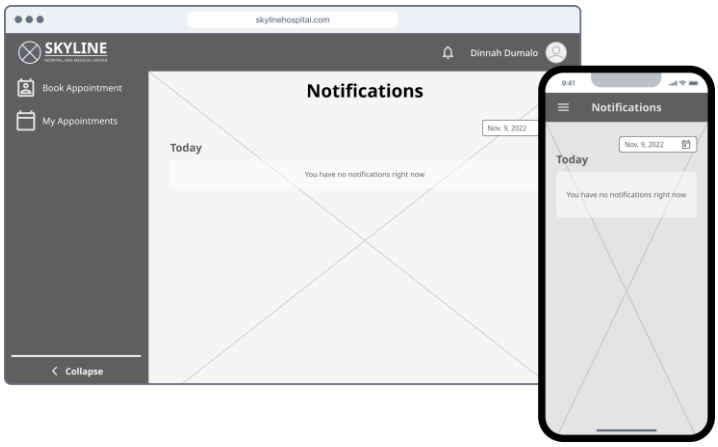


Figure 3.30: Patient notifications interface

Figure 3.30 shows a patient user's notifications page. It allows the patient user to be notified and reminded about scheduled appointments, system announcements, and system messages.

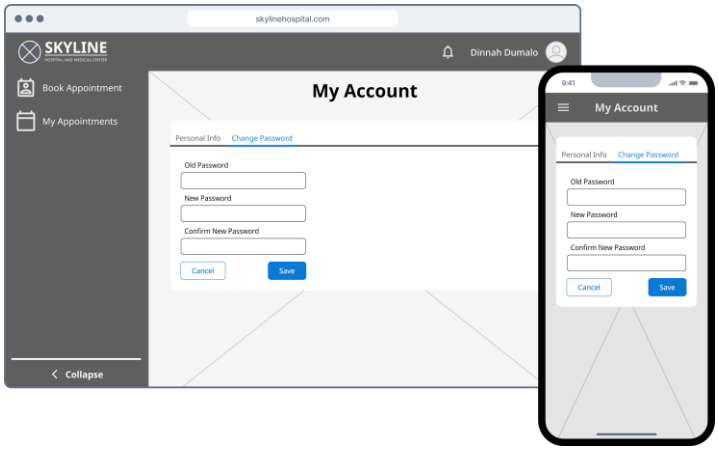


Figure 3.31: Patient change password interface

Figure 3.31 depicts a patient account for changing passwords. It allows the patient user to update the old password for security purposes.

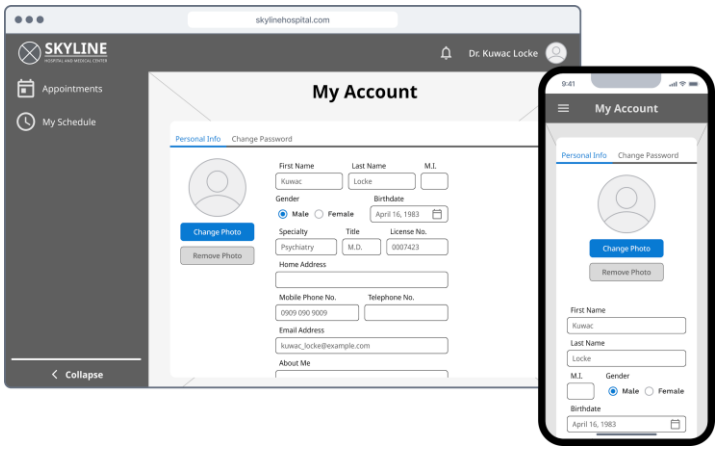


Figure 3.32: Doctor profile interface

Figure 3.32 illustrates a doctor account's profile page. Aside from personal information, a doctor user may be able to provide credentials such as his/her specialty, title, and license number.

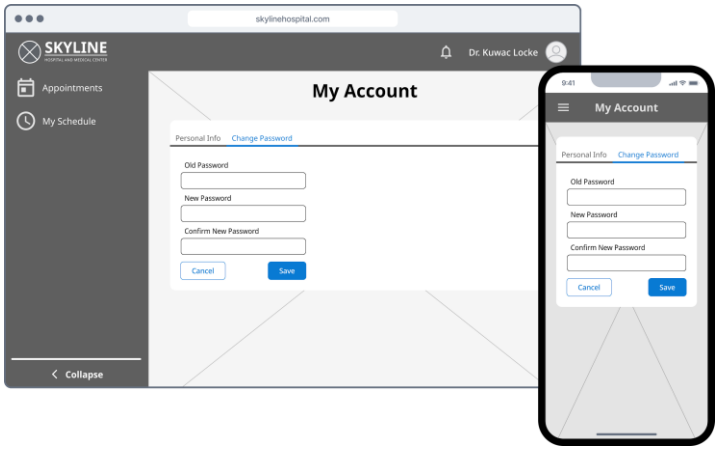


Figure 3.33: Password change interface

Figure 3.33 displays the change password page interface for the doctor account. The doctor user may be able to update his/her password for security purposes.

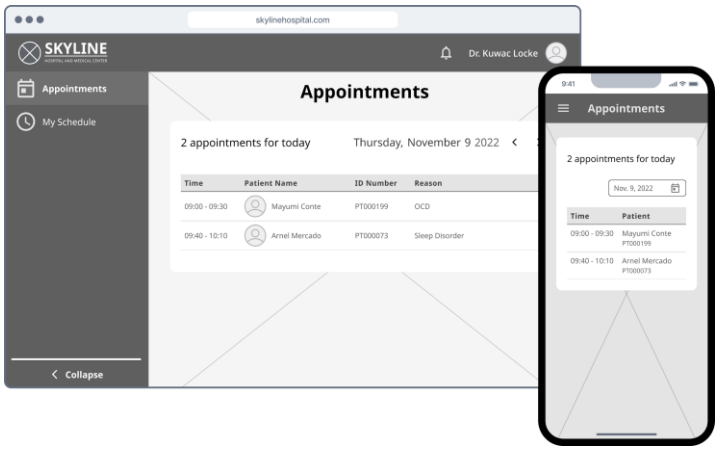


Figure 3.34: Doctor appointments interface

Figure 3.34 illustrates the doctor appointments page interface. It allows the doctor to view the booked appointments for him/her which displays the time of appointment, the patient's name and ID, as well as the patients' reason for booking the appointment.

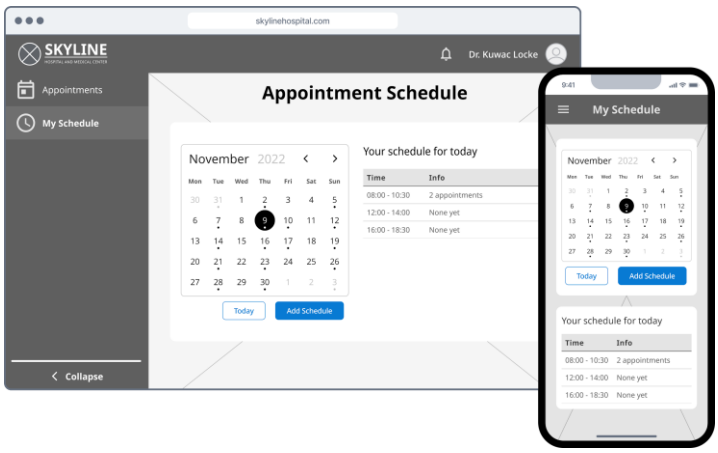


Figure 3.35: Doctor schedule interface

Figure 3.35 represents a doctor user's schedule page. It allows for viewing the appointments booked for the doctor on a specified time and date.

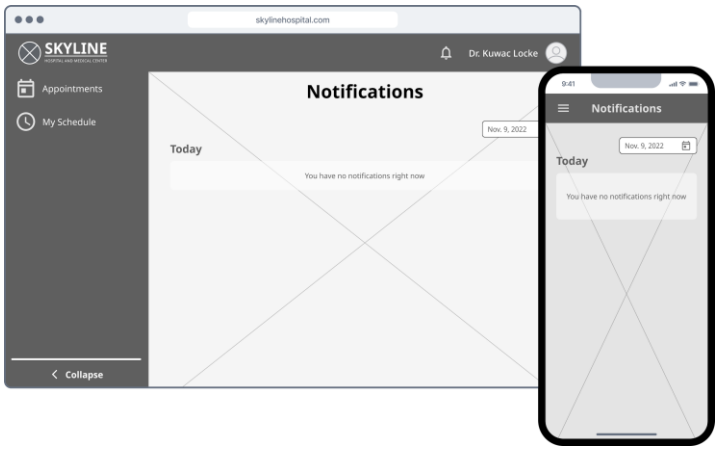


Figure 3.36: Doctor notifications interface

Figure 3.36 illustrates a doctor user's notifications page. It allows the doctor user to be notified and reminded about scheduled appointments, system announcements, and system messages.

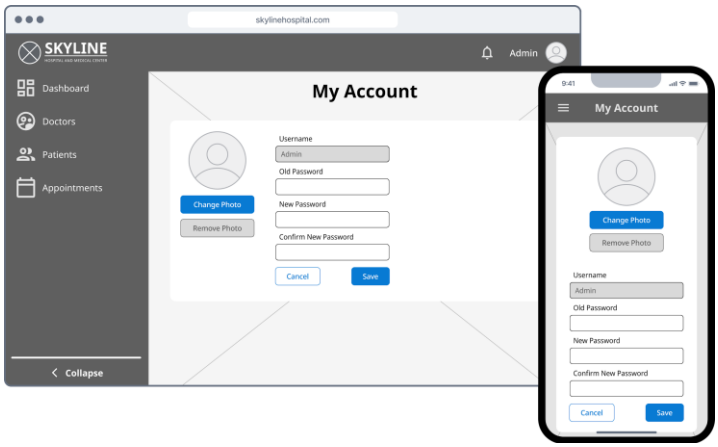


Figure 3.37: Admin change password interface

Figure 3.37 represents an admin user changing the password for the account. The password may be changed or updated by the admin user at any time.

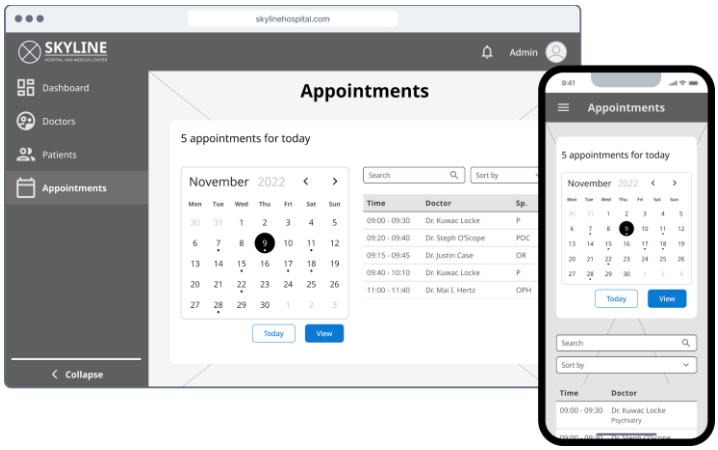


Figure 3.38: Admin appointments interface

Figure 3.38 shows the page for admin for viewing the appointments page. The admin may check all booked appointments and manage an appointment by clicking an appointment displayed in the list.

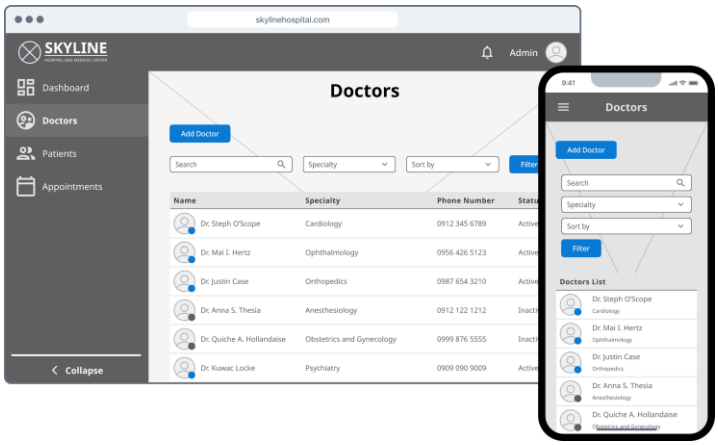


Figure 3.39: Admin doctor list page interface

Figure 3.39 illustrates an admin user's view of the doctor list page. Admins can manage the list by adding doctors, checking their personal information, and checking their statuses.

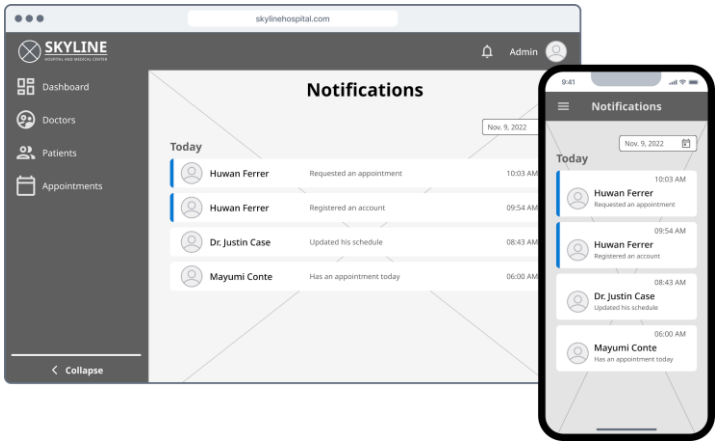


Figure 3.40: Admin notifications interface

Figure 3.40 shows an admin user's view of the notifications page. The admin will be notified about doctor schedule changes and appointments.

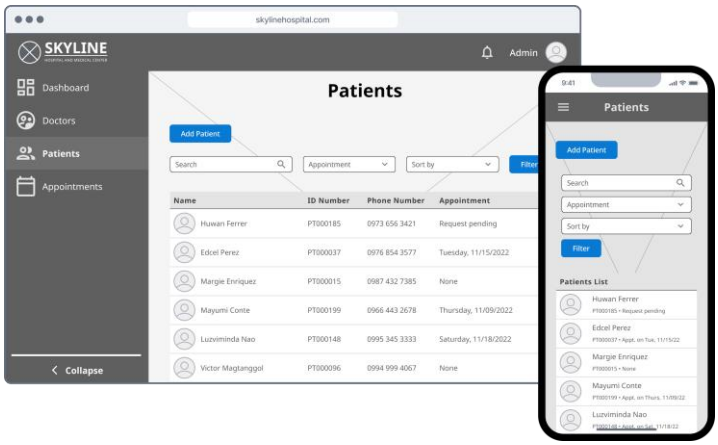


Figure 3.41: Admin patient appointments interface

Figure 3.41 illustrates an admin's view of the patient appointments interface. The admin is able to view and manage patient and appointment information.

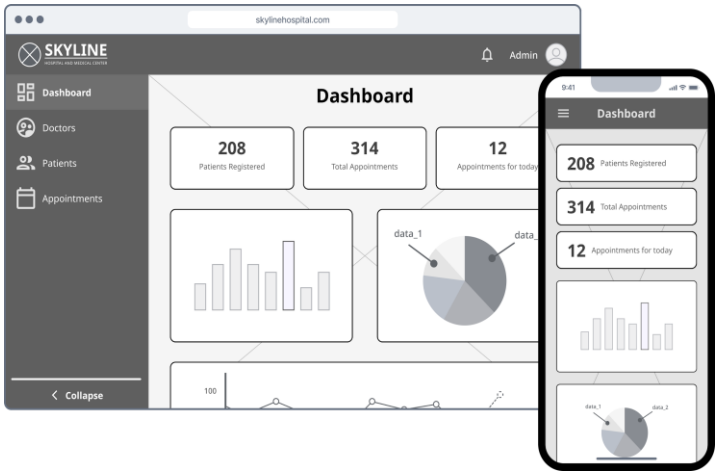


Figure 3.42: Admin data analytics dashboard

Figure 3.42 presents the data analytics dashboard page in the admin's view. This is where forecasting analytics reports are found that will offer information and insights derived from data in Skyline Hospital and Medical Center.

3.7 Data Gathering

Conversational Interview

In this study, the researchers conducted interviews with the available personnel at Skyline Hospital and Medical Center (SHMC) to gather insights and recommendations to improve specific aspects of the appointment management system.

Data Collection

To conduct effective data analytics research, data collection is a crucial step, and it is important to determine where to collect the data from.. In addition, the researchers requested actual raw data from SHMC's historical records of patient arrivals in the outpatient unit. This is then transformed to time series data for analysis and forecasting of future patient arrivals.

Statistical Tool

To measure the agreement level of the responses, a Likert scale ranging from 1 to 5 will be utilized.

$$W = \frac{\sum_{i=1}^{n} w_1 x_i}{\sum_{i=1}^{n} w_1} \qquad (3.1)$$

Equation 3.1 illustrates the formula to use. The weighted mean is computed by multiplying the summation of weight by the summation of x and then dividing it by the summation of weight. The weight variable refers to the respondents while the x variable is their responses. The equation is utilized for the ISO 25010 evaluation that consists of functional suitability, performance efficiency, compatibility, usability, reliability, security,

maintainability, portability, and forecasting ability. The rating scale and its corresponding values are shown below:

Table 3.1: Rating Scale Values

Scale	Range	Descriptive Equivalent
1	1.00 - 1.80	Strongly Disagree
2	1.81 - 2.60	Disagree
3	2.61 - 3.40	Neutral
4	3.41 - 4.20	Agree
5	4.21 - 5.00	Strongly Agree

Table 3.1 shows the rating scale values with columns named Scale, Range, and Descriptive Equivalent. If the average response score falls within 1.00 - 1.80, it would mean that the respondents strongly disagreed with the provided question based on the Likert Scale value of 1. If it is in the 1.81 - 2.60 range, the respondents intended to disagree with the question based on the Likert Scale value of 2. When it falls within the 2.61 - 3.40 range, the respondents neither agree nor disagree with the question based on the Likert Scale value of 3. An average response score that falls in the range of 3.41 - 4.20 means that the respondents agreed with the question based on the Likert Scale of 4. Lastly, if the average response score falls in the range of 4.21 - 5.00, the respondents intended to strongly agree with the question based on the Likert Scale of 5.

3.8 Testing and Evaluation Procedures

Table 3.2: Login and registration test cases

Case	Test Case Scenario	Expected Result
1	Account login (unverified account)	- User should not be able to login if the account is not yet verified
2	Account login (verified account)	- User should be able to login if correct username and password is entered - App should display the correct user interface
3	Keep account logged in	- The user should be able to login automatically if they did not log out their account on normal browsers (not in private/incognito mode)
4	Patient account registration	- Patient should be able to register after filling up valid information and clicking the register button

Table 3.2 demonstrates the scenarios the users can do in the login and registration page. This test is conducted to ensure that the user validation is working properly.

Table 3.3: Admin, Doctor, and Patient account test cases

Case	Test Case Scenario	Expected Result
1	Admin views the dashboard	- The app should display the up to date analytics and data visualizations
2	Admin manages the doctor accounts	- Admin should be able to add, edit, and delete an account
3	Admin manages the patient accounts	- Expected result similar to Case 2
4	Admin manages the appointments	- Admin should be able to view all appointments for the selected date
5	Doctor views his/her appointments	- Doctor should be able to view all his/her appointments for the selected date
6	Doctor sets his/her appointment schedules	- Doctor should be able to view, set, and edit his/her appointment schedule
7	Patient books an appointment	- The app will send the appointment request to admin for processing
8	Patient views his/her appointment history	- The patient should be able to view all of his/her appointments
9	The admin, patient, or doctor checks the notifications	- The app should display the account’s notification/s

Table 3.3 demonstrates the main scenarios the admin, doctor, and patient can do in their specific interface. This test is conducted to ensure that the functionalities in the admin, doctor, and patient account are correct and working properly.

