MindCare: Web-Based Mental Health Management System for RHU Bulan, Sorsogon

Gizelle E. De los Santos, Melody A. Escobedo, Catherine S. Granado, Kathleen Joy J. Jetajobe

College of Information and Communications Technology
Sorsogon State University - Bulan Campus
Bulan, Sorsogon.

E-mails: delossantosgizelle@sorsu.edu.ph; escobedomelody@sorsu.edu.ph; granadocatherine@sorsu.edu.ph; jetajobekath@sorsu.edu.ph;

Executive Summary

This Capstone Project is the program's final requirement designed for students to apply and gain knowledge and skills in a particular discipline in computer technology-related programs. The RHU in Bulan, Sorsogon staff face difficulty in terms of recording, managing, tracking, appointment scheduling, and evaluation in a manual and semi-manual process. As for RHU staff under Bulan, Sorsogon, the main health unit, is currently experiencing problems in record keeping and managing patients' information which is essential for monitoring the current rate and condition of mental health concerns in Bulan. In Addition, the monitoring of patient's current mental status, wherein collaborative communication between patients is not accomplished because of the unavailability, break on an appointment, and unmatched schedule of patients or health personnel on scheduled evaluation or checkups. To address a solution to these difficulties, the proponents proposed the "Mind-care: A Web-Based Mental Healthcare Management System for RHU Bulan, Sorsogon", this system aimed to assist and centralize MHMS to health personnel, patients, and technical admin staff. The system also aimed to carry out and deliver the following features and modules related to managing, scheduling, tracking, and health personnel-to-patient collaboration, which provides efficient and systematic operation via the web: Patient Record Management; for Patient Information, Mental Health Records for Diagnosis, Evaluations, Archiving and Restoring Records, and Assessment; Appointment Booking for booking and scheduled appointments; Medicine Inventory; for stock in, stock out, release and return medicine; Report Download, PDF File. Tracking patients, testing, and Evaluating the proposed system performance, based on the ISO/IEC 9126-1 standards in terms of Functionality, Usability, and Portability of the system. In two semesters, the proponents would develop the system using a RAD Model where the flow of development focuses on its incremental and time-constrained implementation, which then enables the rapid creation of applications, involving stages such as planning requirements, system design, and implementation, ensuring that applications are developed in time mannered and accurately to meet user needs; And lastly, Object-Oriented for the analysis and design applying object-oriented programming; and Bottom-up for the development approach.

1 Introduction

This chapter discusses the proposed project's context, purpose and description, objectives, scope, and limitations to explain the importance of a mental health management system for the Rural Health Unit (RHU) of Bulan, Sorsogon.

1.1 Project Context

The mental healthcare facility of RHU Bulan, Sorsogon was entrusted with providing efficient mental healthcare-related services to the residents of Bulan, who were diagnosed and are currently struggling with mental problems. However, due to the large amount of workload that is still handled by manual processing and manual management of complex patients with diverse mental health conditions and needs, these raise a big concern for the dedicated facility, as paper-based recording, managing, booking, monitoring, and tracking consume irrelevant amount of time, which slows down work process, resulting into inefficiency of mental healthcare services and operation. Storing sensitive patient data physically raises concerns about data security and compliance with privacy regulations. Unauthorized access or loss of records can threaten patient confidentiality and lead to legal consequences.[1].

Designing a streamlined management system, that automates the record system of patients' information, diagnosis, booking, tracking, and evaluation for monitoring and identification of mental health needs, promotes efficiency and accuracy of RHU operation, as irrelevant manual work is reduced and the risk of data lost.[2] According to the commentaries of the International Journal of Eating

Disorder, The unprecedented COVID-19 crisis presents the automation of mental healthcare following the face-to-face contact restriction, and this offers rapid establishment of collaboration between patients and health personnel which delivers an efficacious service that is accessible through digital devices guided by self-help intervention. This Web-based platform is also designed to provide effective support management to the RHU mental health facility and accessible communication between residents of Bulan and RHU, as appointment booking helps diagnosed patients to reach out for mental health support in a timely organized schedule, minimizing the long wait time for both patient and health personnel, ensuring of mental healthcare quality services and patient satisfaction should be prioritized. [3] The cited study supports the presented scheduling features in this study as the framework for a Hybrid Appointment System for Patient Scheduling in Primary Healthcare Centers in Dubai to aid the irrelevant long wait time of the patients. The patients in mental health facilities can also determine the current improvement in individual mental health conditions, as patient session evaluations were being rated and classified as ill, injured, under-progress, progressive, and healthy. Furthermore, this enhances communication between patients and healthcare providers, as interactive communication of feedback on treatment progress and the outcome can serve as a visual representation of patient improvement, essential for treatment mapping and building trust in the healthcare provider[4].

Additionally, utilizing a computerized database like the Mental Health-Patient Information Management System (MH-PIMS) can aid in generating patient reports for case managers and clinicians, facilitating effective management and strategic planning [5]. By incorporating elements for tracking patient progression and generating schedules, the system can ensure accurate and timely record-keeping for monitoring client progress and generating necessary reports.

The project adopts a Rapid Application Development (RAD) methodology to ensure systematic development and implementation. Additionally, it adheres to ISO/IEC 9126-1 standards, incorporating a questionnaire to evaluate the software's quality attributes, including functionality, usability, and portability. These features ensure the robustness and effectiveness of the mental healthcare management system at RHU Bulan.

1.2 Purpose and Descriptions

The purpose of the project is to design a management system that would centralize the recording, monitoring, tracking, and appointment booking. This is to help the RHU staff deliver efficient healthcare services and optimization of staff operation, which can help the facility focus more on patients' mental health treatment plans and goals.

The project also implements a medicine inventory that can help health personnel to stock in medicine information such as quantity, batch number, batch supply, brand name, medicine name, dosage, milligram, and expiry date which enables health personnel to monitor medicine supply, and stock out expired medicine from the inventory stock.

The RHUs' problem concerning manual processes of gathering and managing records for report analysis is addressed by the proponents. The system must implement a download report of the gathered information, converting it into a PDF file. These would automatically provide a copy of the list of all possible needed data for the analysis report.