Table 3.4: Web application test cases

Case	Test Case Scenario	Expected Result
1	App performance testing	- The app runs smoothly on both web and mobile devices
2	Cross browser compatibility testing	- The app should be able to run on different browsers
3	Display responsiveness testing	- The app layout should be responsive when viewing on different devices like desktop, tablet, and mobile
4	Page navigation testing	- The app navigates to the correct page and displays the contents correctly - The app prevents navigation to unauthorized pages

Table 3.4 demonstrates the scenarios of the web application what should be the expected result in testing the application. This test is conducted to ensure that the application is performing well and compatible with most of the browsers and mobile devices.

Table 3.5: Time series forecasting test cases

Case	Test Case Scenario	Expected Result
1	The administrator should be able to view future patient arrival values in the dashboard	- The administrator should be able to view future patient arrival values based on the set forecast horizon.
2	Test the model's forecasting ability in the test set	- The forecasted patient arrival values must closely align with the actual values in the test set within a reasonable error margin.
3	The model should be able to forecast on future dates that is outside the dataset	- The model should be able to successfully generate forecasts for the future

Table 3.5 shows the test cases for the time series forecasting feature that complements the appointment management system. The test is done to know if the forecasting model accurately gives out results.

Chapter IV

RESULTS, ANALYSIS AND DISCUSSION

The fourth chapter presents the answers to the problem statements made in the first chapter, discusses the features and components of the system, and the procedures conducted in the study.

The objective of the research is to create an appointment management web application system with a time series forecasting feature for future patient arrivals. The system consists of three user types: patients, doctors, and administrators. The admin has full control of the system and is the only one who has access to viewing the time series forecasts of future patient arrivals in the dashboard. Patients can book appointments through their accounts, while the doctors and specialists can control the schedules of their given time in their accounts, and view the patients' appointments assigned to them. A time series forecasting model is built and implemented to forecast future outpatient arrivals which would offer insights, aid in decision-making and proper resource allocation.

Research Question 1: What are the features of the system that will organize the following:

The main features of the system are: account authentication feature, where every user needs to register and login in the app to be able to use the web application; authorization feature, which allows or prevents users from accessing the different pages in the application; data analytics feature; appointment booking feature, which is used to schedule patient appointments for available services offered by the hospital; and notification feature, which displays the notification for the specific user. The system also

includes features for managing user accounts and services in the admin page and for managing doctor schedules in the doctor's account interface.

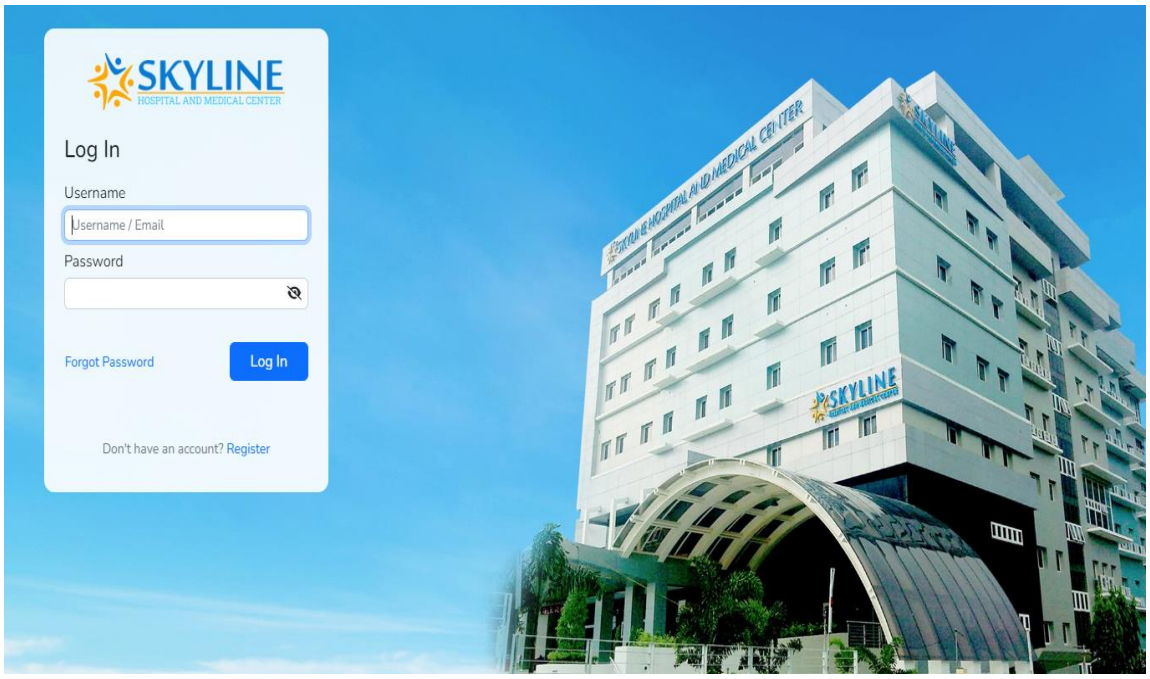


Figure 4.1: Login page

Figure 4.1 shows the login page of the application, it is used as an account registration and authentication for the users. Thus, it gives them the privilege to access their own account and designated interfaces. Users are also given the chance to recover their accounts through changing their password, especially when they forget their own password. Hence, it can be done when clicking the "Forgot Password" button. Furthermore, the interface provides a balance and appealing colors for the designs in order to give users a better experience in using the application.

Figure 4.2: Registration page

Figure 4.2 displays the interface for registering new accounts. This registration page is only for patient users. For the doctor account registration, the admin will be the one who registers the doctor's account in the doctor management page in the admin interface. Admin accounts are initially created upon the initialization of the backend server. Upon clicking the register button, the system will check first if all the inputs are valid before sending it to the backend server for account registration and will provide the appropriate validation message if the inputs are invalid. After the inputs are successfully validated, the system will now send the registration information to the backend server and will be displayed a successful registration popup message with instructions on how to verify the registered account.

Table 4.1: Login & registration test cases

Test Case Scenario	Expected Result/s	Actual Result/s	Pass/Fail
Account login (unverified account)	- User should not be able to login if the account is not yet verified	- The user cannot log in if the account is not yet verified	Passed
Account login (verified account)	- User should be able to login if correct username and password is entered - App should display the correct user interface	- The user can log in if the login credentials are correct. - The correct user interface is presented to the user upon login	Passed
Keep account logged in	- The user should be able to login automatically if they did not log out their account on normal browsers (not in private/incognito mode)	- The account is still logged in even if the user exits the browser	Passed
Patient account registration	- Patient should be able to register after filling up valid information and clicking the register button. -	- The patient can register if the information provided is valid. Also, the patient cannot register for more than one account with the same email address or mobile number	Passed

Table 4.1 shows the test cases and actual results for account login and registration. When logging in an unverified account, the app will display a message telling that the account should be verified first. The verification instruction is provided after registering for an account. After successful registration, the system will send a verification email to the email provided in registration form.

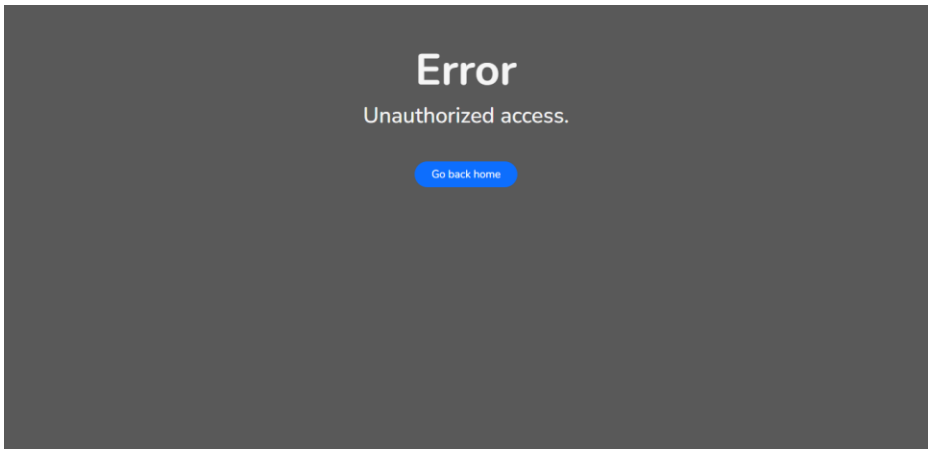


Figure 4.3: Unauthorized access page

Figure 4.3 shows the results of accessing interfaces directly without logging in users' credentials. Thus, for the authorization, the user is presented with the error page containing a message of "Unauthorized access." if they visit other pages they are not authorized to access. For instance, a patient user cannot visit the admin dashboard page. Nevertheless, if the patient user still tries to access the admin interface by entering the page's link, it will result in the error page.

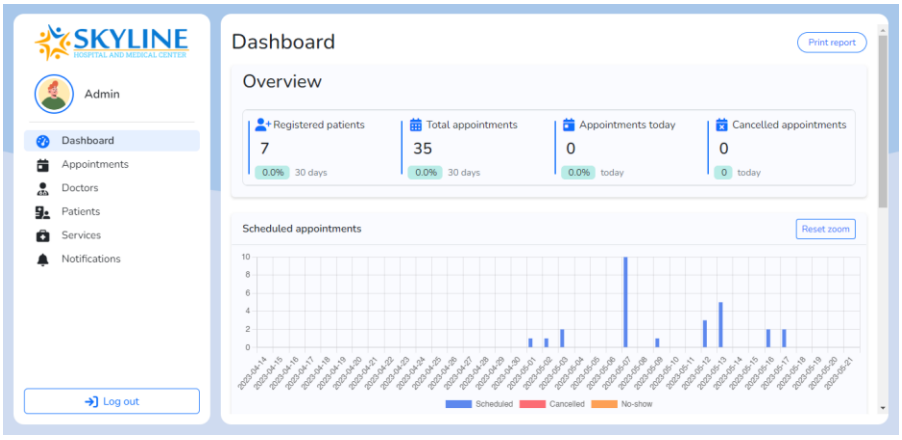


Figure 4.4: Admin dashboard overview

Figure 4.4 illustrates the admin dashboard page, where the analytics for the application and forecasting are displayed. The dashboard includes an overview section for displaying numbers like the registered patient count, the total appointments booked using

the app, the number of appointments for the current date, and the number of canceled appointments. It also displays the percent change for each item. The bar graph below the overview section displays the plot for the number of scheduled, canceled, and no-show appointments.

Table 4.2: Admin account test cases

Test Case Scenario	**Expected Result/s**	**Actual Result/s**	**Pass/Fail**
Admin views the dashboard	- The app should display the up to date analytics and data visualizations	- The app displays the proper and up-to-date data in the admin dashboard	Passed
Admin manages the doctor accounts	- Admin should be able to add, edit, and delete an account	- The admin can add, edit, and delete doctor accounts	Passed
Admin manages the patient accounts	- Expected result similar to test case above	- The admin can add, edit, and delete patient accounts	Passed
Admin manages the appointments	- Admin should be able to view all appointments for the selected date	- The admin can view all the appointments for the selected date	Passed

Table 4.2 shows the test cases and actual results for the admin account. The main features for admin are the analytics visualization in the dashboard, view all appointments, and management of doctors and patients accounts.

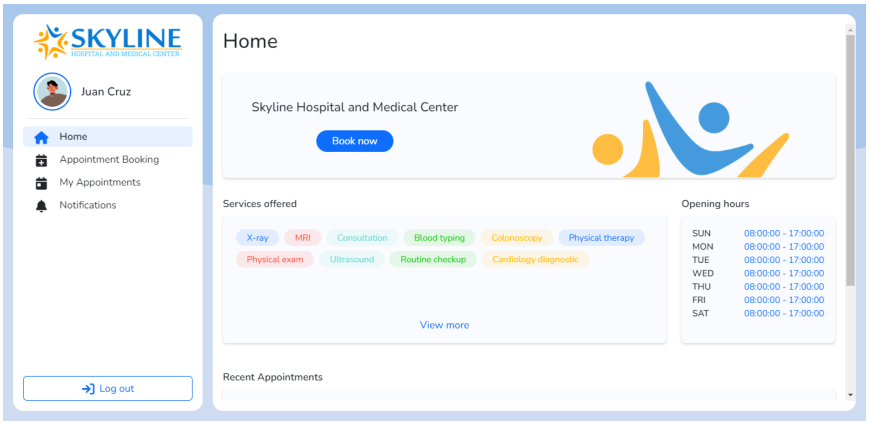


Figure 4.5: Patient home page

Figure 4.5 shows the home page for the patient users. In scheduling an appointment, the patient should be registered and verified first before they can schedule an appointment. Upon login, the patient will be redirected to the home page. In the home page, they will be able to see the available services offered by the hospital and the opening hours. if they previously scheduled an appointment before.

1.1 Appointment of patients;

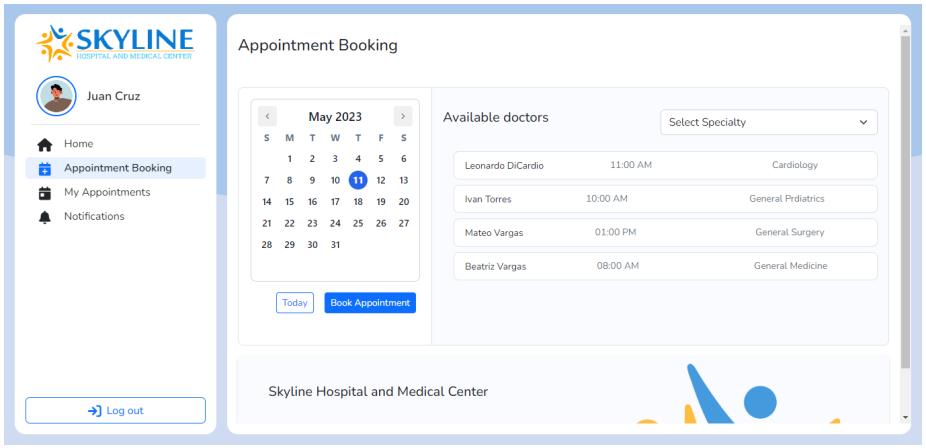


Figure 4.6: Patient appointment booking page

Figure 4.6 shows the appointment booking page for patient users. In this page, the patient can only schedule an appointment for tomorrow or up to one week in advance. They will also be able to see the available doctors for the selected date as well as the doctor's

available time for appointments. The registered patient can also act as a representative and schedule an appointment for another person. In this case, the user will be asked to fill out additional information like their name and contact number. Upon confirmation, the user will be automatically booked.

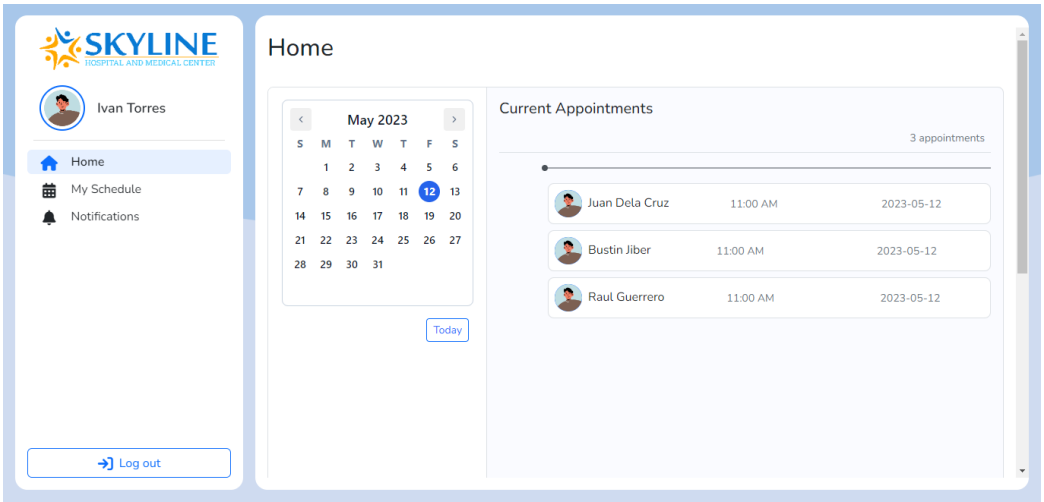


Figure 4.7: Doctor home page

Figure 4.7 shows the home page for doctors. Upon login using a doctor account, the user will be redirected to the doctor's home page. In the home page, they will be able to see a list of all the scheduled appointments for the selected date.

Table 4.3: Patient account test cases

Test Case Scenario	Expected Result/s	Actual Result/s	Pass/Fail
Patient books an appointment	- The app automatically books the appointment and sends a notification to the selected doctor	- Appointments are automatically booked. A notification is sent if the booking is successful	Passed
Patient views their appointment history	- The patient should be able to view all of their appointments	- The patient can view their appointment bookings.	Passed

Table 4.3 shows the test cases and actual results for the patient account. In scheduling for an appointment, the patient is required to fill up the information in the

appointment booking dialog. Upon submission, and if the inputs are valid, the app will display a final confirmation message. After confirming the appointment request, it will then display a success or fail popup dialog and will notify the selected doctor if the appointment request was successful.

1.2 Schedules of doctors and specialists

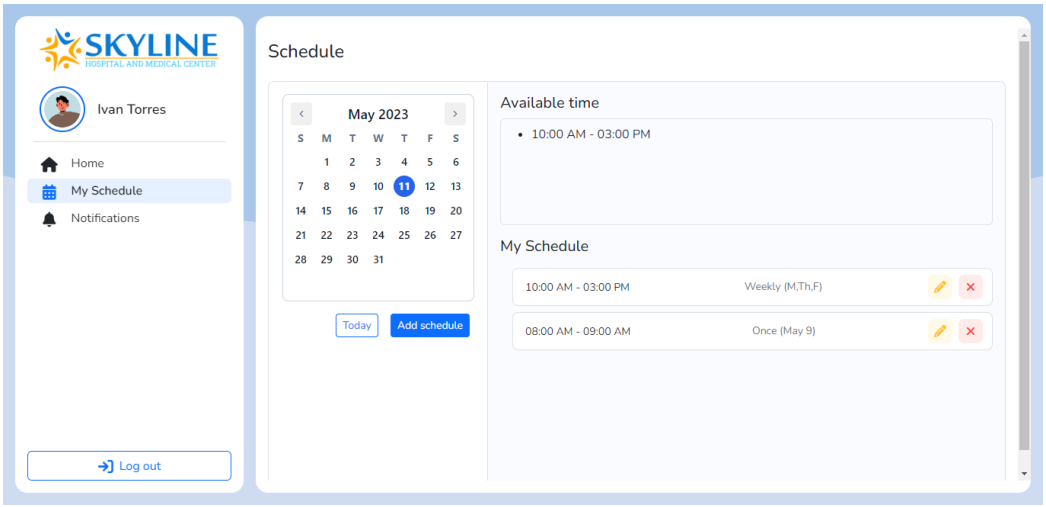


Figure 4.8: Doctor's schedule page

Figure 4.8 shows the doctor schedule page. It allows the doctors and specialists to set their available schedules for appointments. The doctor and specialist can select whether the schedule will repeat only once or weekly. They also need to provide the start and end dates and times for their schedule, and upon confirmation, the schedule will be automatically added.

Table 4.4: Doctor account test cases

Test Case Scenario	Expected Result/s	Actual Result/s	Pass/Fail
Doctor views his/her appointments	- Doctor should be able to view all his/her appointments for the selected date	- The appointments for the selected date are displayed	Passed
Doctor sets his/her appointment schedules	- Doctor should be able to view, set, and edit his/her appointment schedule	- The doctor can view, edit, and delete their appointment schedule	Passed
The doctor checks the notifications	- The app should display the account's notification/s	- The app displays the notifications for the current account	Passed

Table 4.4 test case scenario demonstrates that the Doctor account's test cases are working correctly and executing the test cases accurately. The application displays the appointment schedules, and the doctor can access, modify, or remove it. Additionally, the doctor account also has a notification feature. Ultimately, all of the criteria outlined for the doctor account test cases have been fulfilled.

Research Question 2: How will the system manage the patients' appointments and schedules of doctors and specialists?

In scheduling an appointment, the patient should be registered and verified first before they can schedule an appointment.