The list provided below is the beneficiaries:

- **Administrator -** Admins are responsible for the overall management of the system. Their duties include managing user accounts, patient records, appointments, medicine inventory, and generating pdf file reports. They also have the authority to archive and restore user accounts.

- **Patients -** Patients are primarily responsible for their own healthcare and appointment management. Their duties include booking appointments, accessing their patient records, and potentially reviewing treatment-related reports. Patients receive timely notifications regarding their appointments, including approvals, upcoming appointments, and approaching deadlines. Additionally, they receive notifications for any medication updates related to their treatment plan, ensuring adherence to prescribed regimens and timely intervention.
- **Health Personnel -** Health Personnel are responsible for providing patient care and managing their workload. Their duties include accessing and updating patient records, approving and viewing appointment schedules with the patients, checking and managing medicine inventory like adding medicine stock and releasing medicine, and prescribed medication to the patients. They may also generate reports related to patient progress or treatment plans and also reports from medicine inventory.

1.3 Objectives of the Study

The general objective of the proposed capstone project is to design, develop, and implement a streamlined management system that can provide efficiency of mental healthcare services and operations for Rural Health Unit (RHU) mental healthcare facilities. Specifically, the project aims to achieve the following objectives:

1. To develop a comprehensive system for managing Mental Health Records at RHU Bulan, Sorsogon. The features included are:
   (a) Patient Record Management
   (b) Health Personnel Record Management
   (c) User Management
   (d) Report Generator
2. To design an Appointment Schedule that will manage the following:
   (a) Appointment Booking
   (b) Calendar Viewing
   (c) All Scheduled Appointment Table List
3. To Track Medicine Stock and Expired Medicine

(a) Medicine Inventory

4. Evaluating the system performance using ISO/IEC 9126-1 Standard. The assessment will focus on the following:
   (a) Functionality;
   (b) Usability;
   (c) Portability;

1.4 Scope and Limitations

This proposed title Mind-care: A Web-Based Mental Healthcare Management System focuses on integrating a management system for the Rural Health Unit's (RHU) mental healthcare facility, specifically targeting patient information, monitoring, tracking, booking, recording, and generating reports. It primarily aims to enhance the delivery of health services at Bulan Rural Health Unit, facilitating the tracking of patients.

The implemented system is limited to monitoring, recording, tracking, booking, and medicine inventory. This limitation stems from the lack of expertise or specialized knowledge in other areas within the chosen field. However, the healthcare professionals at the healthcare unit possess the necessary background to provide accurate justification, evaluations, and possible diagnosis or treatment, serving as system administrators to facilitate assistance and monitor patients' mental statuses.

The report generator offers administrators the ability to download reports in two ways. They can choose to download reports for a specific barangay, allowing for a focused analysis of health data at the local level. Alternatively, they can optional to download a consolidated report encompassing all barangays listed in their patient database, providing a broader perspective for strategic planning and decision-making. This flexibility empowers administrators to tailor their data analysis to their specific needs and gain valuable insights into the health status of their patient population.

The system incorporates email verification to enhance security and functionality. Unverified patients cannot edit their information or receive notifications. This measure ensures account authenticity and prevents unauthorized access. Email verification also facilitates record retrieval, allowing RHU health personnel to efficiently access patient information. While email accounts are encouraged, patients without emails can still schedule appointments but will require additional steps, such as creating a unique email account, to enable healthcare personnel to manage their care effectively.

The system incorporates a robust notification system to keep all stakeholders informed. Admins and health personnel receive notifications for various events, including new appointment requests, appointment cancellations, upcoming appointments, and medication updates. Patients also receive notifications for medication updates, appointment approvals, cancellations, and scheduling of new assessments. All notifications are delivered via verified email accounts, ensuring timely and reliable communication.

The proposed system is limited to a web application and is not available as a mobile or desktop application. Nevertheless, the system's web responsiveness enables mobile users to access it through mobile browsers. In terms of the system's usability, guardians or parents can consider accessing patients' accounts due to the incapability of some patients caused by mental health conditions. The system did not include the previous records; it implemented records from its launch, ensuring a focus on current patient data and treatment plans.

2 Review of Related Systems

This section reviewed systems related to the project that could help in developing the “Mind-care: A Web-Based Healthcare Management System”, in Bulan Rural Health Unit (RHU). The purpose of this review was to analyze existing studies that practitioners could use as references.

THEME 1: Enhanced Security and Patient Control of Medical Records

According to the study of (Zala, K. et.al), the PRMS (Patient Medical Records Management System) strengthens the fact that electronically stored records, primarily health-related patient information are highly secure and robust compared with manually kept. The system also offers authorized access of patients to each own personal account which offers control over personal data such as health records.[6], The PRMS focuses on the security of patients’ records, highlighting the information hiding of confidential data.

The provided reference [6] can be related to the current system in terms of management system features. Through an information management system, the recordings of patients’ mental health diagnoses, evaluations, and treatments can be efficiently accessed, stored, managed, and modified through a digital system, wherein the privilege to handle the account is also issued to patients. The similar features that both systems have were users’ role type authorization, in which each users are provided by account, and that are secured with personal email and password. Both systems aim to provide centralization of patient information, medical records, and mental healthcare services, securely. However, the proposed system offers a record download, particularly of patients’ data such as patients’ list of names, address, and etc.

THEME 2: Centralized Management and Accessibility of Mental Health Data

In the study of (Kaur et.al) entitled “i-MANN: A Web-Based System for Data Management of Mental Health Research in India”, the system of management, the i-MANN, is yet a significant role in the research as a web-based system organized for data management of mental health research in India providing a central platform for data storage and analysis during mental health research [7]. The system provides an avenue for organizing and management of the overwhelming amounts of data to ensure the data is accurate, available, and safe.

The paper of [7] compared with the proposed study aimed to produce reports and analyze data for evidence-based decision-making and policy for mental health intervention to standardize and simplify the data management. The presented system report generator feature prioritizes report document modification and download, as the compilation of these evaluation reports would be the basis for medical supply and support needs for the Rural Health Unit in providing consistent and efficient healthcare services.