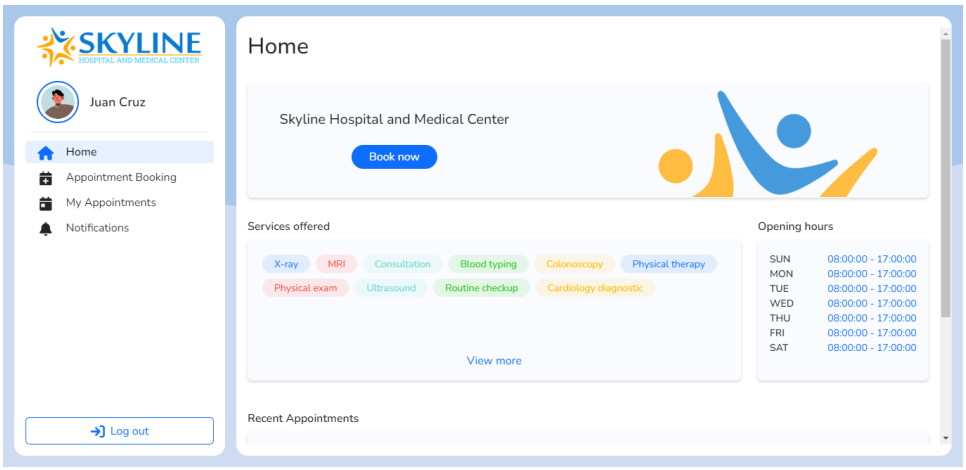


Figure 4.9: Patient's Home page

Figure 4.9 shows the home page for patient users, and upon login, the patient will be redirected to the home page. In this page, they will be able to see the available services offered by the hospital and the opening hours. They will also be able to view their most recent appointments if they had previously scheduled an appointment.

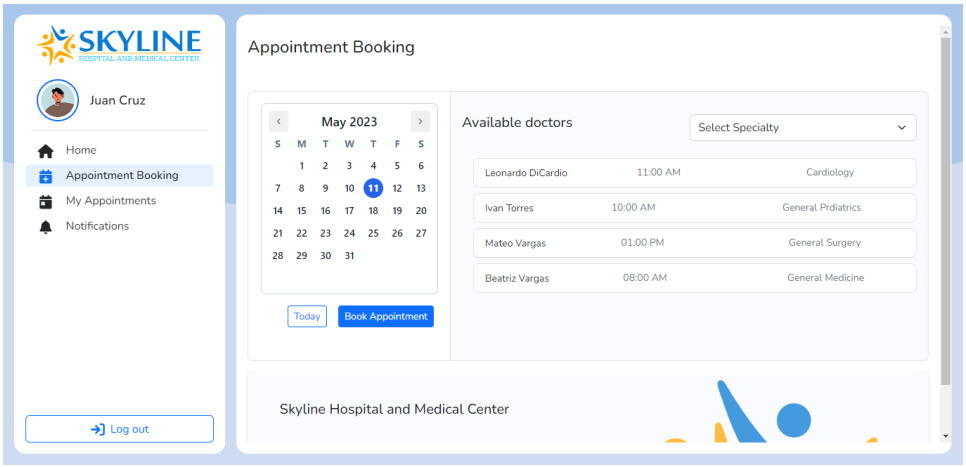


Figure 4.10: Patient's Appointment Booking page

Figure 4.10 shows the patient appointment booking page. The patient can only schedule an appointment for tomorrow or up to one week in advance. They will also be able to see the available doctors for the selected date as well as the doctor's available time for appointments. The registered patient can also act as a representative and schedule an

appointment for another person. In this case, the user will be asked to fill out additional information like their name and contact number.

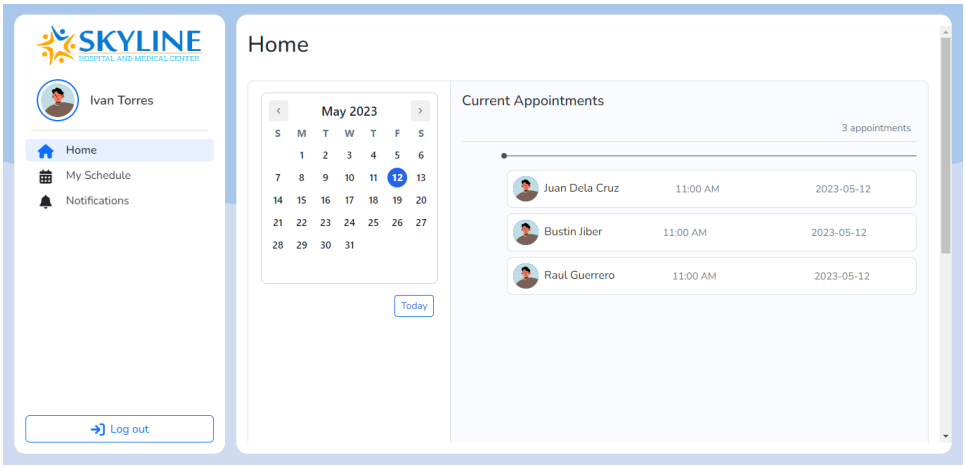


Figure 4.11: Doctor's Home page

Figure 4.11 shows the home page for doctor users. This is the page that will be presented to them upon login using a doctor account. In the home page, they will be able to see a list of all the scheduled appointments for the selected date.

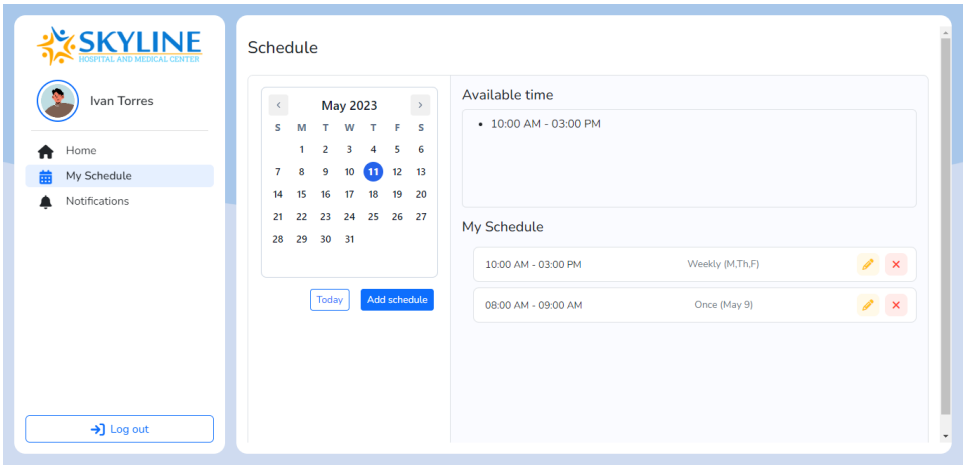


Figure 4.12: Doctor's My Schedule page

Figure 4.12 shows the page for doctor schedules. This page allows the doctors and specialists to set their available schedules for appointments. The doctor and specialist can select whether the schedule will repeat only once or weekly. They also need to provide the

start and end dates and times for their schedule, and upon confirmation, the schedule will be automatically added.

Research Question 3: How will the time series data be organized and create forecasts of future patient arrivals?

The time series forecasting feature of the system is discussed in this segment of the chapter. It goes through the process of loading the data, feature creation and engineering, model building and testing, model evaluation through performance metrics, and future forecast generation of patient arrivals.

Forecasting with the eXtreme Gradient Boosting Model

The eXtreme Gradient Boosting (XGBoost) model is a powerful and scalable machine learning algorithm for decision tree boosting. With the utilization of an internal algorithm that merges the outputs of several trees, accurate forecasting is possible. XGBoost is able to acquire a stronger classifier derived from other classifiers, and can also handle problems like overfitting and missing values in the data. Parallel and distributed processing allows for significant decrease in runtime. The model can be represented with Equation 3.1:

$$Y^k = \sum_{i=1}^{n} l((y_i, y_i^{k-1}) + f_k(x_i)) + \Omega(f_k) \quad (4.1)$$

n represents the training data, while x_i and y_i are the feature vectors at the i^{th} instance. y_i^{k-1} is the forecasted value of the i^{th} instance at the $t-1^{th}$ iteration. l means the loss function (error metric) that computes the measure of difference between the labels, together with the last prediction, added to the fresh tree output. f_k is a fresh tree which

determines the i^th instance of x_i, and represents the regularization term which castigates the intricacy of the fresh tree (Fang, 2022).

XGBoost is not specifically designed for time series forecasting unlike the traditional time series models such as ARMA, ARIMA, and SARIMA. But by adding lag features as input in the model building process, the model would be able to learn the temporal structure patterns related to time in the data.

The time series forecasting process started with importing the necessary libraries and modules and loading the patient arrivals data into the JupyterLab environment.

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
color_pal = sns.color_palette()
plt.style.use('ggplot')
plt.style.use('fivethirtyeight')

import warnings
warnings.filterwarnings('ignore')

import xgboost as xgb
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

Figure 4.13: Importing the necessary libraries

Figure 4.13 shows the libraries imported in the JupyterLab environment. `numpy` is used for numerical computing, while `pandas` is used for loading data, data frame creation, and data wrangling for creating time series features. `matplotlib` and `seaborn` go together to make data visualizations with plot styles of 'ggplot' and 'fivethirtyeight', while `seaborn` uses a color palette. The `warnings` to prevent from raising exceptions and halting the program. Then `warnings.filterwarnings('ignore')` makes all the warning messages ignored and will not be printed out in the console. Then the `xgboost` library is

imported, which is also the machine learning algorithm to be used for building the forecasting model. Lastly, the `mean_squared_error` and `mean_absolute_error` functions were imported from the `sklearn.metrics` (from the `scikit-learn` library) module, these are the performance metrics to be used for evaluating the XGBoost forecasting model.

```
df = pd.read_csv("skyline_hospital_time_series_patient_arrivals.csv",
                 index_col=['Date'],
                 parse_dates=True)
```

Figure 4.14: Loading the time series data

Figure 4.14 shows the Python code for loading the time series dataset stored in the `df` variable signifying a data frame. `index_col=[‘Date’]` means that as the data is loaded it is set as the index of the data frame. The `parse_dates=True` argument will convert the ‘Date’ column into a datetime64 format, allowing it to be truly recognized as time rather than a Python object.

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 958 entries, 2020-05-18 to 2022-12-31
Freq: D
Data columns (total 1 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Patients  856 non-null    float64
dtypes: float64(1)
```

Figure 4.15: Information overview of the data

Figure 4.15 shows overview information about the data. It is stored in a data frame with a Datetime index from 2020-05-18 to 2022-12-31, a frequency of D means that the time series data is daily with 958 observations of patient arrivals. There is one column named ‘Patients’ which contains patient arrival counts per day.

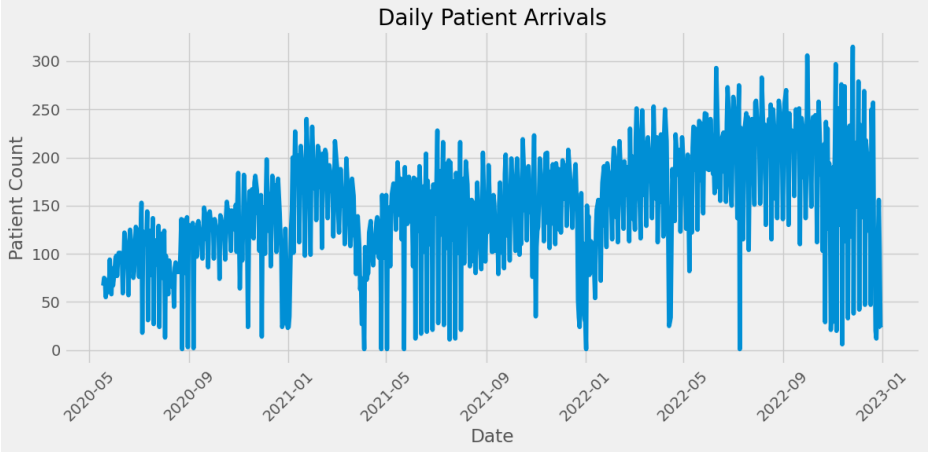


Figure 4.16: Visualizing the patient arrivals data

Figure 4.16 shows the line plot of the time series data. It can be seen that there are extremely low patient counts in some days. We can solve the outliers by using winsorization with the interquartile range (IQR) as the outlier detection method.

```
## Calculate the IQR and bounds
Q1 = patients_series.quantile(0.25)
Q3 = patients_series.quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 0.1 * IQR
upper_bound = Q3 + 1.5 * IQR

print(f'Lower Bound: {lower_bound}')
print(f'Upper Bound: {upper_bound}')
```

```
Lower Bound: 106.2
Upper Bound: 309.0
```

Figure 4.17: Calculating the IQR and setting lower and upper bounds

Figure 4.17 shows Python code for calculating the interquartile range (IQR) of the time series data and the setting of lower and upper bounds. `Q1` or the first quartile represents where 25% of the data are distributed, whereas `Q3` or the third quartile corresponds to where 75% of the data is distributed. The `IQR` is calculated by getting the difference of the third quartile and the first quartile. The `lower_bound` is calculated by

subtracting `Q1` from the threshold that is 0.1, times the `IQR`. Doing this makes the outlier detection on the lower bound more sensitive and better able to detect extremely low outliers that deviate significantly from the distribution of the data. While the `upper_bound` is calculated by adding the `Q3` with the threshold of 1.5, times the `IQR`. This makes it less sensitive to extremely high values as the researchers gave more importance in capturing the uptrends. Giving more importance to capturing the times where there will be a massive influx of patient arrivals would be crucial so that the hospital staff can be informed ahead of time in the days where outpatient demand will be intense. The value for the lower bound is set to be 106.2, or just 106 patient counts, and the upper bound is set to be 309 patient counts. Now that the upper and lower bounds have been set, all patient counts below 106.2 are adjusted and replaced with the lower bound value, while all patient counts above 309 are adjusted and replaced with the upper bound value. Handling outliers in time series data is a crucial step in order to be able to build models that can better fit the data.

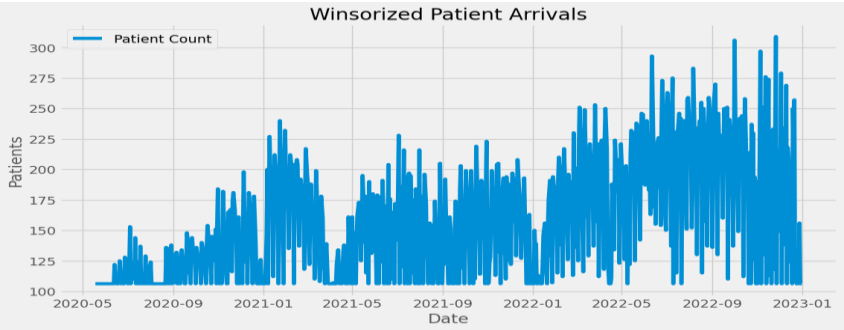


Figure 4.18: Winsorized patient arrivals data

Figure 4.18 illustrates the winsorized time series data. It can be seen that the extremely low outliers have been resolved and the data is now viable for building a forecasting model.

```python
def create_features(df, label=None):
    """
    Create time series features from the datetime index
    """
    df = df.copy()
    df['date'] = df.index
    df['day_of_week'] = df['date'].dt.dayofweek
    df['month'] = df['date'].dt.month
    df['day_of_year'] = df['date'].dt.dayofyear
```

Figure 4.19: Time series feature engineering

Figure 4.19 shows the time series feature engineering that will be used as the input features for the XGBoost model. It is done inside the function named `create_features()`. The 'date' column is extracted from the index since the time series data is daily patient arrivals. the 'day_of_week' column is engineered from the 'date' by using `pandas` built-in function for dates, `df['date'].dt.dayofweek`, the succeeding features also uses the `.dt` built-in function from `pandas`. The 'month' column is created by using `.dt.month`, and the 'day_of_year' column by using `dt.dayofyear`.

```python
window_size = 7  # 7-day rolling window

# Rolling minimum
df['rolling_min'] = df['Patients'].rolling(window=window_size).min()

# Rolling maximum
df['rolling_max'] = df['Patients'].rolling(window=window_size).max()

# Rolling sum
df['rolling_sum'] = df['Patients'].rolling(window=window_size).sum()

# Rolling mean
df['rolling_mean'] = df['Patients'].rolling(window=window_size).mean()

# Rolling median
df['rolling_median'] = df['Patients'].rolling(window=window_size).median()

# Rolling standard deviation
df['rolling_std'] = df['Patients'].rolling(window=window_size).std()
```

Figure 4.20: 7-day rolling statistics

Figure 4.20 shows the creation of rolling statistics as additional input features for the XGBoost model. The window size is set to 7, indicating a 7-day (1 week) rolling window. The `rolling_min` extracts the lowest patient counts for every week, while

`rolling_max` gets the highest patient counts for every week. The `rolling_sum` gets the total patient counts for every week, `rolling_mean` gets the average patient counts per week, `rolling_median` for the middle value for every week, and `rolling_std` gets the dispersion of the values for every week.

```
    # Rolling quantile (25th percentile)
    quantile_25 = 0.25
    df['rolling_quantile_25'] = df['Patients'].rolling(window=window_size).quantile(quantile_25)

    # Rolling quantile(50th percentile)
    quantile_50 = 0.50
    df['rolling_quantile_50'] = df['Patients'].rolling(window=window_size).quantile(quantile_50)

    # Rolling quantile (75th percentile)
    quantile_75 = 0.75
    df['rolling_quantile_75'] = df['Patients'].rolling(window=window_size).quantile(quantile_75)

    return df

df = create_features(df)
```

Figure 4.21: Rolling quantiles and feature creation

Figure 4.21 shows the last input features to be fed into the XGBoost model which are the rolling quantiles still having a 7-day window. ‘rolling_quantile_25’ gets the 7-day rolling 25th percentile of patient counts, ‘rolling_quantile_50’ extracts the 7-day rolling 50th percentile of patient counts, and ‘rolling_quantile_75’ gets the 7-day rolling 75th percentile of patient counts. This concludes the feature creation part and these features are created by running `df = create_features(df)`.

```
def add_lags(df):
    target_map = df['Patients'].to_dict()
    df['lag1'] = (df.index - pd.Timedelta('30 days')).map(target_map)
    df['lag2'] = (df.index - pd.Timedelta('90 days')).map(target_map)
    df['lag3'] = (df.index - pd.Timedelta('180 days')).map(target_map)
    return df

df = add_lags(df)
```

Figure 4.22: Adding lag features

Figure 4.22 shows the creation and addition of lag features as input features for the XGBoost model. Lag features in time series represent past values. As current values in

time series depend on previous values, the main objective of adding lag features is to capture time-related dependencies between the current and past values. It allows the XGBoost model to learn historical values and aid in capturing the trends and patterns in the data. The lag features also determine the forecast horizon, where in this case, the forecast horizon would be up to 180 days. Now that all of the input features have been set, it is time to set up the time series cross-validation environment where the XGBoost forecasting model will be built.

```
from sklearn.model_selection import TimeSeriesSplit
tss = TimeSeriesSplit(n_splits=8, test_size=119, gap=1)
df = df.sort_index()

fold = 0
preds = []

rmse_scores = []
mape_scores = []
smape_scores = []

# Initialize an empty DataFrame to store actual and predicted values
actual_vs_pred_df = pd.DataFrame()

for train_index, val_index in tss.split(df):
    train = df.iloc[train_index]
    test = df.iloc[val_index]

    train = create_features(train)
    test = create_features(test)
```

Figure 4.23: Setting up the time series cross-validation environment

Figure 4.23 shows the Python code for setting up the time series cross-validation environment. The `TimeSeriesSplit` class was used from the imported `sklearn.model_selection`. It implements 8-fold expanding window cross-validation wherein the model is trained on a specified amount of data, and does forecasting for the next period, then that period is added to the training data for the next fold iteration. In this

case, every fold has 119 test data points and a gap of 1 data point (day) to prevent leakage. The data frame `df`` is first sorted to ensure that the dates are properly arranged. The fold is first initialized as 0,and empty lists are initialized for the predictions (`preds = []`), root mean squared error scores (`rmse_scores = []`), mean absolute percentage error scores (`mape_scores = []`), and symmetric mean absolute percentage error scores (`smape_scores = []`), and a data frame for the actual vs predicted values (`actual_vs_pred_df``). The loop starts where the indices for the training set and testing set are generated for each fold. Then, the forecasting model is trained on the training set, evaluated on the testing set, and is repeated for each fold. The performance metrics are computed and stored for each fold in the loop. The `train` and the `test` variable calls the `create_features()` function that was used in doing feature engineering of the original data frame.