THEME 3: Inventory Management and Medication Safety

According to the study of (A.G. Assis et.al), to reach the most appropriate service and operation for the hospital environment, it is necessary to create a proper medicine inventory management system. In managing the medicine stocks, it is needed to classify the input of every medicine in the inventory to enable it to accomplish a single basis like demand, criticality, and monetary value that does not satisfy the solution for the problem for the hospital's medicine inventory management system. Accurate dosage tracking is key. The system stores details like drug strength and quantity, allowing calculation of total available doses. Additionally, the ability to sort by expiry ensures staff prioritizes dispensing medications closest to their expiry date. This proactive approach minimizes medication waste and safeguards patient safety. [8], the classification of medicines and materials developed according to the inventory demands allowed more efficient purchasing decisions, optimizing the stock and dosage of medicines at the hospital while optimizing the inventory manager's activities, and saving time.

Both the study of [8] and the current system aimed to monitor and track the dosage of the medicines to ensure accurate medication administration. Both also offer monitoring in the prevention of dispensing outdated drugs, which would be facilitated by RHU Health Personnel. Both studies also aimed to identify stocks to help maintain adequate supplies and avoid stock-outs. However, the current study is unique with its feature that monitors the expiry dates of medicines to prevent dispensing outdated and potentially harmful drugs.

Effective Inventory Control for Improved Patient Care

The study of (S, Krishnakumar et.al), the effective management of inventory is crucial for healthcare facilities to ensure prompt, quality patient care while reducing costs. To optimize inventory control, stock management systems have been developed but their implementation can be difficult due to a variety of factors such as incorrect data, inadequate training, and lack of standardization [9]. This study provides a comprehensive overview of the current research trends in media stock inventory management systems. The findings indicate that healthcare facilities have a key role to play in optimizing inventory control, reducing costs, and improving the quality of care by introducing efficient stock management systems. However, the implementation of an effective inventory management system requires a comprehensive understanding of the unique needs and challenges of healthcare facilities.

Previous research and the current system share a focus on monitoring medication dosages to guarantee patients receive the correct amount. Additionally, both approaches aim to prevent health personnel from administering expired medications. They also emphasize managing stock levels to avoid running out of critical supplies [9]. However, what sets the current system apart is its ability to specifically track medication expiry dates. This unique feature helps prevent the dispensing of outdated drugs that could potentially harm patients.

Inventory Optimization and Cost Reduction in Healthcare

The study of (Refonaa, et al), the management of inventories in a healthcare system must be compatible with its operations and critical characteristics, ensuring that inventory-related costs are minimized as well as the level of service is maximized while significantly reducing treatment cost and waste. Given these aspects, researchers and practitioners have developed several approaches and methodologies for modeling and analyzing different types of inventory management systems in the healthcare sector over the years [10].

The current management system builds upon previous research [10], by sharing its focus on ensuring accurate medication dosage and preventing the administration of expired drugs by nurses. Both approaches also prioritize maintaining adequate stock levels to avoid shortages. However, the current system takes a crucial step forward by incorporating the unique feature of tracking medication expiry dates. This innovation helps prevent the dispensing of outdated drugs, potentially safeguarding patients from harm.

Medication Error Reduction through Inventory Tracking

According to the study by (Anif, M et.al, 2019) to reduce medication errors it is important to track the medicine that was written in the prescription and modify the medical records of the patients. Miscommunication among physicians, nurses, and pharmacists can lead to many medication errors [11]. Medical information should be fully verified and all the errors must be eliminated. To achieve an excellent medicine management merged with patient's medical information is one of the good ways to avoid errors.

The previous system [11] and the current study both aimed to track medicine inventory in a healthcare facility to check the status of the inventory in terms of stocks of medicine in the facility. However, the current system is unique in a way that it also tracks the nearly expired medicine in the inventory it also monitors the stock in and out of the medicine, and lastly offers the release and return of medicine.

THEME 4: Streamlined Operations and Patient Conveniencee

Streamlined Data Recording and Transaction Management

In the study (Adi et.al) a pharmaceutical company is active in the fields of medicine and healthcare devices, and can often be found supplying medicinal products to physicians. However, in the study [12] there are still obstacles to the supply of medicines, such as the recording of supplies and the management of transaction data, which are still using a ledger. Moreover, staff often make errors in the recording and verification of records as well as drug inventory which frequently leads to incorrect ordering.

Both the study [12] and the current system aimed to record medical transactions of the data in the healthcare facility. Both systems also record the evaluation of patients and medicine inventory. However, the current system is limited to recording just patients' information and evaluation as well as the recording of the stocks of medicine in the inventory.

Appointment Scheduling for Improved Patient Experience

In the study entitled as Manufacturing and Service Operations Management discusses appointment scheduling with per-service level constraints as intended to provide a fair waiting experience for the customer. It takes in an appointment period and tries to make appointments as early as possible for the said period while keeping the fallback window no higher than a threshold waiting time [13]. The authors compare various scheduling systems to demonstrate that service-level-constrained system has efficiencies in terms of waiting time fairness and system efficiency.

Both the system [13] and the current study aimed to implement a scheduling process for the appointment of patients in their healthcare facility. This scheduling also lessens the queues and hassle for the patients and the health personnel of the facility. However, the current system wants to implement online scheduling with interactive features to avoid hectic schedules.

Online Appointment Scheduling for Convenience

According to the study of (Hancerliogullari, Kadir Oymen, 2022), In healthcare management, appointment scheduling is crucial but comes with specific challenges. It is important to minimize patient wait times and doctor idle time, as well as overtime and costs. A scientometric study was conducted to evaluate academic publications on appointment scheduling in healthcare units[14].

While both the existing system [14] and the current research project aim to improve patient appointment booking at the healthcare facility, they take different approaches. Both methods aim to reduce wait times and streamline the process for patients and health personnel alike. However, the current project takes this a step further by proposing the implementation of online booking with interactive features. This online system aims to eliminate hectic scheduling situations and create a smoother experience for everyone involved.