```
FEATURES = ['day_of_year', 'day_of_week', 'month', 'rolling_min', 'rolling_max',
            'rolling_sum', 'rolling_quantile_25', 'rolling_quantile_50',
            'rolling_quantile_75', 'rolling_mean', 'rolling_median',
            'rolling_std', 'lag1', 'lag2', 'lag3']

TARGET = 'Patients'

X_train = train[FEATURES]
y_train = train[TARGET]

X_test = test[FEATURES]
y_test = test[TARGET]
```

Figure 4.24: Defining the feature and target variable

Figure 4.24 illustrates the process of defining the feature and target variable. The `FEATURES` list contains the various time series features, together with the lag features as the input for the XGBoost forecasting model. The target is the only dependent variable, that is the ‘Patients’ variable. `X_train` and `X_test` make feature matrices for the training

and testing data. `y_train` and `y_test` produce the target vectors for the training and testing data.

```
reg = xgb.XGBRegressor(base_score=0.5, booster='gbtree',
                       n_estimators=900,
                       early_stopping_rounds=50,
                       tree_method='hist',
                       objective='reg:squarederror',
                       max_depth=3,
                       min_child_weight=3,
                       gamma=0,
                       learning_rate=0.01,
                       colsample_bytree=0.9,
                       subsample=0.7,
                       reg_lambda=0)

reg.fit(X_train, y_train, eval_set=[(X_train, y_train), (X_test, y_test)],
        verbose=100)

y_pred = reg.predict(X_test)
preds.append(y_pred)
```

Figure 4.25: XGBoost model building

Figure 4.25 demonstrates the process of initializing and building the XGBoost model using the XGBRegressor with specified hyperparameters. The model is then fitted into the training set and the testing set is specified as the evaluation set. `y_pred` stores the model's predictions on the test set.

```
rmse = root_mean_squared_error(y_test, y_pred)
mape = mean_absolute_percentage_error(y_test, y_pred)
smape = symmetric_mean_absolute_percentage_error(y_test, y_pred)

rmse_scores.append(rmse)
mape_scores.append(mape)
smape_scores.append(smape)

# Store actual and predicted values along with their datetime index in the DataFrame
temp_df = pd.DataFrame({"Actual": y_test, "Predicted": y_pred}, index=df.iloc[val_index].index)
actual_vs_pred_df = pd.concat([actual_vs_pred_df, temp_df])
```

Figure 4.26: Storing performance scores

Figure 4.26 illustrates the storing of performance metrics across all folds and how it was appended to the initialized lists. The actual and forecasted values are then stored and made into a data frame