Theme 5: Quality Assessment Framework for Mental Health Services

According to the study by (Samartzis et.al, 2020) entitled "Assessing and Improving the Quality in Mental Health Services", to improve the quality of the mental health service system the researchers use ISO 9126-1. [15], this quality testing indicates that accessibility and acceptability of service indicators are important for the attractiveness of services related to their use by the population. One of the economic indicators that can affect the sustainability and availability of the service is the profitability standard which is also a factors that are eventually taken into account on todays any health policy.

Both the [15] and the current system aimed to indicate quality assessments like (1) Suitability of services, (2) Accessibility of patients to services, (3) Acceptance of services by patients, (4) Ability of healthcare professionals to provide services, (5) Efficiency of health professionals and providers, (6) Continuity of service over time, (7) Efficiency of health professionals and services, and (Safety for patients and health professionals). However, the current system is unique in offering a standardized framework that provides a clear structure for the assessment, focusing on specific software quality attributes like functionality, usability, and portability. Since ISO 9126-1 is designed for software evaluation, it ensures the focus stays on key aspects of the management system, such as user interface, responsiveness, and data security.

The review of related systems identified several studies that informed the development of ”Mindcare: Web-Based Mental Health Management System” for Bulan Rural Health Unit (RHU). The existing systems provided functionalities in areas such as mental health management, data management for research, inventory control, appointment booking, and quality progress. ”Mindcare” incorporates these functionalities while offering unique features such as psychotherapeutic tools for mental health monitoring, report generation for informed decision-making, expiry date tracking for medication safety, and online appointment booking for a smoother experience. Overall, the analysis of related systems demonstrates the potential of ”Mindcare” to improve mental healthcare delivery at RHUS Bulan.

3 Technical Background

In this chapter, the various project components are briefly outlined. Both software and hardware elements are covered, along with their functionality and application to the project.

3.1 System’s Development Specification

This section outlined the plan and model for the Web-Based Mental Health Management System that includes its operations, physical components, program specifications, services, and end-user requirements. It was designed to guide the development process and ensure that it met the unique needs of RHU Bulan’s patients.

3.1.1 Hardware Specification

The proponents utilized Windows 11 Home Intel Core i5-1135G7 with 2.40 Gigahertz clock speed, NVIDIA GeForce MX330 graphics card, and 2GB RAM. The hardware specifications used for system development were sufficient for handling a variety of tasks, and workloads and can be installed in different computer systems. These specifications efficiently ran with standard performance and good quality, enabling the proponents to work in an orderly manner on the system development. The specifications mentioned above were sufficient enough to offer the best and most efficient performance needed in developing the web-based web application.

Table 1: Table 1: Developers' Hardware Specification

HARDWARE	RECOMMENDED REQUIREMENTS
Memory	8GB RAM
Storage	1TB
Processor	Intel Core i5-1135G7 2.40GHz
System Type	64-bit operating system, x64-based processor
Graphics Card	NVIDIA GeForce MX330 2GB (Built-in)

3.1.2 Software Specification

This subsection showed the specifications of the software used by the proponents.

Table 2: Table 1: Developers' Software Specification

SOFTWARE	RECOMMENDED REQUIREMENTS
Operating System	Windows 11 Home Single Language Version: 22H2
Browser	Microsoft Edge
Integrated Development Environment	Microsoft Visual Studio V.1.82.2
Database	MySQL
Wireframe Design	Figma

The system proposed used the Windows 11 operating system and ran the localhost in the Microsoft Edge browser. For the back-end development of the system, the Integrated Development Environment (IDE) utilized was Visual Studio Code and the database used was MySQL. In creating wireframes and other

UI design templates for web page design the proponents utilized Figma, taking advantage of a free premium account for students. Additionally, the proponents employed various applications to support the timely development of the proposed system.

3.1.3 Service Specification

This subjection presented the specifications of the service used by the proponents.

PLDT functions as the internet service provider (ISP). It allows users to access and run the web-based system, such as the Web-Based Mental Health Management System (MHMS) for RHU Bulan. The web hosting that the researchers used is Hostinger, which provides hosting services that would post website content throughout the internet, enabling users to gain access.

Table 3: Service Specification

SERVICE	**PROVIDER**
Internet Service Provider	PLDT
Web-hosting	Hostinger

3.2 User's System Specification Requirements

In this section, the researchers outlined the minimum and recommended hardware and software specifications for users' systems. Accurately specifying these requirements was crucial to ensure optimal system performance in the designated environment.

Table 4: User Hardware and Software Requirements

COMPONENT	MINIMUM	RECOMMENDED
Processor	Dual Core	Intel(R) Pentium(R) Silver N600 @ 1.10GHz 1.11GHz
Memory	4GB RAM	8 GB RAM
Hard Disk	120GB	120GB or Higher
Internet Connection	10mbps	20mbps or Higher
Peripherals	Monitor, Keyboard, Mouse	Monitor, Keyboard, Mouse
Operating System	Windows 10	Windows 11
Browser	Microsoft Bing	Microsoft Bing

3.3 Technical Terms

Throughout this study, various terms were used. The goal of this section is to clearly define these words both conceptually and operationally to provide a better understanding of their usage in the study.

Data Management - Data management is a process of collecting, managing, and securing stored data that is owned by a certain organization for efficient utilization. In the proposed system, records containing data are used for evaluating and analyzing the overall patient's mental health condition in Bulan, which is essential for problem-solving, evidence-based analysis, and treatment planning. [16].

Reporting and Analytics - Reporting and analytics served as key tools for presenting and analyzing data in the proposed system. These features were vital for improving the efficiency and quality of mental health care, facilitating monitoring services, supporting evidence-based practices, and driving better outcomes in mental health policies and programs. [17]

UI Design - User interface (UI) design is the process of building an interface for software or web-based systems, basically focusing on designing a user-friendly interface.[18]

Appointment Booking - Appointment booking for scheduled check-ups and assessments is a crucial process in the healthcare industry that enables medical practices and clinics to efficiently manage patients and doctor's scheduled appointments. It involves a scheduling system for booking and appointment management for users, including initial consultations, follow-up visits, and medical procedures [19].