```
RMSE across folds: 27.473
MAPE across folds: 12.811%
sMAPE across folds: 13.070%
```

Figure 4.27: Performance metric scores

Figure 4.27 shows the performance metric scores. The root mean squared error (RMSE) represents the magnitude error on average or the differences between the predicted and actual values. An RMSE of 27.473 means that the forecasts of patient arrival counts are off by 27.473. 12.811% is the mean absolute percentage error (MAPE), and it means that the patient arrival forecasts are off by 12.811% on average. Finally, the symmetric mean absolute percentage error (sMAPE) with a value of 13.070% means that the patient arrival forecasts are off by the given percentage.

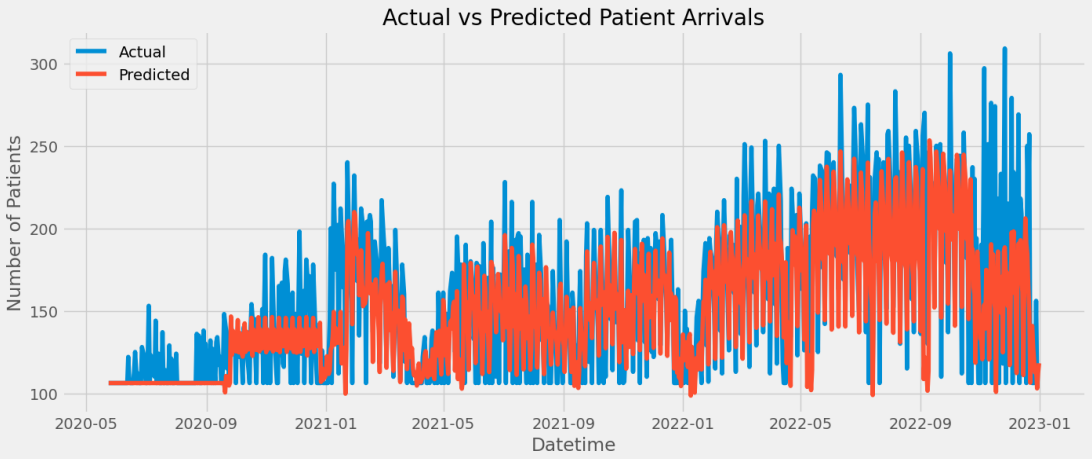


Figure 4.28: Plotting the XGBoost model's forecasts

Figure 4.28 shows the model's performance as a visualization. As the XGBoost model's forecasts are plotted, it can be seen that the forecasts do not seem to be able to capture the data's time patterns. It only starts to visibly capture them around the year 2021, and improves even further as time goes by.

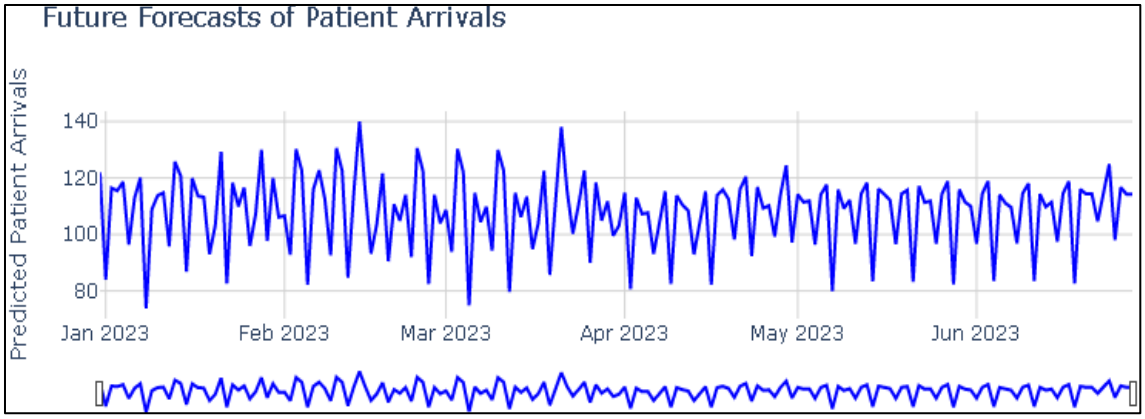


Figure 4.29: Forecasting future patient arrivals

Figure 4.29 shows the XGBoost forecasting model's future patient arrivals forecasts. Now that the XGBoost model has been built with considerable performance, it is now time for forecasting future patient arrivals. A new data frame, `future_df``, with the same frequency that is daily added a new column named 'isFuture', with boolean values that correspond to whether or not a day is from the future or not. The historical data frame and the `future_df` are then merged into a single data frame, and was also fed with the time series features (day of the week, day of the year, and month), rolling statistics (rolling mean, rolling minimum, rolling median, rolling standard deviation, rolling 25th quantile, rolling 50th quantile, rolling 75th quantile, all with a 7-day rolling window), and the lag features (7-day lag feature, 90-day lag feature, and 180-day lag feature). The built XGBoost model is utilized to forecast future patient arrivals, in which the forecasts are saved in a new column named `pred` and the `future_with_features` data frame. The 180-day forecast is done and plotted, finally being able to reach the main goal of forecasting future patient arrivals.

Table 4.5: Tabulated Representation of the testing set's actual vs forecasted values of future patient arrivals

Test Case Scenario	Actual Values	Forecasted Values	Remarks
1	114	106.185	Passed
2	134	123.164	Passed
3	117	114.445	Passed
4	167	156.613	Passed
5	182	193.579	Passed
6	197	199.751	Passed
7	227	224.359	Passed
8	174	165.588	Passed
9	128	134.496	Passed
10	127	133.514	Passed

Table 4.5 shows the test case scenarios for the forecasting model. It has 4 columns, namely, Test Case Scenario, Actual Values, Forecasted Values, and Remarks. The Test Case Scenario column lists down the number of test cases, it is numbered from 1 to 10. The Actual Values column represents patient counts in the actual patient arrivals time series data. The Forecasted Values column represents the predicted values made by the XGBoost forecasting model. The final column, named Remarks, tells whether or not the forecasted values have passed or failed. If the remark is passed, it means that the forecasted value is within the margin of error that is outputted from the performance metrics. Based on the forecasted values above, they are within the 12-13% margin of error that was outputted from the performance metrics.

Research Question 4: How will the system be evaluated using the ISO 25010 software standard criteria in terms of:

4.1 Functional Suitability;

Table 4.6: ISO 25010 Functional Suitability Evaluation

ISO 25010	STRONGLY AGREE		AGREE		NEUTRAL		DISAGREE		STRONGLY DISAGREE		MEAN	INTERPRETATION
FUNCTIONAL SUITABILITY		%		%		%		%		%		
1. The appointment management system provides all the necessary features I need for managing appointments.	3	13	17	73.9	2	8.7	1	4.3	0	0	3.96	AGREE
2. The appointment management system processes data accurately and produces correct results.	2	8.7	11	47.8	8	34.8	2	8.7	0	0	3.57	AGREE
3. The appointment management system's features are appropriate for its intended purpose.	7	30.4	9	39.1	4	17.4	2	8.7	1	4.3	3.83	AGREE
AVERAGE WEIGHTED MEAN											3.78	AGREE

Table 4.6 shows the ISO 25010 evaluation results for the criteria that is named Functional Suitability. The Functional Suitability evaluation results indicate that the majority of respondents agree that the appointment management system provides necessary features, processes data accurately, and has appropriate features for its intended purpose. Overall, the system's functional suitability is evaluated positively.

4.2 Performance Efficiency;

Table 4.7: ISO 25010 Performance Efficiency Evaluation

ISO 25010	STRONGLY AGREE		AGREE		NEUTRAL		DISAGREE		STRONGLY DISAGREE		MEAN	INTERPRETATION
PERFORMANCE EFFICIENCY		%		%		%		%		%		
1. The appointment management system responds quickly to user actions and requests.	4	17.4	7	30.4	8	34.8	3	13	1	4.3	3.43	AGREE
2. The appointment management system utilizes system resources efficiently.	2	8.7	9	39.1	10	43.5	1	4.3	1	4.3	3.43	AGREE
3. The appointment management system needs further optimization.	6	26.1	10	43.5	7	30.4	0	0	0	0	3.95	AGREE
AVERAGE WEIGHTED MEAN											3.60	AGREE

Table 4.7 shows that the evaluation suggests that the system generally responds quickly to user actions, utilizes system resources efficiently, and requires further optimization which indicates an overall agreement with the performance efficiency of the appointment management system.

4.3 Compatibility;

Table 4.8: ISO 25010 Compatibility Evaluation

ISO 25010	STRONGLY AGREE		AGREE		NEUTRAL		DISAGREE		STRONGLY DISAGREE		MEAN	INTERPRETATION
COMPATIBILITY		%		%		%		%		%		
1. The appointment management system is responsive in both mobile and desktop browsers.	4	17.4	9	39.1	7	30.4	1	4.3	2	8.7	3.52	AGREE
2. The appointment management system causes few conflicts or performance issues.	1	4.3	4	17.4	14	60.9	2	8.7	2	8.7	3.43	AGREE
AVERAGE WEIGHTED MEAN											3.48	AGREE

Table 4.8 shows that the respondents agree that the system is generally responsive in both mobile and desktop browsers, and it causes only a few conflicts or performance issues thus the compatibility of the system is evaluated positively.

4.4 Usability;

Table 4.9: ISO 25010 Usability Evaluation

ISO 25010	STRONGLY AGREE		AGREE		NEUTRAL		DISAGREE		STRONGLY DISAGREE		MEAN	INTERPRETATION
USABILITY		%		%		%		%		%		
1. The appointment management system's features and functions are easy to recognize and understand.	3	13	13	56.5	5	21.7	1	4.3	1	4.3	3.70	AGREE
2. I can learn how to use the appointment management system quickly and easily.	4	17.4	8	34.8	7	30.4	2	8.7	2	8.7	3.43	AGREE
3. The appointment management system is easy to operate and navigate.	3	13	12	52.2	6	26.1	1	4.3	1	4.3	3.65	AGREE
4. The appointment management system helps prevent and recover from user errors.	2	8.7	8	34.8	11	47.8	0	0	2	8.7	3.35	NEUTRAL
5. The user interface of the appointment management system is visually appealing and pleasant to use.	4	17.4	9	39.1	7	30.4	2	8.7	1	4.3	3.57	AGREE
AVERAGE WEIGHTED MEAN											3.54	AGREE

Table 4.9 shows that the system's features and functions are easy to recognize and understand. Users can learn how to use the system quickly, and it is easy to operate and navigate as the system has a visually appealing user interface. However, it is seen as needing improvement in helping prevent and recover from user errors. Overall, the usability of the system is evaluated positively.

4.5 Reliability;

Table 4.10: ISO 25010 Reliability Evaluation

ISO 25010	STRONGLY AGREE		AGREE		NEUTRAL		DISAGREE		STRONGLY DISAGREE		MEAN	INTERPRETATION
RELIABILITY		%		%		%		%		%		
1. The appointment management system is available when it's needed.	3	13	12	52.2	6	26.1	1	4.3	1	4.3	3.65	AGREE
2. The appointment management system consistently performs well without errors or crashes.	1	4.3	9	39.1	11	47.8	1	4.3	1	4.3	3.35	NEUTRAL
3. The appointment management system consistently performs well without errors or crashes.	1	4.3	12	52.2	7	30.4	3	13	0	0	3.48	AGREE
4. The appointment management system can recover quickly from crashes or errors.	2	8.7	10	43.5	8	34.8	2	8.7	1	4.3	3.43	NEUTRAL
									AVERAGE WEIGHTED MEAN		3.48	AGREE

Table 4.10 shows that the participants indicate that the system is generally accessible when required and operates smoothly without any errors or crashes. Nevertheless, there is room for improvement in terms of swiftly recovering from crashes or errors. Overall, the system's reliability is viewed positively, although there are some neutral aspects to consider.

4.6 Security;

Table 4.11: ISO 25010 Security Evaluation

ISO 25010	STRONGLY AGREE		AGREE		NEUTRAL		DISAGREE		STRONGLY DISAGREE		MEAN	INTERPRETATION
SECURITY		%		%		%		%		%		
1. The appointment management system protects my personal and patient information from unauthorized access.	8	34.8	10	43.5	3	13	0	0	2	8.7	3.96	AGREE
2. The appointment management system ensures the integrity of data by preventing unauthorized modification.	6	26.1	8	34.8	7	30.4	1	4.3	1	4.3	3.74	AGREE
3. The appointment management system verifies the identity of users before granting access to sensitive information.	7	30.4	8	34.8	5	21.7	0	0	3	13	3.70	AGREE
4. The appointment management system verifies the identity of users before granting access to sensitive information.	5	21.7	10	43.5	4	17.4	1	4.3	3	13	3.57	AGREE
AVERAGE WEIGHTED MEAN											3.74	AGREE

Table 4.11 shows that the evaluation indicates a generally positive response regarding its security aspects. The system effectively protects personal and patient information from unauthorized access, ensures data integrity, and verifies user identities before granting access to sensitive information, therefore, displaying an overall agreement with the system's security measures.

4.7 Maintainability;

Table 4.12: ISO 25010 Maintainability Evaluation

ISO 25010	STRONGLY AGREE		AGREE		NEUTRAL		DISAGREE		STRONGLY DISAGREE		MEAN	INTERPRETATION
MAINTAINABILITY		%		%		%		%		%		
1. The appointment management system is easy to modify and update.	2	8.7	14	60.9	4	17.4	2	8.7	1	4.3	3.61	AGREE
2. It is easy to diagnose and analyze issues with the appointment management system.	2	8.7	11	47.8	8	34.8	2	8.7	0	0	3.57	AGREE
3. The appointment management system can be modified without introducing new defects or degrading existing quality.	2	8.7	12	52.2	6	26.1	3	13	0	0	3.57	AGREE
AVERAGE WEIGHTED MEAN											3.58	AGREE

Table 4.12 shows the ISO 25010 evaluation results for the criteria that is maintainability. The respondents agree that the appointment management system's maintainability feature is generally easy to modify and update. It is also considered easy to diagnose and analyze issues, and modifications can be made without introducing new defects or degrading existing quality. Overall, the maintainability of the system is evaluated positively.

4.8 Portability;

Table 4.13: ISO 25010 Portability Evaluation

ISO 25010	STRONGLY AGREE		AGREE		NEUTRAL		DISAGREE		STRONGLY DISAGREE		MEAN	INTERPRETATION
PORTABILITY		%		%		%		%		%		
1. The appointment management system adapts well to different devices, browsers, and operating systems.	4	17.4	14	60.9	3	13	1	4.3	1	4.3	3.83	AGREE
2. The appointment management system is easy to access	4	17.4	13	56.5	5	21.7	0	0	1	4.3	3.83	AGREE
									AVERAGE WEIGHTED MEAN		3.83	AGREE

Table 4.13 shows the ISO 25010 evaluation results for the criteria that is named Portability. The results gained from the respondents of the study indicate that the system demonstrates good compatibility with various devices, browsers, and operating systems in both desktop and mobile device platforms. Additionally, it is perceived as user-friendly in terms of accessibility thus the system receives a positive evaluation in terms of its portability.

4.9 Forecasting Ability;

Table 4.14: ISO 25010 Forecasting Ability Evaluation

ISO 25010	STRONGLY AGREE		AGREE		NEUTRAL		DISAGREE		STRONGLY DISAGREE		MEAN	INTERPRETATION
FORECASTING ABILITY		%		%		%		%		%		
1. The time series forecasting feature provides accurate predictions of future patient arrivals.	2	8.7	14	60.9	5	21.7	0	0	2	8.7	3.61	AGREE
2. The time series forecasting feature helps to inform in better resource allocation and planning.	3	13	8	34.8	9	39.1	1	4.3	2	8.7	3.39	NEUTRAL
AVERAGE WEIGHTED MEAN											3.5	NEUTRAL

Table 4.14 shows the ISO 25010 evaluation results for the added criteria that is named Forecasting Ability. The Forecasting Ability criteria is added to evaluate the forecasting feature of the system, which is the time series forecasting of patient arrivals in the hospital's outpatient department. The evaluation results indicate that the time series forecasting feature of the system is capable of generating accurate predictions for future patient arrivals, however, its effectiveness in resource allocation and planning receives a neutral evaluation. The system's forecasting ability is assessed as neutral. This meant that the respondents of the study may want to improve the time series forecasting ability as it may not have been deemed as acceptable or even highly acceptable.

Chapter V

SUMMARY, CONCLUSION AND RECOMMENDATIONS

This chapter provides the overall summary of the paper and it includes the summary of findings, conclusions, and recommendations based on the evaluation results and the suggestions of the panelists. The purpose of the research was to improve outpatient management in the hospital by developing an appointment management system for them. Furthermore, a time series forecasting feature was added, to forecast future patient arrivals in the outpatient unit.

5.1 Summary of Findings

The appointment management system can be accessed through desktop and mobile browsers. It includes patient accounts, and accounts for doctors and specialists. The system has a forecasting tool that utilizes the XGBoost model to generate future patient arrival forecasts. The evaluation of the system, using ISO 25010 as a reference and measurement tool, resulted in positive scores in most areas. It indicates its effectiveness in managing appointments and providing a responsive, user-friendly, secure, maintainable, and portable software solution. However, there is still room for improvement in terms of reliability and the accuracy of its forecasting ability.

1. The features of the system are responsive and can be accessed through desktop and mobile browsers, allowing the users in different devices to navigate through the system more easily. The system includes patients and accounts for the doctors and

specialists. On the appointment page, they will need to register first to have an account after that they can now login and book appointments. The doctors and specialists page includes the scheduling of appointments of the patients and also the schedule of their arrivals in the hospital. All the information that is put in by the user is kept in the database for keeping the information organized and stored. It also has a forecasting system that is put to use in scheduling tools.

2. The system can manage the patient appointments by giving them an account to create and they can start to fill up the information on the registration page. The administrator can authorize the schedules of the doctors and specialists if they need to change the schedules and both of them can change it. The administrator will set and provide the given schedules of doctors and specialists and it includes the patient's page the admin will see the changes in the schedules in the appointment page. After the patient booked an appointment, the doctor will see the appointment of the patient, and also the admin will see it as well. The doctor can set their available time or schedule their appointments.
3. The time series data consisted of 958 daily observations of outpatient arrival counts from May 18, 2020 to December 31, 2022, in Skyline Hospital and Medical Center. The XGBoost model, fed with lag features and time series features, was utilized to generate future patient arrival forecasts 180 days ahead, which is from December 31, 2022 to June 29, 2023.
4. The appointment management system underwent an evaluation using ISO 25010 as a reference and measurement tool. The evaluation covered different aspects of software quality, resulting in the following scores: Functional Suitability with 3.78,

Performance Efficiency with 3.60, Compatibility with 3.48, Usability with 3.54, Reliability with 3.48, Security with 3.74, Maintainability with 3.58, Portability with 3.83, and Forecasting Ability with 3.5. The system received positive evaluations in most areas, indicating its effectiveness in managing appointments and providing a responsive, user-friendly, secure, maintainable, and portable software solution. However, there is room for improvement in terms of reliability and the accuracy of its forecasting ability.

5.2 Conclusions

The development and implementation of an appointment management web application system, with a time series forecasting feature for future outpatient arrivals was a success. The appointment management system provided the necessary features for the patient users, doctor users, and the admin user. Patient users are able to schedule appointments, check appointments history, view notifications, and update profile information. Doctor users can view the booked appointments made by the patients to them, check notifications, set available times, schedules, and edit profile information. The administrator user has full control of the system, and is responsible for managing patient and doctor accounts, appointments, schedules, and viewing the time series forecasting dashboard. With the utilization of the ISO 25010 software standard criteria, the system was tested with decent results, though significant improvements for the compatibility and reliability criteria should be considered. Finally, time series forecasting of future patient arrivals in the outpatient unit was successful with a forecast horizon of up to 180 days.

1. The appointment system was able to provide the features and functionality needed for the system, it also has the time series forecasting model, allowing the hospital to know daily future patient arrivals. The system has the administrator to control the system functionality and they handle the patient's appointments, doctor and specialist's accounts and schedules. Additionally, the appointment system offers convenience for the patients as they can make an appointment from the comfort of their own home, eliminating the need for long wait times over the phone or in person. The system also notify the patients of any upcoming appointment, reducing the likelihood of missed appointments and allowing for efficient use of healthcare resources. The time series forecasting feature also enables the hospital to monitor trends and anticipate busy periods, allowing for the proper allocation of resources and staff.
2. The appointment management system manages the scheduling of the patient users and doctor users in the system with the administrator. The patient information is included once they are booked for their appointments and their preferred schedule on their appointments. The notification system allowed users in the system to be alerted whenever there are changes and appointment cancellations that will occur.
3. The data analytics dashboard with time series forecasting provides administrators with valuable insights that can improve operational efficiency and outpatient management. Additionally, utilizing time series forecasting makes the hospital informed about the future number of patient arrivals. This has potential to improve decision-making, resource allocation, and better hospital experience overall.

4. The appointment system was evaluated using the ISO 25010 via survey implementing the User Acceptance Test. The assessment scored an overall 3.50 average which represents an ‘agree’ evaluation from the audience. The Portability criteria received the highest score which shows that the system can adapt to different devices and is easy to access. On the other hand, the Reliability and Compatibility criteria gained the lowest scores in which the system needs to be improved in its responsiveness and error handling capabilities.

5.3 Recommendations

This section presents the recommendations suggested to further enhance the overall functionality of the system in accordance with its scope objectives and limitations. With the conclusions and findings demonstrates that the study can be improved, these recommendations are offered:

1. The system has features for appointment booking in the outpatient department. However, it is possible also to improve it by including a payment system and the admissions of patients in the hospital. Push notifications can be included in the system; it can help notify the users by reminding them of their appointments. The system has only a basic notification which only retrieves notifications in the database.
2. The management of an appointment system can improve and analyze data on collecting data of the patients’ preferences. Collecting and analyzing data can improve the system to optimize the scheduling efficiency of the system and also

user satisfaction. Appointment system can benefit the user also by including the payment and the alert so it can provide another assistance provided by the system. Also the feedback for patients can be included in the appointment system so it can improve and gather feedback on the users how they use the system.

3. Gather more time series data of patient arrivals. Because the more observations the data has, the higher the probability of making more accurate forecasts. The second recommendation would be to have a better and higher quality of gathered data. The time series data from 2020-05-18 (May 18, 2020) to 2022-12-31 (December 31, 2022) suffered from several outliers, and 108 missing observations. Having high-quality data can have a positive significant impact on the forecasting model's performance, and will make simpler forecasting models feasible and effective for implementation. The third recommendation is that instead of just gathering the total patient arrival count in time series for the whole outpatient unit, request patient arrival data across every sub-department in the outpatient unit (e.g. ENT, Dermatology, Orthopedics, etc.).
4. The system was tested using the ISO 25010 showing effectiveness in the criteria Functional Suitability, Performance Efficiency, Usability, Security, Maintainability, Portability, and Forecasting Ability. However, in terms of Compatibility, the appointment system needs to be more responsive in mobile and desktop browsers as well as its performance issues. Furthermore, its Reliability needs to improve in regards to the system’s availability, and its consistency to perform without errors and recover from it.

References

Agrawal, M. *Online doctor appointments: Convenient Virtual doctor visits available now.* (2022). Retrieved from https://doi.org/10.55041/ijsrem12512

Akhtar, I. (2016). *Research Design. SSRN Electronic Journal*. Retrieved from https://doi.org/10.2139/ssrn.2862445

Akshay, S, A. B., Alagappan, R., & Gnanavel, S. (2019). *BOOKAZOR - an Online Appointment Booking System*. In 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN). Retrieved from https://doi.org/10.1109/vitecon.2019.8899460

Ali, A., & Ahmad, A. (2011). *Designing an Appointment Management System for the Mother and Child Health Department of the Klinik Kesihatan Changlun*. Retrieved from https://etd.uum.edu.my/2817/

Alim, M. A., Ye, G., Guan, P., Huang, D., Zhou, B., & Wu, W. (2020). *Comparison of ARIMA model and XGBoost model for prediction of human brucellosis in mainland China: a time-series study*. BMJ Open, 10(12), e039676. Retrieved from https://doi.org/10.1136/bmjopen-2020-039676

Antonini, C. (2023). *Vue.js as an enterprise solution*. Positive Thinking Company. Retrieved https://positivethinking.tech/insights/vue-js-as-an-enterprise-solution/

Balba, N. P., Rosas, N. M. F., Drilon, P. Z., Fallaria, S. J. F., & Fallaria, J. F. (2019). *ISAKAY: Android Based Booking System for Tri-Bike Operators and Drivers Association with Cloud-Based Data Analytics*. International Journal of Innovative Technology and Exploring Engineering, 8(10), 1265–1269. Retrieved from https://doi.org/10.35940/ijitee.i7535.0881019

Batoon, J. A., Flores, G. G., Jade, M., Lorenzo, P., Jude, F., Noel, K., Samson, D. C., Aaliyah, J., & Angeles, L. (2022). *Development of a maternity clinic information management system*. International Journal of Advanced Trends in Computer Science and Engineering, 11(2), 71–76. Retrieved from https://doi.org/10.30534/ijatcse/2022/061122022

B., R. (2022, October 24). *What is MySQL: MySQL Explained For Beginners*. Hosting Tutorials. Retrieved from https://w ww.hostinger.ph/tutorials/what-is-mysql

Cox, L. K. (2021, July 21). *Web Design 101: How HTML, CSS, and JavaScript Work*. Retrieved from https://blog.hubspot.com/marketing/web-design-html-css-javascript

Christie, T. (n.d.). *Home - Django REST framework*. Retrieved from https://www.django-rest-framework.org/

Cline, B. (2021, January 19). *Bootstrap Logo*. Brian Cline. Retrieved from https://www.brcline.com/blog/how-to-use-bootstrap-in-visualforce/bootstrap-logo

Di Maio, L. (2021, June 14). *PostgreSQL Consulting & Managed Services | Digitalis*. digitalis.io. Retrieved from https://digitalis.io/postgresql-services/

Fang, Z., Yang, S., Lv, C., An, S., & Wu, W. (2022). *Application of a data-driven XGBoost model for the prediction of COVID-19 in the USA: a time-series study*. BMJ Open, 12(7), e056685. Retrieved from https://doi.org/10.1136/bmjopen-2021-056685

German, J. D., Mina, J. K. P., Alfonso, C. M. N., & Yang, K. H. (2018). *A study on shortage of hospital beds in the Philippines using system dynamics*. 2018 5th International

Conference on Industrial Engineering and Applications (ICIEA). Retrieved from https://doi.org/10.1109/iea.2018.8387073

Gumah, W. (2018). *Zer0Clock: An online appointment booking system*. Retrieved from http://hdl.handle.net/20.500.11988/423

Hewasinghe, N. (2021). *Web Based Appointment and Patient Management System for Dermatology Clinic at Teaching Hospital Ragama*. Retrieved from https://dl.ucsc.cmb.ac.lk/jspui/handle/123456789/4346

Hidalgo, E. S. (2019). *Adapting the scrum framework for Agile Project Management in science:Case study of a distributed research initiative*. Heliyon, 5(3). Retrieved from https://doi.org/10.1016/j.heliyon.2019.e01447

Idowu, A. P., Adeosun, O. O., & Williams, K. O. (2014). *Dependable Online Appointment Booking System for Nhis Outpatient in Nigerian Teaching Hospitals*. International Journal of Computer Science and Information Technology, 6(4), 59–73. Retrieved from https://doi.org/10.5121/ijcsit.2014.6405

Kadri, F., Harrou, F., Chaabane, S., & Tahon, C. (2014). *Time Series Modelling and Forecasting of Emergency Department Overcrowding*. Journal of Medical Systems, 38(9). Retrieved from https://doi.org/10.1007/s10916-014-0107-0

Khan, M. M., & Karim, R. (2020). *Development of Smart e-Health System for Covid-19 Pandemic*. 2020 23rd International Conference on Computer and Information Technology (ICCIT). Retrieved from https://doi.org/10.1109/iccit51783.2020.9392743

Malik, S., Bibi, N., Khan, S., Sultana, R., & Rauf, S. A. (2017). *Mr. Doc: A doctor appointment application system. arXiv preprint*. Retrieved from https://doi.org/10.48550/arXiv.1701.08786

Mandava, M., Lubamba, C., Ismail, A., Bagula, A., & Bagula, H. (2016). *Cyber-healthcare for public healthcare in the developing world*. 2016 IEEE Symposium on Computers and Communication (ISCC). Retrieved from https://doi.org/10.1109/iscc.2016.7543707

McKinney, W. (2017). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython (2nd ed.)*. O'Reilly Media.

Mehta, N., & Pandit, A. (2018). *Concurrence of big data analytics and healthcare: A systematic review*. International Journal of Medical Informatics, 114, 57–65. Retrieved from https://doi.org/10.1016/j.ijmedinf.2018.03.013

Mendoza, S., Padpad, R. C., Vael, A. J., Alcazar, C., & Pula, R. (2019). *A web-based "InstaSked" appointment scheduling system at Perpetual Help Medical Center outpatient department*. World Congress on Engineering and Technology; Innovation and Its Sustainability 2018, 3–14. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython (2nd ed.)* Retrieved from https://doi.org/10.1007/978-3-030-20904-9_1

Osunade, O., Osho, A., & Oyebamiji, S.O. (2014). *ANDROID APPOINTMENT MANAGER APPLICATION DEVELOPMENT WITH GOOGLE APIs*. Retrieved from http://www.tjournal.org/tjst_april_2014/06.pdf

Our Django Python expertise at your service. (2021, October 12). *Ubidreams.* Retrieved https://ubidreams.fr/en/expertises/development/our-django-python-expertise-at-your-service

Project Management Methodologies - Everything You Need to Know. (n.d.). Retrieved November 11, 2022. Retrieved from https://www.teamwork.com/project-management-guide/project-management-methodologies/

Raghupathi, W., & Raghupathi, V. (2014). *Big data analytics in healthcare: promise and potential*. Health Information Science and Systems, 2(1). Retrieved from https://doi.org/10.1186/2047-2501-2-3

Reddy, G. C. (2021, July 27). *Python Language.* Retrieved from https://www.gcreddy.com/2021/07/python-language-syntax.html

Sharma, A. (2021). *Online Doctor Appointment System*. Retrieved from http://103.47.12.35/bitstream/handle/1/1378/1713104025_AKASH%20SHARMA_Finalprojectreport.pdf%20-%20Akash%20Sharma.pdf?sequence=1&isAllowed=y

Shelwante, S., Thakare, A., Sakharkar, A., Birelliwar, A., & Borkar, K. (2019). *Smart Health Doctor Appointment System.* International Journal of Research in Engineering, Science and Management, 2(2), 849–853. Retrieved from https://www.ijresm.com/Vol.2_2019/Vol2_Iss2_February19/IJRESM_V2_I2_228.pdf

Soini, P. (2021). *Implementing Appointment Scheduling and Virtual Meeting Systems*. Retrieved from

https://www.theseus.fi/bitstream/handle/10024/497602/Soini_Perttu.pdf?sequence=2&isAllowed=y

Stadler, J. G., Donlon, K., Siewert, J. D., Franken, T., & Lewis, N. E. (2016). *Improving the efficiency and ease of healthcare analysis through use of data visualization dashboards*. Big Data, 4(2), 129–135. Retrieved from https://doi.org/10.1089/big.2015.0059

Tamayo, J. E. (2018). *Development of a multi-platform outpatient appointment system with automated interactive SMS service*. Asian Journal of Business and Technology,1(1). Retrieved from https://asianjournal.org/online/index.php/ajbts/article/view/398/215

Velibor, B. O. Ž. I. Ć. (2019). *Smart hospital – our experience*. Smart Cities and Regional Development Journal, 3(2), 95–99. Retrieved from https:// doi.org/10.13140/RG.2.2.20383.43689

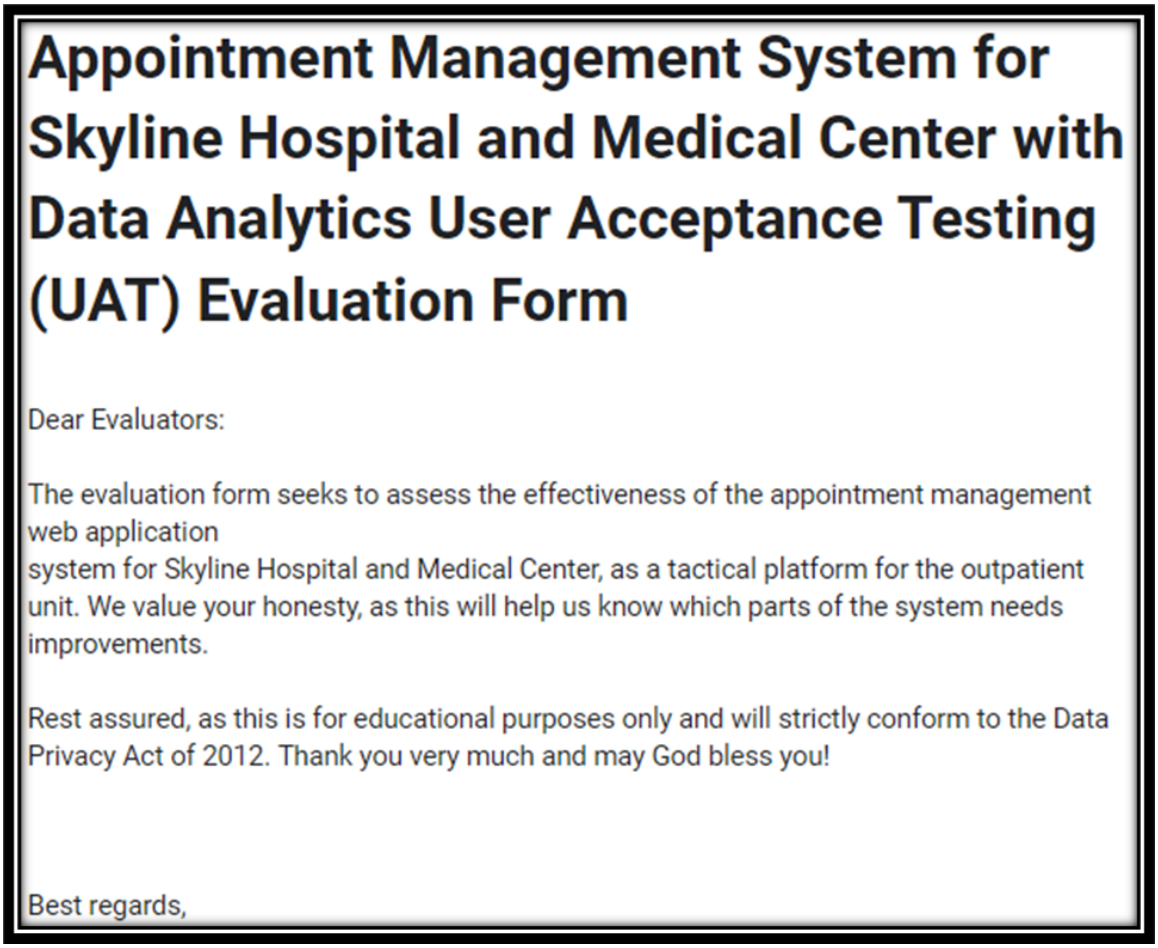
Verma, R., Sharma, J., & Jindal, S. (2020). *Time Series Forecasting Using Machine Learning*. Communications in Computer and Information Science, 372–381. Retrieved from https://doi.org/10.1007/978-981-15-6634-9_34

Weed, A. (2020, August 31). *Jupyter @ Swarthmore - Swarthmore College ITS Blog*. Swarthmore College ITS Blog. Retrieved from https://blogs.swarthmore.edu/its/2020/08/31/jupyter-at-swarthmore/

Žižović, M., Pamucar, D., Miljković, B., & Karan, A. (2021). *Multiple-criteria evaluation model for medical professionals assigned to temporary SARS-CoV-2 hospitals. De*cision Making: Applications in Management and Engineering, 4(1), 153–173. Retrieved from https://doi.org/10.31181/dmame2104153m

APPENDIX A

Google forms Survey Questionnaire



Appointment Management System for Skyline Hospital and Medical Center with Data Analytics User Acceptance Testing (UAT) Evaluation Form

Dear Evaluators:

The evaluation form seeks to assess the effectiveness of the appointment management web application
system for Skyline Hospital and Medical Center, as a tactical platform for the outpatient unit. We value your honesty, as this will help us know which parts of the system needs improvements.

Rest assured, as this is for educational purposes only and will strictly conform to the Data Privacy Act of 2012. Thank you very much and may God bless you!

Best regards,

Bajar, Jhusthine G.
4th Year Student
Bachelor of Science in Computer Engineering (BsCpE)

Britanico, Albert R.
4th Year Student
Bachelor of Science in Computer Engineering (BsCpE)

Gavan, Isabelle Therese L.
4th Year Student
Bachelor of Science in Computer Engineering (BsCpE)

Lanzuela, Daniel Mark A.
4th Year Student
Bachelor of Science in Computer Engineering (BsCpE)

Samson, Joaquin Paulo N.
4th Year Student
Bachelor of Science in Computer Engineering (BsCpE)

Dr. Nelson C. Rodelas
Thesis Adviser

University of the East - Caloocan | College of Engineering

ISO/IEC 25010 Software Standard Criteria

The following survey questions are utilized to assess software quality based on the ISO/IEC 25010 Software Standard Criteria. Given the Likert Scale below:

1 - Strongly Disagree

2 - Disagree

3 - Neutral

4 - Agree

5 - Strongly Disagree

Choose the score that best evaluates the system.

This is a sample question for interacting with the linear scale (It is not necessary to answer this question, you may skip it).

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Functional Suitability

1. The appointment management system provides all the necessary features I need for managing appointments. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

2. The appointment management system processes data accurately and produces correct results. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

3. The appointment management system's features are appropriate for its intended purpose. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Performance Efficiency

4. The appointment management system responds quickly to user actions and requests. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

5. The appointment management system utilizes system resources efficiently. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

5. The appointment management system utilizes system resources efficiently. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

6. The appointment management system needs further optimization. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Compatibility

7. The appointment management system is responsive in both mobile and desktop browsers. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

8. The appointment management system causes few conflicts or performance issues. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Usability

9. The appointment management system's features and functions are easy to recognize and understand. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

10. I can learn how to use the appointment management system quickly and easily. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

11. The appointment management system is easy to operate and navigate. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

12. The appointment management system helps prevent and recover from user errors. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

13. The user interface of the appointment management system is visually appealing and pleasant to use. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Reliability

14. The appointment management system is available when it's needed. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

15. The appointment management system consistently performs well without errors or crashes. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

16. The appointment management system consistently performs well without errors or crashes. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

17. The appointment management system can recover quickly from crashes or errors. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Security

18. The appointment management system protects my personal and patient information from unauthorized access. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

19. The appointment management system ensures the integrity of data by preventing unauthorized modification. *

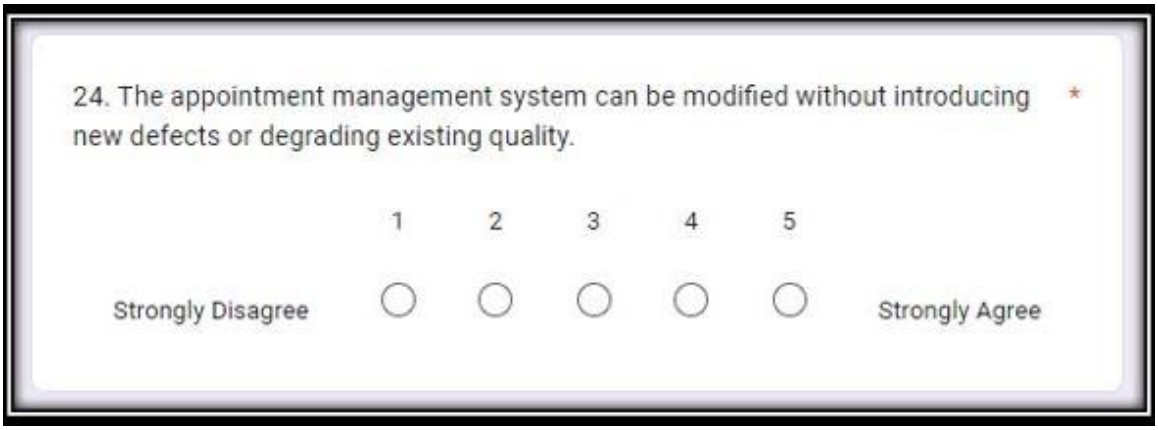
	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

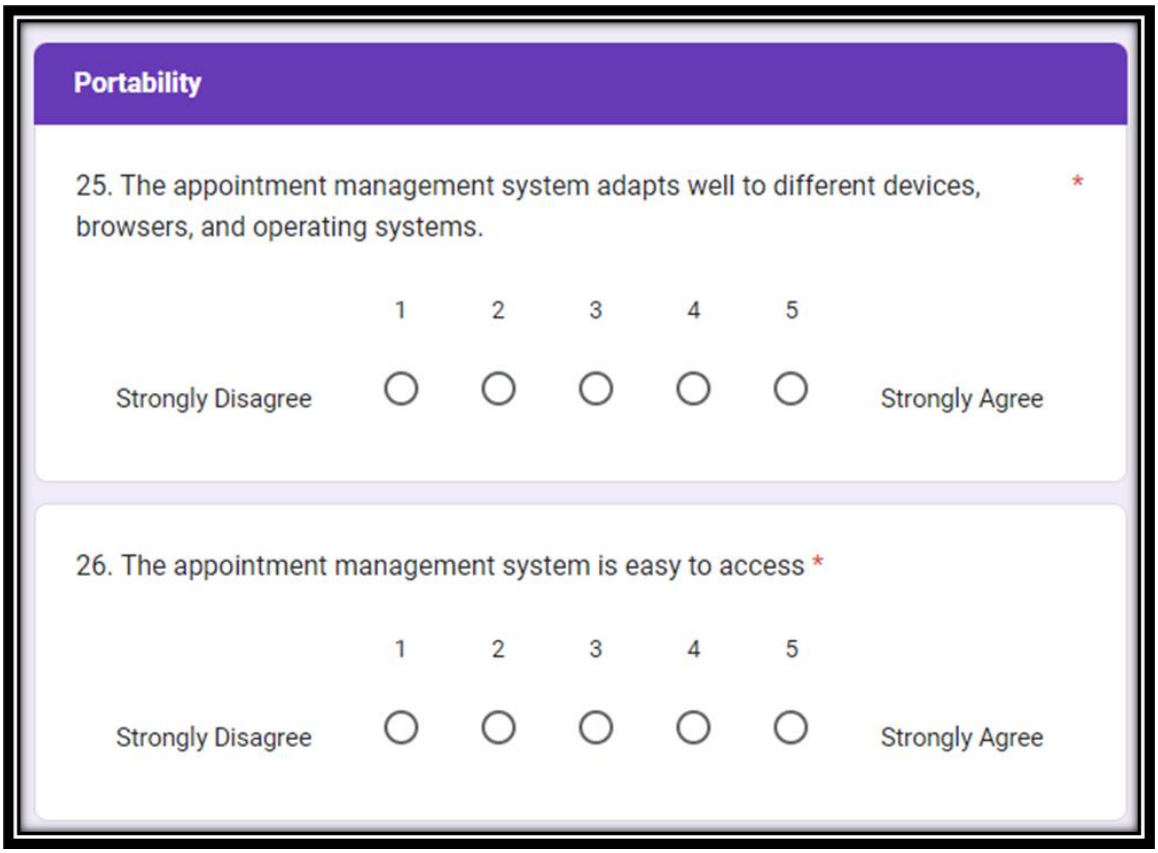
20. The appointment management system verifies the identity of users before granting access to sensitive information. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

21. The appointment management system verifies the identity of users before granting access to sensitive information. *

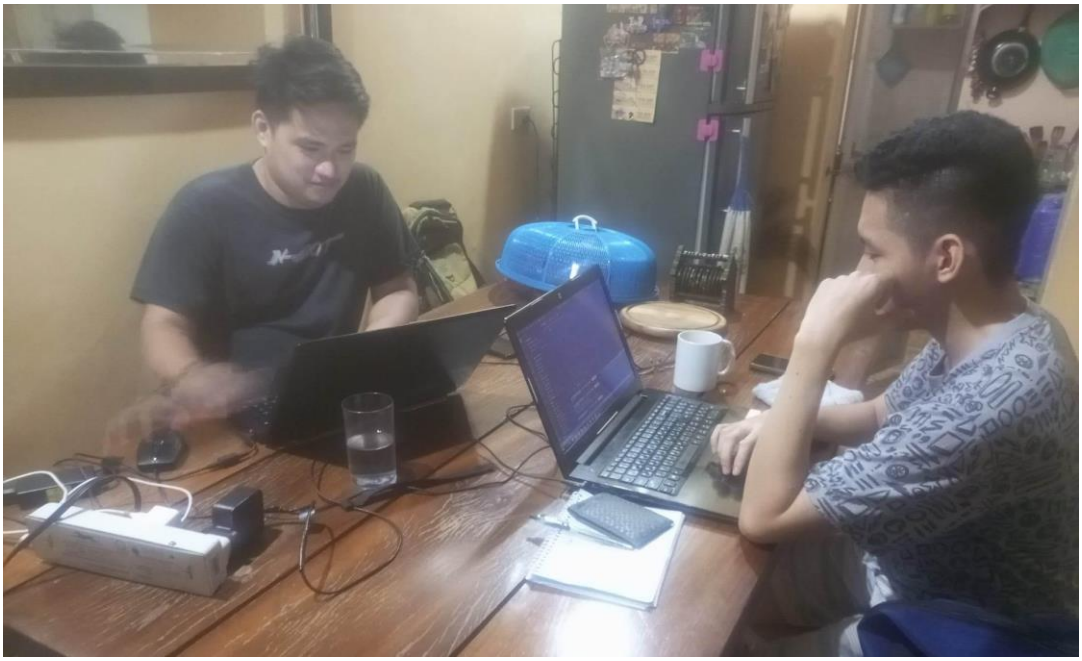
	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree


24. The appointment management system can be modified without introducing new defects or degrading existing quality. *
1 2 3 4 5
Strongly Disagree
Strongly Agree


Portability
25. The appointment management system adapts well to different devices, browsers, and operating systems. *
1 2 3 4 5
Strongly Disagree
Strongly Agree
26. The appointment management system is easy to access *
1 2 3 4 5
Strongly Disagree
Strongly Agree

APPENDIX B

Pictures at Work



Web Application Development

Data Gathering

Data Analytics Development

APPENDIX C

Project Costs

PROJECT COST			
Description	Quantity	Price	Sub -Total
HOSTING (AWS)	1	55.53	55.53
DOMAIN NAME (GO DADDY)	1	119	119
TOTAL			174.53

APPENDIX D

Turnitin Results

turnitin

Similarity Report ID: oid:27881:35497995

16% Overall Similarity

Top sources found in the following databases:

- 7% Internet database
- Crossref database
- 13% Submitted Works database
- 4% Publications database
- Crossref Posted Content database

APPENDIX E

Forecasting Source Code

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from plotly.subplots import
        make_subplots
import plotly.graph_objs as go
color_pal = sns.color_palette()
plt.style.use('ggplot')
plt.style.use('fivethirtyeight')

import warnings
warnings.filterwarnings('ignore')

import xgboost as xgb
from sklearn.metrics import
        mean_squared_error,
        mean_absolute_error
df =
        pd.read_csv("skyline_hospital_ti
        me_series_patient_arrivals.csv",
        index_col=['Date'],
        parse_dates=True)
# Filter data to include only records
        starting from May 18, 2020, and
        onwards
start_date = "2020-05-18"
df = df.query(f"index >= '{start_date}'")
new_index = pd.date_range(start='2020-
        05-18', end='2022-12-31',
        freq='D')
df = df.reindex(new_index)
# Create a plot using plotly express
fig = px.line(df, x=df.index, y='Patients',
        title='Daily Patient Arrivals')

# Set plot axis labels
fig.update_layout(xaxis_title='Date',
        yaxis_title='Patient Count')
```

```
# Show plot
fig.show()

df = df.interpolate(method='linear')
#print(df.to_string())

# round up floating point interpolated
    values
new_values =
    (np.ceil(df.values)).astype(int)
df['Patients'] = new_values
print(df.to_string())

# Create a plot using plotly express
fig = px.line(df, x=df.index, y='Patients',
    title='Daily Patient Arrivals')

# Set plot axis labels
fig.update_layout(xaxis_title='Date',
    yaxis_title='Patient Count')

# Show plot
fig.show()

from statsmodels.tsa.seasonal import
        STL

patients_series = df['Patients']
patients_series

## Calculate the IQR and bounds
Q1 = patients_series.quantile(0.25)
Q3 = patients_series.quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 0.1 * IQR
upper_bound = Q3 + 1.5 * IQR

print(f'Lower Bound: {lower_bound}')
print(f'Upper Bound: {upper_bound}')

def custom_winsorize(x):
    if x < lower_bound:
        return lower_bound
    elif x > upper_bound:
        return upper_bound
```

```
    else:
        return x

cleaned_series = patients_series.copy()
cleaned_series =
cleaned_series.apply(custom_winsorize)
df['Patients'] = cleaned_series.copy()

fig, ax = plt.subplots(figsize=(12, 6))
df.plot(ax=ax)

# Create a figure and axis
fig, ax = plt.subplots(figsize=(12, 6))

# Plot the original patients series
ax.plot(df.index, df['Patients'],
label='Patient Count', alpha=0.7)

# Set the title and labels for the plot
ax.set_title('Winsorized Patient
Arrivals')
ax.set_xlabel('Date')
ax.set_ylabel('Patients')

# Show the legend
ax.legend()

# Display the plot
plt.show()

def create_features(df, label=None):
    """
    Create time series features from the
datetime index
    """
    df = df.copy()
    print(df)
    df['date'] = df.index
    df['day_of_week'] =
df['date'].dt.dayofweek
    df['month'] = df['date'].dt.month
    df['day_of_year'] =
df['date'].dt.dayofyear

    window_size = 7  # 7-day rolling
window
```

```
    # Rolling minimum
    df['rolling_min'] =
df['Patients'].rolling(window=window_si
ze).min()

    # Rolling maximum
    df['rolling_max'] =
df['Patients'].rolling(window=window_si
ze).max()

    # Rolling sum
    df['rolling_sum'] =
df['Patients'].rolling(window=window_si
ze).sum()

    # Rolling mean
    df['rolling_mean'] =
df['Patients'].rolling(window=window_si
ze).mean()

    # Rolling median
    df['rolling_median'] =
df['Patients'].rolling(window=window_si
ze).median()

    # Rolling standard deviation
    df['rolling_std'] =
df['Patients'].rolling(window=window_si
ze).std()

    # Rolling quantile (25th percentile)
    quantile_25 = 0.25
    df['rolling_quantile_25'] =
df['Patients'].rolling(window=window_si
ze).quantile(quantile_25)

    # Rolling quantile(50th percentile)
    quantile_50 = 0.50
    df['rolling_quantile_50'] =
df['Patients'].rolling(window=window_si
ze).quantile(quantile_50)

    # Rolling quantile (75th percentile)
    quantile_75 = 0.75
```

```
    df['rolling_quantile_75'] =
df['Patients'].rolling(window=window_si
ze).quantile(quantile_75)

    return df

df = create_features(df)


from sklearn.model_selection import
TimeSeriesSplit
tss = TimeSeriesSplit(n_splits=8,
test_size=119, gap=1)
df = df.sort_index()
fig, axs = plt.subplots(8, 1, figsize=(15,
15), sharex=True)

fold = 0
for train_index, val_index in tss.split(df):
    train = df.iloc[train_index]
    test = df.iloc[val_index]
    train['Patients'].plot(ax=axs[fold],
                           label='Training Set',
                           title=f'Train / Test Split
Fold {fold}')
    test['Patients'].plot(ax=axs[fold],
label='Test Set')

    axs[fold].axvline(test.index.min(),
color='black', ls='--')
    fold += 1


def add_lags(df):
    target_map = df['Patients'].to_dict()
    df['lag1'] = (df.index -
pd.Timedelta('30
days')).map(target_map)
    df['lag2'] = (df.index -
pd.Timedelta('90
days')).map(target_map)
    df['lag3'] = (df.index -
pd.Timedelta('180
days')).map(target_map)
    return df

df = add_lags(df)
```

```
def root_mean_squared_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.sqrt(np.mean((y_true - y_pred)**2))

def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

def symmetric_mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(2 * np.abs(y_true - y_pred) / (np.abs(y_true) + np.abs(y_pred))) * 100

from sklearn.model_selection import RandomizedSearchCV

# ROUND 1

# Create the XGBoost Regressor object
reg = xgb.XGBRegressor(objective='reg:squarederror')

# Define the parameter grid
param_grid = {
    'n_estimators': [100, 500, 1000, 2000, 3000],
    'max_depth': [3, 5, 7, 9],
    'min_child_weight': [1, 3, 5, 7],
    'gamma': [0, 0.1, 0.2, 0.3, 0.4],
    'subsample': [0.5, 0.6, 0.7, 0.8, 0.9],
```

```
    'colsample_bytree': [0.5, 0.6, 0.7, 0.8,
0.9],
    'learning_rate': [0.01, 0.05, 0.1, 0.2]
}

# Initialize the RandomizedSearchCV
object
random_search =
RandomizedSearchCV(estimator=reg,
param_distributions=param_grid,
scoring='neg_mean_squared_error',
                           cv=3, verbose=2,
n_jobs=-1, n_iter=50, random_state=42)

# Fit the RandomizedSearchCV object to
your training data
random_search.fit(X_train, y_train)

# Get the best parameters and the best
score
best_params =
random_search.best_params_
best_score = random_search.best_score_

print(f'Best Params: {best_params}')
print(f'Best Scores: {best_score}')

# ROUND 2

# Create the XGBoost Regressor object
reg =
xgb.XGBRegressor(base_score=0.5,
booster='gbtree',
                    n_estimators=300,

early_stopping_rounds=50,
                    tree_method='hist',

objective='reg:squarederror',
                    max_depth=3,
                    min_child_weight=7,
                    gamma=0.25,
                    learning_rate=0.05,
                    colsample_bytree=0.9,
                    subsample=0.8,
                    reg_lambda=2,
alpha=1)
```

```
                                                    cv=10, verbose=2,
# Define the parameter grid                         n_jobs=-1, n_iter=300,
param_grid = {                                      random_state=43)
    'n_estimators': [200, 300, 400, 500,
900],                                               # Fit the RandomizedSearchCV object to
    'learning_rate': [0.01, 0.05, 0.1, 0.2,         your training data
0.3],                                               random_search.fit(X_train, y_train)
    'gamma': [0, 0.1, 0.25, 0.5, 1],
    'max_depth': [3, 4, 5, 6],                      # Get the best parameters and the best
    'min_child_weight': [1, 3, 5, 7],               score
    'subsample': [0.5, 0.6, 0.7, 0.8, 0.9, 1],      best_params =
    'colsample_bytree': [0.6, 0.7, 0.8, 0.9,        random_search.best_params_
1],                                                 best_score = random_search.best_score_
    'reg_lambda': [0, 0.5, 1, 1.5, 2],              print(f'Best Params:\n{best_params}')
    'reg_alpha': [0, 0.5, 1, 1.5, 2]                print(f'Best Scores:\n{best_score}')
}


# Initialize the RandomizedSearchCV                 tss = TimeSeriesSplit(n_splits=8,
object                                              test_size=119, gap=1)
random_search =                                     df = df.sort_index()
RandomizedSearchCV(estimator=reg,
param_distributions=param_grid,                     fold = 0
scoring='neg_mean_squared_error',                   preds = []
```

```
rmse_scores = []
mae_scores = []
mape_scores = []
smape_scores = []

# Initialize an empty DataFrame to store
actual and predicted values
actual_vs_pred_df = pd.DataFrame()

for train_index, val_index in tss.split(df):
    train = df.iloc[train_index]
    test = df.iloc[val_index]

    train = create_features(train)
    test = create_features(test)

    FEATURES = ['day_of_year',
'day_of_week', 'month', 'rolling_min',
'rolling_max',
                'rolling_sum',
'rolling_quantile_25',
'rolling_quantile_50',
'rolling_quantile_75',
                'rolling_mean',
'rolling_median', 'rolling_std', 'lag1',
'lag2', 'lag3']

    TARGET = 'Patients'

    X_train = train[FEATURES]
    y_train = train[TARGET]
    X_train = X_train.loc[:,
~X_train.columns.duplicated()]

    X_test = test[FEATURES]
    y_test = test[TARGET]
    X_test = X_test.loc[:,
~X_test.columns.duplicated()]

    reg =
xgb.XGBRegressor(base_score=0.5,
booster='gbtree',
                       n_estimators=900,
```

```
early_stopping_rounds=50,
                tree_method='hist',
objective='reg:squarederror',
                max_depth=3,
                min_child_weight=3,
                gamma=0,
                learning_rate=0.01,
                colsample_bytree=0.9,
                subsample=0.7,
                reg_lambda=0)

    reg.fit(X_train, y_train,
eval_set=[(X_train, y_train), (X_test,
y_test)],
          verbose=100)

    y_pred = reg.predict(X_test)
    preds.append(y_pred)
    rmse =
root_mean_squared_error(y_test,
y_pred)
    mae = mean_absolute_error(y_test,
y_pred)
    mape =
mean_absolute_percentage_error(y_test,
y_pred)
    smape =
symmetric_mean_absolute_percentage_e
rror(y_test, y_pred)

    rmse_scores.append(rmse)
    mape_scores.append(mape)
    smape_scores.append(smape)

    # Store actual and predicted values
along with their datetime index in the
DataFrame
    temp_df = pd.DataFrame({"Actual":
y_test, "Predicted": y_pred},
index=df.iloc[val_index].index)
```

```
    actual_vs_pred_df =
pd.concat([actual_vs_pred_df, temp_df])


# Print the results
print(f'RMSE across folds:
{np.mean(rmse_scores):.3f}')
print(f'MAE across folds:
{np.mean(mape_scores):.3f}%')
print(f'MAPE across folds:
{np.mean(mape_scores):.3f}%')
print(f'sMAPE across folds:
{np.mean(smape_scores):.3f}%')


plt.figure(figsize=(15, 6))
plt.plot(actual_vs_pred_df.index,
actual_vs_pred_df['Actual'],
label='Actual')
plt.plot(actual_vs_pred_df.index,
actual_vs_pred_df['Predicted'],
label='Predicted')
plt.xlabel('Datetime')
plt.ylabel('Number of Patients')
plt.title('Actual vs Predicted Patient
Arrivals')
plt.legend()
plt.show()


# Retrain on all data
df = create_features(df)


FEATURES = ['day_of_year',
'day_of_week', 'month', 'rolling_min',
'rolling_max',
        'rolling_sum',
'rolling_quantile_25',
'rolling_quantile_50',
'rolling_quantile_75',
        'rolling_mean', 'rolling_median',
'rolling_std', 'lag1', 'lag2', 'lag3']


TARGET = 'Patients'


X_all = df[FEATURES]
y_all = df[TARGET]
```

```
reg =
xgb.XGBRegressor(base_score=0.5,
booster='gbtree',
                 n_estimators=900,

early_stopping_rounds=50,
                 tree_method='hist',

objective='reg:squarederror',
                 max_depth=3,
                 min_child_weight=3,
                 gamma=0,
                 learning_rate=0.01,
                 colsample_bytree=0.9,
                 subsample=0.7,
                 reg_lambda=0)


reg.fit(X_all, y_all, eval_set=[(X_all,
y_all)],
      verbose=100)
```

```
# Create future DataFrame
future = pd.date_range('2022-12-31',
'2023-06-28', freq='D')
future_df = pd.DataFrame(index=future)

future_df['isFuture'] = True
df['isFuture'] = False

df_and_future = pd.concat([df,
future_df])
df_and_future =
create_features(df_and_future)
df_and_future =
add_lags(df_and_future)

future_with_features =
df_and_future.query('isFuture').copy()
future_with_features


feature_importance =
pd.DataFrame(data=reg.feature_importa
nces_, index=reg.feature_names_in_,
columns=['importance'])
```

```
feature_importance

feature_importance.sort_values('importa
nce').plot(kind='barh', title='Feature
Importance')
future_with_features['pred'] =
reg.predict(future_with_features[FEAT
URES])
future_with_features.info()


# Create a new figure
fig = go.Figure()


fig.add_trace(go.Scatter(x=future_with_
features.index,
y=future_with_features['pred'],
                name='Predicted Patient
Arrivals',
                line=dict(color='blue',
width=2),

hovertemplate='<b>Date</b>: %{x|%m-
%d-%Y}<br>' +
'<b>Patients</b>: %{y}<br>' +
                '<b>Day of
Week</b>: %{x| %A}<extra></extra>'))


fig.update_layout(
   title='Future Forecasts of Patient
Arrivals',
   xaxis=dict(
      rangeslider=dict(visible=True),
      type='date'
   ),
   yaxis_title='Predicted Patient
Arrivals',
   template='plotly_white'
)


fig.update_xaxes(showgrid=True,
gridwidth=1, gridcolor='lightgrey')
fig.update_yaxes(showgrid=True,
gridwidth=1, gridcolor='lightgrey')


fig.show()
```

```
# Save the model
reg.save_model('model.json')

# Reload the model
reg_new = xgb.XGBRegressor()
reg_new.load_model('model.json')

future_with_features['pred'] =
reg_new.predict(future_with_features[F
EATURES])

print(np.ceil(future_with_features['pred']
).astype(int))
# Create a new figure
fig = go.Figure()

fig.add_trace(go.Scatter(x=future_with_
features.index,
y=future_with_features['pred'],
                name='Predicted Patient
Arrivals',
                line=dict(color='blue',
width=2),
hovertemplate='<b>Date</b>: %{x|%m-
%d-%Y}<br>' +
'<b>Patients</b>: %{y}<br>' +
                '<b>Day of
Week</b>: %{x| %A}<extra></extra>'))

fig.update_layout(
   title='Future Forecasts of Patient
Arrivals',
   xaxis=dict(
      rangeslider=dict(visible=True),
      type='date'
   ),
   yaxis_title='Predicted Patient
Arrivals',
   template='plotly_white'
)
```

```
fig.update_xaxes(showgrid=True,
gridwidth=1, gridcolor='lightgrey')
fig.update_yaxes(showgrid=True,
gridwidth=1, gridcolor='lightgrey')
fig.show()
```