Medicine Inventory - A medicine inventory in a mental health management system is a comprehensive record of medications prescribed and dispensed to patients. This inventory tracks key information such as medication names, dosages, quantities, expiration dates, and stock-in and stock-out information. Effective medicine inventory management is essential for ensuring patient safety, optimizing medication adherence, and minimizing waste. By monitoring stock levels, identifying potential shortages, and streamlining the medication dispensing process, mental health providers can improve overall patient care [].

Dosage - Dosage refers to the prescribed amount of medication. It plays a critical role in ensuring accurate prescriptions, patient safety, effective inventory management, clinical oversight, and compliance with regulations in mental health care. [20].

Milligram - Milligram (mg) is a unit of measurement for medication weight. It's crucial for precise dosing, inventory management, formulary management, and clinical decision-making in mental health systems. [21].

Expiry Date - Expiry date is the final date a medication is safe and effective. It's crucial for patient safety, inventory management, and regulatory compliance in mental health systems. [22].

Quantity - Quantity in a mental health management system's medicine inventory refers to the number of units or dosage forms of a specific medication available for dispensing. Accurate tracking of medication quantities is crucial for ensuring that sufficient supplies are on hand to meet patient needs, preventing stock-outs, and avoiding unnecessary waste. By monitoring quantity levels, healthcare providers can optimize inventory management, reduce costs, and improve patient care [23].

Batch Supply - Batch supply in a mental health management system refers to the process of purchasing and storing medications in specific quantities, often referred to as batches. This approach is commonly used to optimize inventory management and reduce costs. By purchasing medications in bulk, healthcare providers can benefit from potential discounts and minimize the frequency of reordering. However, it's essential to balance the advantages of batch supply with the risk of overstocking, which can lead to medication expiration and financial loss [24].

Batch Number - A batch number is a unique identifier for a specific production run of a medication. It's crucial for tracking a drug's origin, expiration date, and potential recall. This ensures quality control, regulatory compliance, and efficient inventory management. In mental health systems, accurate batch number tracking is essential for patient safety and effective treatment. [25].

Prescription - A prescription in a mental health management system is a written order from a healthcare provider to a pharmacist for a specific medication. It typically includes details such as the patient's name, the medication name, dosage, quantity, and duration of treatment. By tracking prescriptions, mental health providers can ensure accurate medication dispensing, monitor patient adherence, and identify potential drug interactions [26].

Tracking - Tracking of medicine is also a crucial part of monitoring and management of healthcare facilities as these hold the patient's medication intake, essential for identifying the most used medicine. Basically, it organizes the information on medicine supply stock. [27].

Stock In - Stock-in in medicine inventory refers to the process of replenishing depleted medication supplies. This involves various activities such as placing orders, receiving shipments, and storing medications in appropriate locations. Effective stock-in management is crucial for ensuring the timely availability of essential medications, particularly in healthcare settings with fluctuating demand and complex supply chain dynamics. [28].

Stock Out - A stock-out in medicine inventory occurs when a specific medication is unavailable to meet patient demand. This can lead to significant consequences, such as delayed treatments, increased patient dissatisfaction, and potential adverse health outcomes. Stock-outs can be caused by various factors, including supply chain disruptions, unexpected surges in demand, or poor inventory management practices. Effective inventory management strategies are essential to minimize stock-outs and ensure the continuous availability of essential medications. [29].

Archive - An archive in a mental health management system is a secure repository for storing patient records, treatment plans, progress notes, and other relevant documentation. This archive ensures the long-term preservation of sensitive health information, facilitating continuity of care, legal compliance, and research purposes. By maintaining a well-organized and accessible archive, mental health providers can uphold patient confidentiality, support clinical decision-making, and contribute to the advancement of mental health research [30].

Restore - Restoring archived information in a mental health management system involves retrieving previously stored patient records, treatment plans, or other relevant data. This process is often necessary for various reasons, such as legal requirements, clinical research, or patient care continuity. By carefully accessing and utilizing archived information, healthcare providers can gain valuable insights into a patient's history, identify trends, and make informed treatment decisions[31].

Calendar Viewing - Calendar viewing in a mental health management system is a feature that allows users to visualize available appointment slots and schedule appointments with healthcare providers. This tool helps streamline the appointment booking process, reducing wait times and improving patient access to care. By providing a clear overview of provider availability, patients can easily select suitable appointment times that fit their schedules[32].

Online Status - The online status feature in the mental health management system offers real-time visibility into the availability of patients and healthcare personnel. This feature enhances communication and collaboration, facilitating efficient scheduling, messaging, and virtual consultations. By visually indicating users as "online" or "offline" and allowing for customizable presence settings, the system promotes timely interactions. Additionally, users can receive notifications when specific contacts or groups change their online status, further optimizing communication. Integrating online status with messaging and video conferencing tools streamlines scheduling and minimizes wait times. This feature ultimately contributes to a more efficient and effective mental health management system.[33].

Offline Status - The offline status feature in the mental health management system indicates when a user is currently unavailable or not actively using the system. This feature provides transparency and helps manage expectations, particularly for time-sensitive communications.

When a user is offline, the system may display a specific status message or notification, such as "Currently Unavailable" or "Will Respond Later." This helps to avoid misunderstandings and unnecessary interruptions. By effectively utilizing offline status, the mental health management system enhances user experience and promotes a healthy work-life balance for healthcare professionals and patients.[34].

Email Verification Email verification is a security measure implemented in web-based mental health management systems to authenticate user identities and safeguard sensitive health information. This process typically involves sending a verification message to the user's registered email address. Upon successful verification, the user's account is activated, granting them access to the system's features and functionalities. By requiring email verification, these systems mitigate the risk of unauthorized access, protect user privacy, and ensure the integrity of the information stored within the platform. [35]

Notification Alert Notification alert messages within a web-based mental health management system serve as critical communication channels, informing users of pertinent updates, reminders, and action items. These messages can range from simple notifications about new appointment reminders to more complex alerts regarding changes in treatment plans or urgent health concerns. Tailored to individual user preferences and needs, these alerts can be delivered via various modalities, including email, and database notifications. By providing timely and relevant information, these messages empower users to actively engage in their mental health journey. Moreover, they can serve as a valuable tool for healthcare providers, enabling them to maintain effective interaction with their patients and monitor their progress. [36]

4 Design and Methodology

This section describes the procedures, techniques, tools, and documentation used in the study, along with diagrams, figures, and tables that illustrate the methodology.

4.1 Concept

The project aimed to create a web-based mental health management system for monitoring, recording, managing, and assessing patients with mental health problems in RHU Bulan. The system also featured a medicine inventory accessible only to RHU staff, who could modify, and monitor the stocks, dosage, and expiration dates of medicines. It was primarily used by patients, health personnel, and administrators at the Rural Health Unit of Bulan. The project followed Object-Oriented Analysis and Design (OOAD), adopted a RAD-type approach for the development life cycle, and utilized Bottom-Up development strategy. Additionally, it incorporated various technologies such as PHP, Laravel, Javascript, HTML, CSS, MYSQL, and Microsoft Visio. These methods and tools will be further explored in the following sections. The accompanying activity diagram provides a visual representation of the system's flow.

4.1.1 System Architecture Diagram

System architecture diagrams provide a visual illustration of the system's various components and show how they communicate and interact with each other. [37]. These diagrams document a system's structure and architecture. This allows users to have a clear understanding of how the system works and how it can be improved. The following were the System Architecture diagrams meant to show the deployment architecture of the system in both general and specified or cloud view.

System_Architecture_Diagram.drawio.png

Figure 4.1. System Architecture Diagram (General View)

Figure 4.1 shows the general view of the architectural diagram that defines the structure of the system. The users interact with the web server by requesting and receiving through Hypertext Transfer Protocol (HTTP). The Web Server consisted of the various layers of the application that confirmed the Model-View-Controller (MVC) design pattern. The Application Layer manages the flow of the application, implements the logic of the system, and liaises with the data layer to process requests from users and their responses. The Presentation Layer was where the interaction between the users and the system would first take place. The Data Layer handled the domain data and provided persistence and retrieval services for the database. The Database using MySQL was where the data persisted and retrieved. Finally, web services handle interactions with other applications.

The Web Server, Database, and Web Services were hosted in the cloud through Hostinger. In this way, proponents were not limited to the hardware specifications for deploying the system, it would be available on the internet, and with fast and reliable performance, security, and scalability.

4.1.2 Data Flow Diagram

dataflow_lvl_0.drawio.png

Figure 4.2. Data Flow Diagram Level 0 (Context Diagram)

Figure 4.2 presents the level 0 DFD or the context-level data flow of the system. The system was represented as a rectangle located in the middle of the figure with the number zero at the top, indicating the 0th level of the diagram. Users – administrators, health personnel, and patients were also represented as a rectangle and their interaction to and from the system was represented as arrows. As shown in the figure, the system accepts different inputs from the user and returns different outputs following their privileges.

Figure 4.3. was the level 1 DFD. While the context-level data flow showed the whole system as a single process, Level 1 DFD notated each of the main subprocesses that comprised the system. It included external entities (Administrators, Health Personnel, and Patients), the processes they do in the system (RHU Bulan Staff and Employees among others), and the data store where the processes were stored or retrieved data to be processed.

This Level 1 data flow diagram (DFD) illustrates how information moves through a mental healthcare management system, showcasing the interactions between Health Personnel, Administrators, and Patients, along with various processes and data stores.

The system begins with Process 1.0, which manages patient information and connects them with health personnel, storing data in Data Store D1. A parallel *Process 1.0* manages health personnel details in the same data store. *Process 2.0* handles scheduling by linking patients and health personnel with their appointments, which are recorded in Data Store D2 to make scheduling details accessible.

Processes 3.0 and 4.0 manage health personnel and patient appointment schedules, respectively, using Data Stores D3 and D4. These steps ensure smooth scheduling and avoid appointment overlaps, enhancing accessibility for both parties.

Process 5.0 stores mental health records in Data Store D4, allowing health personnel to access and update patient information, which is essential for tracking progress and ensuring consistent care.

Next, Process 6.0 manages prescribed medications, while *Process 7.0 oversees the medicine inventory, with stock details stored in Data Store D7. This setup ensures medications are current and supplies are monitored, supporting effective medication management.

The Administrator has access throughout, overseeing operations and data to maintain security and integrity. This DFD Level 1 offers a clear view of the mental healthcare system's processes, highlighting efficient data flow and coordination among the health personnel, patients, and administrators, which enhances service delivery and data protection.

dataflow_lvl_1.drawio.png

Figure 4.3. Data Flow Diagram Level 1

4.1.3 Use-Case Diagram

According to [38] use case diagram is a visual tool used in software design, specifically within the Unified Modeling Language (UML) framework. It depicts the interactions between users, represented by actors, and the functionalities offered by a system, known as use cases.

Figure 4.4. Use Case Diagram

Figure 4.4 shows the system and its boundary as a rectangle, the functionalities in an oval shape, the actors, and their interaction with the functionalities within the system. The Table Descriptions of the use cases utilized in the diagram can be found in Appendix B: Use Case Table Description. The following actors and their scope of interaction were as follows:

Administrator

The Administrator oversees the broader management of the system. Their responsibilities are vast and include managing both User and Patient Records. In the Manage Users feature, they can add, edit, or archive users (including inactive patients and health personnel), ensuring that only active users can access the system. They also have control over the Manage Patient Records feature, where they can update and generate reports for any patient under their supervision, maintaining a comprehensive and well-documented system. Additionally, the Administrator has a pivotal role in overseeing Appointments, managing the entire booking system, and ensuring that health personnel and patients can easily schedule or modify appointments. Finally, they manage the medicine inventory, ensuring that the stock of medical supplies is up-to-date and that they can generate reports related to the inventory as needed.

Health Personnel

The Health Personnel plays a central role in managing patient care. They have access to critical features such as Manage Patient Records, allowing them to add, show, and update patient information, which is essential for maintaining up-to-date health records. Health Personnel can also generate reports via the Reports feature, which assists them in assessing patient progress and outcomes. Furthermore, they are involved in managing appointments through the Booked Appointment and Manage All Booked Appointment features, ensuring they can oversee and manage their schedules efficiently. Lastly, they also have a hand in the Manage Medicine Inventory, where they can add, show, update, and archive medicine, ensuring that their patients receive the correct medications when needed.