Frontend - Admin dashboard

```
<template>
 <!-- contents here will be placed in MainView.vue <template #body> using the RouterView -->
 <div
  class="row g-0 justify-content-between align-items-center pt-2 pb-3 w-100"
 >
  <h2 class="col-auto d-none d-sm-inline m-0">Dashboard</h2>
  <div class="col-auto">
   <button class="btn btn-sm btn-outline-primary rounded-pill px-3" disabled>
    Print report
   </button>
  </div>
 </div>

 <!-- Overview -->
 <div class="card border border-light bg-light shadow-sm">
  <div class="card-header border-0 bg-transparent">
   <h3 class="m-0">Overview</h3>
  </div>

  <div class="card-body">
   <div class="row g-0 border rounded-4 overflow-hidden">
    <!-- [x]: Componetized overview items -->
    <OverviewItem
```

```
        v-for="item, index in overviewItems"
        :key="index"
        :icon="item.icon"
        :label="item.label"
        :value="item.value"
        :stats-icon="item.statsIcon"
        :stats-value="item.statsValue"
        :stats-color="item.statsColor"
        :stats-comparison="item.statsComparison"
       />
     </div>
    </div>
  </div>

  <div class="spacer p-2"></div>

  <div class="card h-100 border border-light bg-light shadow-sm my-2">
    <div class="card-header bg-transparent border-bottom text-truncate d-flex align-items-center justify-content-between">
      <span class="fs-6 fw-semibold">Scheduled appointments</span>
      <button
       class="btn btn-sm btn-outline-primary"
       @click="resetZoom(barChart)"
      >
       Reset zoom
      </button>
    </div>
    <div class="card-body p-2">
     <div class="row g-0">
      <Bar
       ref="barChart"
       :data="chartData1"
       :options="chartOptions"
       height="250"
      />
     </div>
    </div>
```

```
  </div>

  <div class="spacer p-2"></div>

  <!-- Time series plot -->

  <div class="card h-100 border border-light bg-light shadow-sm my-2">
    <div
      class="card-header bg-transparent border-bottom text-truncate d-flex align-items-center justify-content-between"
    >
      <span class="fs-6 fw-semibold">Patient visits</span>
      <button
        class="btn btn-sm btn-outline-primary"
        @click="resetZoom(lineChart)"
      >
        Reset zoom
      </button>
    </div>
    <div class="card-body p-2">
      <div class="row g-0">
        <Line
          ref="lineChart"
          :data="chartData2"
          :options="chartOptions"
          height="250"
        />
      </div>
    </div>
  </div>

  <div class="spacer p-2"></div>

  <!-- Doughnut -->
  <div class="card h-100 border border-light bg-light shadow-sm my-2">
    <div class="card-header bg-transparent border-bottom text-truncate">
      <span class="fs-6 fw-semibold">Top services</span>
    </div>
```

```
    <div class="card-body p-2">
      <div class="row g-0">
        <Doughnut
          ref="doughnutChart"
          :data="chartData3"
          :options="doughnutChartOptions"
          height="200"
        />
      </div>
    </div>
  </div>
</template>

<script setup>
import { ref, onMounted } from "vue";
import {
  Chart as ChartJS,
  ArcElement,
  Tooltip,
  Legend,
  Colors,
  BarElement,
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
} from "chart.js";
import { Bar, Line, Doughnut } from "vue-chartjs";
import { useAuthStore } from "../../store/auth";
import axios from "axios";
import zoomPlugin from "chartjs-plugin-zoom";

import OverviewItem from "../../components/dashboard/OverviewItem.vue";

document.title = "Dashboard";
ChartJS.register(
  CategoryScale,
  LinearScale,
  BarElement,
  ArcElement,
  PointElement,
```

```
  LineElement,
  Tooltip,
  Legend,
  Colors,
  zoomPlugin
);

const emits = defineEmits(["pageLabel"]);

const auth = useAuthStore();

let barLabels = [];
let barDataset = {
  scheduled: { label: "Scheduled", data: [] },
  cancelled: { label: "Cancelled", data: [] },
  noshow: { label: "No-show", data: [] },
};
let plotLabels = [];
let plotData = [];
let doughnutLabels = [];
let doughnutData = [];

const barChart = ref("");
const lineChart = ref("");
const doughnutChart = ref("");
const overviewItems = ref([
  {
    id: 1,
    icon: "fa-user-plus",
    label: "Registered patients",
    value: "0",
    statsIcon: "",
    statsValue: "0.0%",
    statsColor: "bg-info",
    statsComparison: "",
  },
  {
    id: 2,
    icon: "fa-calendar-days",
    label: "Total appointments",
    value: "0",
    statsIcon: "",
    statsValue: "0.0%",
```

```
  statsColor: "bg-info",
  statsComparison: "",
},
{
  id: 3,
  icon: "fa-calendar-day",
  label: "Appointments today",
  value: "0",
  statsIcon: "",
  statsValue: "0.0%",
  statsColor: "bg-info",
  statsComparison: "",
},
{
  id: 4,
  icon: "fa-calendar-xmark",
  label: "Cancelled appointments",
  value: "0",
  statsIcon: "",
  statsValue: "0",
  statsColor: "bg-info",
  statsComparison: "",
},
]);

// chart data
const chartData1 = {
  labels: barLabels,
  datasets: [
    {
      label: barDataset.scheduled.label,
      backgroundColor: "#4b7bece6",
      data: barDataset.scheduled.data,
    },
    {
      label: barDataset.cancelled.label,
      backgroundColor: "#fc5c65e6",
      data: barDataset.cancelled.data,
    },
    {
      label: barDataset.noshow.label,
      backgroundColor: "#fd9644e6",
      data: barDataset.noshow.data,
    },
  ],
};
```

```
const chartData2 = {
  labels: plotLabels,
  datasets: [
    {
      label: "Patient visits",
      borderColor: "#45aaf2bf",
      backgroundColor: "#4b7bece6",
      cubicInterpolationMode:
"monotone",
      tension: 0.4,
      data: plotData,
    },
  ],
};
const chartData3 = {
  labels: doughnutLabels,
  datasets: [
    {
      backgroundColor: [
        "#fc5c65e6",
        "#fed330e6",
        "#2bcbbae6",
        "#4b7bece6",
        "#d1d8e0e6",
        "#fd9644e6",
        "#26de81e6",
        "#45aaf2e6",
        "#a55eeae6",
        "#4b6584e6",
      ],
      data: doughnutData,
    },
  ],
};
// chart options
const chartOptions = {
  responsive: true,
  maintainAspectRatio: false,
  interaction: {
    intersect: false,
  },
  plugins: {
    legend: {
      position: "bottom",
    },
    zoom: {
```

```
      pan: {
        enabled: true,
        mode: "x",
      },
      zoom: {
        wheel: {
          enabled: true,
        },
        pinch: {
          enabled: true,
        },
        mode: "x",
      },
    },
  },
};
const doughnutChartOptions = {
  responsive: true,
  maintainAspectRatio: false,
  interaction: {
    intersect: false,
  },
  plugins: {
    legend: {
      position: "right",
    },
  },
};

function resetZoom(element) {
  element.chart.resetZoom();
}

async function getOverview() {
  await Promise.all([
    axios.get("/api/analytics/registered-
patients/", {
      headers: { Authorization: "token " +
auth.token },
    }),
    axios.get("/api/analytics/total-
appointments/", {
      headers: { Authorization: "token " +
auth.token },
    }),
```

```
axios.get("/api/analytics/appointments-now/", {
    headers: { Authorization: "token " + auth.token },
  }),
  axios.get("/api/analytics/cancelled-appointments/", {
    headers: { Authorization: "token " + auth.token },
  }),
 ]).then((response) => {
  let registeredPatients = response[0].data;
  let totalAppointments = response[1].data;
  let appointmentsNow = response[2].data;
  let appointmentsCancelled = response[3].data;

  function displayStats(ovItem, responseData, valueProp, stat = "change") {
    ovItem.value = responseData[valueProp];
    ovItem.statsComparison = responseData.range;

    if (stat === "change") {
     ovItem.statsValue = responseData.percent_change;

      let change = 0;
      if (responseData.percent_change === "0.0%") {
        change = 0;
      } else if (responseData.percent_change[0] === "-") {
        change = -1;
      } else {
        change = 1;
      }
```

```
        //
        if (change === 0) {
          ovItem.statsIcon = "";
          ovItem.statsColor = "bg-info";
        } else if (change === -1) {
          ovItem.statsIcon = "fa-arrow-down";
          ovItem.statsColor = "bg-danger";
        } else {
          ovItem.statsIcon = "fa-arrow-up";
          ovItem.statsColor = "bg-success";
        }
      } else if (stat === "increase") {
        ovItem.statsValue = responseData.increase;
        ovItem.statsIcon = responseData.increase > 0 ? "fa-plus" : "";
        ovItem.statsColor =
          responseData.increase > 0 ? "bg-success" : "bg-info";
      }
    }
    // registered patients:
overviewItems.value[0]
    displayStats(
      overviewItems.value[0],
      registeredPatients,
      "total_patients",
      "change"
    );

    // total appointments:
overviewItems.value[1]
    displayStats(
      overviewItems.value[1],
      totalAppointments,
      "total_appointments",
      "change"
    );

    // appointments now:
overviewItems.value[2]
    displayStats(
      overviewItems.value[2],
```

```
    appointmentsNow,
    "appointments_now",
    "change"
  );

  // cancelled appointments: overviewItems.value[3]
  displayStats(
    overviewItems.value[3],
    appointmentsCancelled,
    "total_cancelled",
    "increase"
  );
  // console.log(response[0].data);
  // console.log(response[1].data);
  // console.log(response[2].data);
  // console.log(response[3].data);
 });
}
async function getCancelledAndNoshow() {
 await axios
   .get("/api/analytics/cancelled-and-noshow/", {
    headers: {
     Authorization: "token " + auth.token,
    },
   })
   .then((response) => {
    let data = response.data;

    data.forEach((item) => {
     barLabels.push(item.date)

barDataset.scheduled.data.push(item.scheduled)

barDataset.cancelled.data.push(item.cancelled)

barDataset.noshow.data.push(item.noshow)
    });
```

```
    barChart.value.chart.update();
  });
}
async function getPatientArrivalsForecast() {
 await axios
  .get("/api/analytics/patient-arrivals/", {
    headers: {
      Authorization: "token " + auth.token,
    },
  })
  .then((response) => {
    let v = response.data;

    v.forEach((element) => {
      plotLabels.push(element.date);

plotData.push(element.patient_count);
    });
    lineChart.value.chart.update();
  });
}
async function getTopServices() {
 await axios
  .get("/api/analytics/top-services/", {
    headers: {
      Authorization: "token " + auth.token,
    },
  })
  .then((response) => {
    let data = response.data;
    for (let val in data) {
      doughnutLabels.push(val);
      doughnutData.push(data[val]);
    }
    doughnutChart.value.chart.update();
  });
}

onMounted(async () => {
 emits("pageLabel", "Dashboard");

 // overview
```

```
  getOverview();
  // chart 1
  getCancelledAndNoshow();
  // chart 2
  getPatientArrivalsForecast();
  // chart 3
  getTopServices();
});
</script>
```