Patients

Patients' role primarily focuses on accessing and interacting with their appointment bookings and medical records. They have limited access to the system but can log in/register to view personal details and appointments. Through the Booked Appointment feature, they can view their upcoming or previous appointments, ensuring they are always aware of their therapy schedule. This streamlined approach for patients ensures a simple yet effective interaction with the system, focusing on their own health journey without complicating involvement in administrative tasks.

4.1.4 Activity Diagram

An activity diagram is a visual representation of the sequential flow of actions within a system, along with the decisions and object manipulations that occur [39]. It depicts the steps involved in a specific process, like placing an online order or logging into a system. Figure 4.6 to Figure 4.10 presented the Activity Diagram or the control flow that occurred in the system from login or registering to the various activities of the different users.

Figure 4.5, this diagram outlines the process flow for a system's login and registration, showcasing how users interact with the system and how it handles different scenarios with a mix of complexity and variability.

The process starts with users being directed to either the Login Page or the Registration Page. If a user chooses to log in, they must enter their Username and Password. Upon submission, the system checks if the credentials are valid. If they are, the user is routed to their respective interface depending on their role—whether they are a Health Personnel, Patient, or Administrator. However, if the

Activity_Flow_Chart.drawio.png

Figure 4.5. Login and Register Activity Diagram

credentials are invalid, the system counts the number of failed attempts. If there have been fewer than three attempts, the user is allowed to retry. After three failed attempts, though, the system enforces a brief lockout, preventing any further login attempts for 1 minute. This ensures security by discouraging repeated login failures, which might signal a malicious attempt to breach the system.

On the registration side, new users navigate to the Registration Page where they are required to provide personal details including First Name, Last Name, Middle Name, Email, Username, and Password, along with a password confirmation to ensure accuracy. Once submitted, the system validates the input. If all information is valid, the user is successfully registered and assigned to their appropriate role in the system. If the data is invalid, an Error Alert Prompt is triggered, asking the user to review and correct their inputs.

Figure 4.6, The Administrator Activity Flow Diagram outlines the key processes that administrators follow to manage appointments, patient records, and medical inventory in a healthcare setting. The diagram is structured into four levels, each adding more detail and complexity to the tasks as the administrator progresses through their day.

In Level 1, the administrator starts by selecting a broad area to focus on: managing appointment bookings, patient records, medicine inventory, or accessing the dashboard. This level serves as a high-level decision point, laying out the primary functions that will guide the administrator's workflow.

When moving into Level 2, the tasks become more specific. For instance, if the administrator selects appointment bookings, they can then choose to manage therapist schedules or patient appointments. Similarly, selecting patient records allows the administrator to access and update vital information. Choosing the medicine inventory option leads to managing medical stock, which is crucial for ensuring the availability of necessary supplies.

By the time the administrator reaches Level 3, they are handling more detailed tasks, such as viewing health personnel appointment schedules, patient appointment histories, or updating patient information. This level represents a critical hands-on stage where the administrator ensures that records are accurate and up-to-date, which is vital for the smooth operation of the healthcare facility. Additionally, administrators manage medicine stocks by viewing and updating inventory data, ensuring that supplies are properly tracked and available when needed.

Activity_Flow_Chart_Admin.drawio.png

Figure 4.6. Administrator User Activity Diagram

At Level 4, the tasks become more intricate, requiring the administrator to handle updates, evaluations, and archives. For appointment management, they can view, update, or archive schedules to reflect real-time changes. In the patient records section, administrators gain access to detailed medical information such as evaluations, diagnoses, prescriptions, and treatment progress. This stage is essential for maintaining the accuracy of patient care records. The diagram also highlights the management of medicine inventory, where the administrator handles tasks such as stock in, stock out, and processing returns. This area adds a degree of burstiness to the workflow, as inventory levels need frequent monitoring and adjustment.

Activity_Flow_Chart_Patient.drawio.png

Figure 4.7. Patients Users Activity Diagram

Figure 4.7, The Patient Activity Flow Diagram illustrates the sequence of events in managing patient data and appointments in a healthcare system. It is divided into six lanes, each representing a distinct area of patient interaction and data management. Lane 1 begins with general actions like booking an appointment or visiting the dashboard. This suggests that this lane deals with initiating patient-related activities, including scheduling and accessing an overview of patient records.

Moving to Lane 2, the focus shifts to patient records, where users can access and review individual patient data. This lane is essential for maintaining and viewing patient history, a critical part of healthcare data management.

In Lane 3, the diagram shows how appointments are handled. It includes steps like viewing the patient appointment schedule and accessing specific appointments. This lane is crucial for ensuring that patient visits are well-organized and can be easily tracked by healthcare staff. Lane 4 further develops this by introducing processes for updating, adding, and archiving patient information. This ensures that the system remains updated, reflecting any new or modified patient data, such as updated contact information or changes to a patient's health status.

Lane 5 introduces decision-making, where the system must account for changes or cancellations in appointments. A diamond-shaped decision node represents these choices, asking if the appointment needs to be canceled or rescheduled. Depending on the answer, the flow either leads back to updating appointment bookings or forward to the dashboard for a new action.

Lastly, Lane 6 deals with updating appointment bookings, finalizing any changes made in previous steps. This lane ensures that all updates or cancellations are properly reflected in the system, maintaining accuracy and consistency in patient scheduling.

Activity_Flow_Chart_HealthPersonnel.drawio.png

Figure 4.8. Health Personnel User Activity Diagram

Figure 4.8, The Health Personnel Activity Flow Diagram illustrates the systematic flow of activities performed by therapists in managing their daily operations. It's structured into six distinct levels, representing various stages of the process. At Level 1, the journey begins with general actions like visiting appointment booking, patient records, medicine inventory, or accessing the dashboard. This stage serves as an entry point where different tasks are triggered based on the needs of the therapist.