Frontend - Booking view for patient page

```
<template>
  <div class="spacer p-2"></div>
  <div class="d-flex justify-content-between">
    <h4>Appointment Booking</h4>
  </div>

  <div class="row g-0 justify-content-between align-items-center pt-2 pb-3"></div>

  <div class="spacer p-2"></div>
  <div class="bg-light border rounded-4 overflow-hidden">
    <div class="row g-0 h-100">
      <div class="col col-lg-auto c1 bg-white p-3 d-flex">
        <div class="date-picker d-inline-block">
          <VDatePicker
            ref="vDatePicker"
            v-model="currentDate"
            :min-date="minDate"
            :max-date="maxDate"
            mode="date"
            color="blue"
          />
          <div class="d-flex justify-content-end mt-2">
```

```
                <button class="btn btn-sm
btn-outline-primary m-2"
@click="gotoDate()">Today</button>

                <button
                 class="btn btn-sm btn-
primary m-2"
                 data-bs-toggle="modal"
                 data-bs-
target="#BookNow"
                 @click="cc"
                >
                 Book Appointment
                </button>
               </div>
              </div>
             </div>

             <div class="col p-3 outline"
style="min-width: 300px">
              <div class="spacer p-
2"></div>
              <div class="d-flex justify-
content-between">
               <h5>Available
doctors</h5>
              </div>
              <div class="spacer p-
2"></div>

              <div class="scrollbox"
style="height: 280px">
               <div class="scrollbox-
content">
                <!-- cards -->
                <template v-for="(doctor,
index) in Doctor_name" :key="index">
                 <!--btn-->
                 <div class="border ms-3
my-2 p-2 rounded-4 card">
                  <div class="row g-0
align-items-center overflow-hidden">
                   <div class="col-auto
text-truncate">
```

```
                    <span class="ms-2
small"> {{ doctor.user_id.first_name }}
</span>
                </div>

                <div class="col ms-3
text-center text-truncate small">08:00
AM</div>

                <!-- Specialty -->
                <div class="col ms-
text-secondary text-truncate
small">OPEN</div>
                </div>
              </div>
            </template>
          </div>
        </div>
      </div>
    </div>

    <!-- Modal Book Now -->
    <div
      id="BookNow"
      class="modal fade"
      tabindex="-1"
      aria-labelledby="modalLabel"
      aria-hidden="true"
    >
      <div class="modal-dialog">
        <div class="modal-content">
          <div class="modal-header">
            <h1 id="modalLabel"
class="modal-title fs-5">Services</h1>
            <button
              type="button"
              class="btn-close"
              data-bs-dismiss="modal"
              aria-label="Close"
            ></button>
          </div>
          <div class="modal-body">
            <div class="d-flex">
              <!-- Doctor Services -->
```

```
                <select
                 class="form-select p-2
m-2"
                 aria-label="size 3 select
example"
                 required
                 @change="zzz"
                >
                 <option selected
disabled>Appointment Type</option>
                 <option v-for="(serv,
index) in Service_type" :key="index"
:value="serv.id">
                  {{ serv.name }}
                 </option>
                </select>

                <!-- Doctor Name -->
                <select
                 class="form-select p-2
m-2"
                 aria-label="size 3 select
example"
                 required
                 @change="test"
                >
                 <option selected
disabled>Select doctor</option>
                 <option
                  v-for="(doctor, index)
in availableDoctors"
                  :key="index"
:value="doctor.user_id.id"
                 >
                  {{
doctor.user_id.first_name }} {{
doctor.user_id.last_name }}
                 </option>

                 <!-- <li v-
for="(doctor,index) in Doctor_name">
{{ doctor.user_id.first_name}} </li> -->
                </select>
               </div>
```

```
<!-- Date -->
<div class="spacer p-2"></div>
<label><b> Date </b></label>
<div class="spacer p-1"></div>
<div style="width: 200px">
  <input
    id="datePicker"
    type="date"
    class="form-control"
    :value="selectedDate"
    :min="dateInputMin"
    :max="dateInputMax"
    required
  />
</div>

<!-- Time -->
<div class="spacer p-3"></div>
<label> <b> Time </b></label>
<div class="container d-flex justify-content-evenly">
  <template v-for="(T, index) in AvailableTime" :key="index">
    <label>
      <input
        id="inlineRadio1"
        v-model="selectedTime"
        type="radio"
        name="inlineRadioOptions"
        class="form-check-input"
        aria-required="true"
        :value="(T.time_start + ").slice(0, 5)"
```

```
                    required
                    />
                    {{ (T.time_start + ").slice(0, 5) }}-{{ (T.time_end + ").slice(0, 5) }}
                  </label>
                </template>
              </div>

              <!-- Additional Information -->
              <div class="spacer p-3"></div>
              <label> Are you the patient? </label> <br /><br />

              <div class="d-flex justify-content-evenly">
                <label>
                  <input
                    class="form-check-input"
                    type="radio"
                    name="flexRadioDefault"
                    checked
                    required
                    @click.stop="okay = false"
                  />
                  Yes
                </label>
                <label>
                  <input
                    class="form-check-input"
                    type="radio"
                    name="flexRadioDefault"
                    required
                    @click.stop="okay = true"
                  />
                  No
```

```
                </label>
              </div>

              <!-- Add Additional Info-->
              <form v-if="okay">
                <hr />

                <h3>Patient Information</h3>

                <div class="spacer p-2"></div>

                <!-- Add-FIRSTNAME -->
                <div class="input-group mb-3">
                  <span class="input-group-text">FirstName</span>
                  <input
                    v-model="first_name"
                    type="text"
                    class="form-control"
                    placeholder="First Name"
                    aria-label="First Name"
                    required
                  />
                </div>
                <!-- Add-LASTNAME -->
                <div class="input-group mb-3">
                  <span class="input-group-text">LastName</span>
                  <input
                    v-model="last_name"
                    type="text"
                    class="form-control"
                    placeholder="Last Name"
                    aria-label="Last Name"
                    required
                  />
```

```
                </div>
                <!--  Add-AGE -->
                <div class="input-group mb-3">
                  <span class="input-group-text"> Mobile number </span>
                  <input
                    v-model="mobile_number"
                    type="charcter"
                    class="form-control"
                    placeholder="+63"
                    minlenght="11"
                    maxlength="11"
                    required
                  />
                </div>

                <div class="spacer p-2"></div>
              </form>
            </div>
```

```
                <div class="modal-footer">
                  <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
                  <button
                    type="button"
                    class="btn btn-primary"
                    data-bs-toggle="modal"
                    data-bs-target="#confirm_modal"
                  >
                    Confirm
                  </button>
                </div>
              </div>
            </div>

            <!--banner-->
            <div class="spacer p-2"></div>
            <div class="card border border-light bg-light shadow-sm overflow-hidden">
```

```
            <div class="bg-logo opacity-75"></div>
            <div class="card-body h-100" style="z-index: 1">
              <div class="text-center p-4" style="max-width: 24rem">
                <h5>Skyline Hospital and Medical Center</h5>

                <div class="spacer p-4"></div>
              </div>
            </div>
          </div>

          <!-- Modal -->
          <div
            id="confirm_modal"
            class="modal fade"
            tabindex="-1"
            aria-labelledby="confirmModalLabel"
            aria-hidden="true"
          >
            <div class="modal-dialog">
              <div class="modal-content">
                <div class="modal-header">
                  <h1 id="confirmModalLabel" class="modal-title fs-5">Confirmation Message</h1>
                  <button
                    type="button"
                    class="btn-close"
                    data-bs-dismiss="modal"
                    aria-label="Close"
                  ></button>
                </div>
                <div class="modal-body">Do you want to save the selected booking?</div>
                <div class="modal-footer">
                  <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">No</button>
                  <button type="button" class="btn btn-primary"
```

```
:disabled="submitted"
@click="confirm">
                    Yes
                  </button>
                </div>
              </div>
            </div>
          </div>


        <!-- Booking success/fail Modal -->
        <div class="modal fade" id="sc" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
          <div class="modal-dialog">
            <div class="modal-content">
              <div class="modal-header">
                <h5 class="modal-title" id="exampleModalLabel">Modal title</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
              </div>
              <div class="modal-body">
                ...
              </div>
              <div class="modal-footer">
                <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
                <button type="button" class="btn btn-primary">Save changes</button>
              </div>
            </div>
          </div>
        </div>

      </template>

      <script setup>
```

```
import axios from 'axios';
import { ref, shallowRef, onBeforeMount, onMounted } from 'vue';
import { useAuthStore } from '../../store/auth';
import { useUserStore } from '../../store/user';
import { useBookingStore } from '../../store/booking';
import { Modal } from 'bootstrap';

document.title = 'Appointment booking';

const emits = defineEmits(['pageLabel']);
const auth = useAuthStore();
const user = useUserStore();
const booking = useBookingStore();

const dateToday = new Date();
const vDatePicker = ref();
const selectedDate = ref(new Date());

let okay = ref(false);

const date = new Date();
let currentDate = ref(date);

let confirm_modal = null;

//Fetch API
let Specialty = ref([]);
let doctorIds = ref([]);
let Service_type = ref([]);
let Doctor_name = ref([]);

//Time
let Times = ref([]);
let AvailableTime = ref([]);
let selectedTime = ref();
//Doctors
```

```
        let availableDoctors = ref([]);

        //test
        let selectedService = null;
        let doctorsname = null;
        let doctorsId = ref();
        let check = 0;
        let my_booking = ref([]);

        let first_name = ref('');
        let last_name = ref('');
        let mobile_number = ref('');
        let submitted = ref(false);

        function test(e) {
          let val = e.target.value;

availableDoctors.value.forEach((element
) => {
            if (val == element.user_id.id) {
                doctorsname =
element.user_id.first_name +
element.user_id.last_name;
                doctorsId.value =
element.user_id.id;
              }
            }),
              (AvailableTime.value = []);
            Times.value.forEach((element)
=> {
              if
(val.includes(element.doctor_id))
AvailableTime.value.push(element);
            });

console.log(AvailableTime.value);
          }

          async function confirm() {
            submitted.value = true;
            let mobile = okay.value ?
mobile_number.value : user.mobile;
```

```
        let pt_fn = okay.value ? first_name.value : user.firstName;
        let pt_ln = okay.value ? last_name.value : user.lastName;

        let ff = null;



my_booking.value.forEach((element) => {
          if (pt_fn == element.pt_first_name) {
            ff = element.pt_first_name;
          }
        });

        // if (pt_fn != ff) {
        axios.post(
          '/api/appointments/add/',
          {
            time: selectedTime.value,
            doctor_name: doctorsname.value,
            pt_first_name: pt_fn,
            pt_last_name: pt_ln,
            mobile_number: mobile,
            for_another_person: okay.value,
            service_id: check,
            doctor_id: doctorsId.value,
            patient_id: user.id,
            date: selectedDate.value
          },
          {
            headers: {
              Authorization: 'token ' + auth.token
            }
          }
        );
        confirm_modal.hide();
        console.log(okay.value);
        console.log(check);
        console.log('success');

        console.log(pt_fn);
```

```
    console.log(ff);
  }

  function cc(ev) {
    submitted.value = false;
    let dt = currentDate.value;
    let d = dt.getDate();
    let m = dt.getMonth();
    let y = dt.getFullYear();
    let date = `${y}-${(m + 1).toString().padStart(2, '0')}-${d.toString().padStart(2, '0')}`;

    console.log(ev.target.value);
    console.log(date);
    console.log(dt);
    selectedDate.value = date < dateInputMin ? dateInputMin : date;
  }

  function zzz(ev) {
    let w = Service_type.value;
    console.log(w);
    availableDoctors.value = [];

    let doctorIds = null;
    w.forEach((x) => {
      if (x.id == ev.target.value) {
        selectedService = x.name;
        check = x.id;
        doctorIds = x.doctor_ids;
      }
    });

    console.log(w);

Doctor_name.value.forEach((element) => {
      if (doctorIds.includes(element.user_id.id))
availableDoctors.value.push(element);
    });
    console.log(doctorIds);
  }
```

```
        const minDate = shallowRef('');
        const maxDate = shallowRef('');
        let dateInputMin = '';
        let dateInputMax = '';
        function setMinMaxDate() {
          let dateMin = new Date();
          let dateMax = new Date();

dateMin.setDate(dateToday.getDate() +
1).toLocaleString();

dateMax.setDate(dateToday.getDate() +
7).toLocaleString();

          minDate.value = dateMin;
          maxDate.value = dateMax;
          dateInputMin =
getDateFormat(dateMin);
          dateInputMax =
getDateFormat(dateMax);
        }
        function getDateFormat(date) {
          let dt = new Date(date);
          let day = (dt.getDate() +
'').padStart(2, '0');
          let month = (dt.getMonth() + 1 +
'').padStart(2, '0');
          let year = dt.getFullYear();

          let dateFormat = `${year}-
${month}-${day}`;
          return dateFormat;
        }
        async function gotoDate(date) {
          selectedDate.value = date ? date
: dateToday;
          await
vDatePicker.value.move(date ? date :
new Date());
        }

        onMounted(async () => {
          emits('pageLabel', 'Booking');
```

```
        confirm_modal = new
Modal('#confirm_modal');

        setMinMaxDate();
        //
console.log(booking.allBookings);
        //
console.log(booking.getBookingDetails(
{ id: 1 }));
      });

      onBeforeMount(async () => {
        await Promise.all([
          axios.get('/api/misc/services/',
{
            headers: {
              Authorization: 'token ' +
auth.token
            }
          }),

axios.get('/api/misc/specialties/', {
            headers: {
              Authorization: 'token ' +
auth.token
            }
          }),

axios.get('/api/accounts/doctors/', {
            headers: {
              Authorization: 'token ' +
auth.token
            }
          }),

          axios.get('api/misc/schedules/',
{
            headers: {
              Authorization: 'token ' +
auth.token
            }
          }),
```

```
axios.get('api/appointments/my-bookings', {
        headers: {
          Authorization: 'token ' + auth.token
        }
      })
    ]).then((response) => {
      Service_type.value = response[0].data;
      Specialty.value = response[1].data;
      Doctor_name.value = response[2].data;
      Times.value = response[3].data;
      my_booking.value = response[4].data;
    });
  });
</script>
```

```
<style scoped>
/* View specific style */
tbody > tr {
  background-color: white !important;
  cursor: default;
}
/* Dropdown */
.downdown {
  background: transparent;
  color: black;
}
.btntime {
  background-color: transparent;
  color: black;
  border-color: black;
}
.container {
  padding-top: 20px;
  text-align: center;
  background-color: transparent;
}
</style>
```

Frontend - Doctor home page

```
<template>

    <div class="row g-0 justify-content-
between align-items-center pt-2 pb-3">

     <h2 class="col-auto d-none d-sm-
inline m-0">Home</h2>

    </div>

    <!-- Space -->
    <div class="spacer p-2"></div>

    <DoctorHomeView v-if="user.role
=== 'doctor'" />
    <PatientHomeView v-else />

</template>

<script setup>

import { ref, onMounted } from "vue";
import { useUserStore } from
"../../store/user";

import PatientHomeView from
"./home/PatientHomeView.vue";
import DoctorHomeView from
"./home/DoctorHomeView.vue";

document.title = "Home";

const user = useUserStore()

const emits =
defineEmits(["pageLabel"])

onMounted(()=>{

  emits("pageLabel", "Home")

});

</script>
```

```
<!-- DoctorHomeView -->

<template>
  <!-- Calendar -->
    <div class="bg-light border rounded-4 overflow-hidden">
    <div class="row g-0">
    <div class="col col-lg-auto c1 bg-white p-3 d-flex">
    <div class="date-picker d-inline-block">

     <VDatePicker v-model="currentDate" mode="date" color="blue" />

    <div class="d-flex justify-content-end mt-2">

        <button
          class="btn btn-sm btn-outline-primary m-2"
                @click="currentDate = date" >
Today
        </button>

    </div>
    </div>
    </div>

    <div class="col p-3 outline" style="min-width: 300px">
    <div class="h5">Upcoming Appointments</div>
    <div class="px-2 mt-3 border-bottom">
    <div class="mb-2 d-flex flex-wrap align-items-center justify-content-between">
    <div class=""></div>
    <div class="small text-muted">Appointments</div>

    </div>
    </div>
```

```
    <!--  -->
    <div class="scrollbox" style="height: 430px">
    <div class="scrollbox-content">

<template v-for="i in 1">

    <div class="py-2 mb-2">
    <div class="time-div d-flex align-items-center">
    <div class="time" >

        <span class="am-pm text-muted"></span>

    </div>

        <CustomDivider class="ms-2" />

    </div>

    <!-- Modal Info -->
<div class="modal fade" id="info" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h1 class="modal-title fs-5" id="exampleModalLabel">Modal title</h1>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        <template v-for="(info, index ) in appt " :key="index">
          {{ info.pt_first_name}}

        </template>
      </div>
      <div class="modal-footer">
```

```
      <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
      <button type="button" class="btn btn-primary">Save changes</button>
    </div>
   </div>
  </div>
</div>



<template v-for="(apt, index) in appt " :key="index" >

    <!-- cards -->
    <div class="ms-5" >
    <div class="btn border ms-3 my-2 p-2 rounded-4 card"
        @click="" >

    <div class="row g-0 align-items-center overflow-hidden">
        <div class="col-auto text-truncate">

            <img

src="../../../assets/images/avatars/patient/user-1.png"
            class="sq-sm align-middle rounded-circle bg-primary" />

            <span class="ms-2" >{{apt.pt_first_name}} {{ apt.pt_last_name }} </span>

        </div>

        <div class="col ms-3 text-secondary text-truncate small" > {{apt.time}}

        </div>
        <div class="col ms-3 text-secondary text-truncate small" >{{apt.date}}</div>
        </div>
```

```
    </div>
    </div>

</template>

    </div>

</template>

    </div>
    </div>
    </div>
    </div>

    <!--  Space -->
    <div class="spacer p-2"></div>


    <!--banner-->
    <div class="spacer p-2"></div>
    <div class="card border border-light
bg-light shadow-sm overflow-hidden">
    <div class="bg-logo opacity-
75"></div>
    <div class="card-body h-100"
style="z-index: 1">
    <div class="text-center p-4"
style="max-width: 24rem">

      <h5> Skyline Hospital and
Medical Center </h5>

    <div class="spacer p-4"></div>

    </div>
    </div>
    </div>

</template>

<script setup>
import axios from "axios";
import { useAuthStore } from
"../../../store/auth";
```

```
import { ref, onMounted,onBeforeMount
} from "vue";

import CustomDivider from
"../../../components/icons/CustomDivider
.vue";
import TableSlot from
"../../../components/slots/TableSlot.vue";
import ActionButton from
"../../../components/ActionButton.vue";

const date = new Date();
let schedule = ref()
let currentDate = ref(date);
let auth =  useAuthStore()
let appt = ref()
let appts = ref()

function selectItem(val) {
  // temp function on item click
  alert("Item #" + val + " clicked");

}
function accept() {
  // temp
  alert("Accepted");
}
function reject() {
  // temp
  alert("Rejected");
}

onMounted(() => {
  console.log("mounted: Doctor Home
view");
});

onBeforeMount(async()=>{

    await
axios.get("/api/appointments/my-
appointments", {
```

```
      headers: {
        Authorization: "token " + auth.token,
      },
    }),
    await axios.get("/api/appointments/my-appointments", {
      headers: {
        Authorization: "token " + auth.token,
      },
    })
    .then((response) => {
      appt.value = response.data
      console.log(response.data)
    });
})
</script>
```

APPENDIX F

Curriculum Vitae

Jhusthine G. Bajar

blk 5 Lot 5 Victor Homes Subd Gen t deleon Val City
bajar.jhusthine@gmail.com

Personal Information

Date of Birth:	December 17, 1999
Place of Birth:	Novaliches, Quezon City
Citizenship:	Filipino
Gender:	Male
Marital Status:	Single
Course:	Bachelor of Science in Computer Engineering

Educational Background

Elementary	**Gen T deleon Elementary School (2008 – 2012)**
Junior High School	**Our lady of Lourdes College (2012 – 2016)**
Senior High School	**University of the East-Caloocan (2016 – 2018)**
College	**University of the East-Caloocan (2018 – 2023)**

Albert R. Britanico
488 Libis Talisay Caloocan City
britanico.alber@ue.edu.ph

Personal Information

Date of Birth:	October 02, 1999
Place of Birth:	Caloocan City
Citizenship:	Filipino
Gender:	Male
Marital Status:	Single
Course:	Bachelor of Science in Computer Engineering

Educational Background

Elementary	**Barrio Obrero Elementary School (2005 – 2012)**
Junior High School	**Caloocan High School (2012 – 2016)**
Senior High School	**STI College Caloocan (2016 – 2018)**
College	**University of the East-Caloocan (2018 – 2023)**

Isabelle Therese L. Gavan
Francisco Homes City of San Jose Del Monte Bulacan
gavanisabelle@gmail.com

Personal Information

Date of Birth:	October 4, 2000
Place of Birth:	City of San Jose Del Monte Bulacan
Citizenship:	Filipino
Gender:	Female
Marital Status:	Single
Course:	Bachelor of Science in Computer Engineering

Educational Background

Elementary	**St. John School(2005 – 2012)**
Junior High School	**First City Providential College (2012 – 2016)**
Senior High School	**First City Providential College (2016 – 2018)**
College	**University of the East-Caloocan (2019 – 2023)**

Daniel Mark A. Lanzuela
59 Remigio Street, Maysilo Malabon City, NCR 1477
daniellanzuela@gmail.com

Personal Information

Date of Birth:	March 5, 2000
Place of Birth:	Manila, Philippines
Citizenship:	Filipino
Gender:	Male
Marital Status:	Single
Course:	Bachelor of Science in Computer Engineering

Educational Background

Elementary	**SME Child Development Center (2005 – 2012)**
Junior High School	**Saint Gabriel Academy (2012 – 2016)**
Senior High School	**University of the East-Caloocan (2016 – 2018)**
College	**University of the East-Caloocan (2018 – 2023)**

Joaquin Paulo N. Samson
261 Sinia St., Pulong Buhangin, Sta. Maria Bulacan
samson.joaquinpaulo@gmail.com

Personal Information

Date of Birth:	July 7, 2001
Place of Birth:	Quezon City, Manila, Philippines
Citizenship:	Filipino
Gender:	Male
Marital Status:	Single
Course:	Bachelor of Science in Computer Engineering

Educational Background

Elementary	**Angelicum College (2005 – 2012)**
Junior High School	**Mater Dei Academy (2012 – 2016)**
Senior High School	**STI College (2016 – 2018)**
College	**University of the East-Caloocan (2018 – 2023)**