Moving to Level 2, tasks are broken down further. For instance, upon selecting the appointment booking option, the health personnel proceeds to visit the appointment section. Similarly, they can access patient records, revealing a structured approach to manage patient data, or they can visit the medicine inventory for stock-related activities. At Level 3, the health personnel delves deeper, viewing patient schedules, updating patient information, or inspecting medical stock records. This level introduces more granular management, showing how information is retrieved and modified. Notably, patient data can be archived, ensuring an organized and comprehensive system.

At Level 4, we witness more critical health personnel duties such as adding evaluations, treatment outcomes, diagnoses, and prescribed medications to the patient record, highlighting the workflow from diagnosis to treatment. This ensures each patient's treatment is continuously tracked, evaluated, and adjusted.

A unique twist occurs in Level 5, introducing decision points like rescheduling or canceling appointments. This stage emphasizes flexibility, where appointments can be archived or rescheduled, reflecting real-world changes in schedules or patient availability. Health Personnel can also update and archive patient records, maintaining the integrity and currency of medical history.

Finally, Level 6 focuses on updating the appointment schedule, effectively closing the loop in the appointment management process.

4.1.5 Entity-Relationship Diagram

According to [40] Entity-Relationship Diagram (ERD) is a graphical representation in database design that depicts the entities (real-world objects) and their relationships within a system. The entity-relationship model (ER model) is a blueprint that describes the structure of a database with the use of a diagram which can then be implemented as the actual database.

The Entity-Relationship Diagram (ERD) presented here provides a comprehensive overview of a mental health management system. It illustrates the interconnectedness of various entities, including patients, prescriptions, health personnel, admins, inventory, and archived records. The connecting lines between entities highlight how these pieces of data relate and interact, forming a comprehensive structure for managing patient care, appointments, and administrative tasks.

At the heart of the system lies the patient entity, encompassing details like personal information, medical history, and appointment schedules. Prescriptions are closely tied to patients, detailing medications, dosages, and quantities. Health Personnel, the system's caretakers, are responsible for patient management and treatment plans. Their information, including specialization and workload, is meticulously recorded. Admins oversee the system's operations, managing user accounts and system settings. The inventory entity tracks medication supplies, ensuring their availability. Lastly, the archive entity maintains historical records of patient interactions and system events.

This ERD highlights the system's focus on patient care, with a strong emphasis on tracking medical records, managing appointments, and coordinating therapy sessions. The integration of inventory management ensures a smooth supply chain for medications. Additionally, the archive provides a valuable resource for data analysis and future reference. The system's design appears to prioritize data security and privacy, with robust authentication and authorization mechanisms for admins and therapists.

Overall, this ERD represents a robust and well-structured database for a mental health management system, capable of supporting efficient operations and delivering high-quality patient care.

ERD_REVISE_VER.png

Figure 4.9. Entity Relationship Diagram

4.2 Analysis and Design

This study adopted a powerful software development methodology known as Object-Oriented Analysis and Design (OOAD). Unlike traditional linear approaches, OOAD utilizes an iterative and incremental process. According to [41], the use of OOAD means the system is built and analyzed in stages, with each step adding new functionalities and refining the design based on feedback.

Object-Oriented Analysis and Design (OOAD) is a software engineering approach that uses objects to model and design intricate systems. By breaking down complex problems into smaller, manageable units called objects, OOAD helps create modular, scalable, and maintainable software solutions. It helps create systems that are easier to understand, maintain, and extend by organizing functionality into reusable and interconnected components [42].

4.2.1 Requirement Analysis

Requirements analysis is a crucial process in software development that involves gathering, analyzing, and documenting the specific needs and expectations of a new product or system. By clearly understanding these requirements, development teams can ensure that the final product meets the needs of its users and stakeholders. [43].

Following the Object-Oriented Approach in Analysis and Design, the following tables were the functional and non-functional requirements that the system complied with. Diagrams were also presented to further visualize the functional requirements of the system.

Functional Requirementse

According to [44], Functional requirements are defined as the specific actions or functionalities a system must perform to meet its intended purpose. Table 5 presents the functional requirements of the system, specifically its task description and reference. The task requirements reference is directly linked to the objectives of this study.

Non-Functional Requirementse

TASK REQUIREMENTS	TASK REFERENCES
The mental health management system should provide robust user management functionalities. It should allow for the creation, modification, and archiving of user accounts, login status well as the assignment of appropriate user-type and permissions. The system should enforce strong password policies and implement multi-factor authentication to enhance security. Additionally, it should track user activity, log access attempts, and generate audit trails for compliance and security purposes.	User Management
The mental health management system should manage patient records, including registration, login, medication tracking, treatment plans, view patient information, and appointment booking. It should provide a secure platform for patient-provider communication.	Patients Record Management
The mental health management system should allow for efficient appointment booking. It should enable users to schedule appointments with providers, view available appointment slots, and receive reminders. The system should integrate with calendars to avoid scheduling conflicts and send automated appointment notifications. Additionally, it should allow for easy cancellation and rescheduling of appointments.	Appointment Booking
The system should effectively manage the inventory of medications. It should track medication stock levels ,quantity, batch supplies and expiration dates. Additionally, it should generate alerts for low stock levels and expiring medications. The system should facilitate the tracking of medication distribution to patients, including dosage, frequency, and refill information.	Medicine Inventory Management
The system should track the movement of medications within the inventory. It should record when medications are received, dispensed to patients, or returned to the inventory. This tracking helps to maintain accurate stock levels, identify any discrepancies, and ensure proper medication management.	Tracking of Medicine

Table 5: Functional Requirements

According to [45], In contrast to functional requirements, non-functional requirements (NFRs) describe the qualities or characteristics of a system, rather than what specific actions it should perform. Table 6 resented the non-functional requirements of the systems. The description of each requirement and its task reference were also presented. The references were adapted from the International Organization for Standardization Software Quality Standards, specifically in ISO/IEC 9126-1.

TASK REQUIREMENTS	TASK REFERENCES
The system should be able to perform all the functional requirements and provide the appropriate results.	Functionality
The system should be accessible to it's different users, have interactive and aesthetic user interface appropriate for it's users, and be easily recognize by the users for the use and appropriateness of the system.	Usability
The system should function across different devices (desktops, laptops, tablets etc.) and operating systems regardless of the specific location.	Portability

Table 6: Non-Functional Requirements

4.3 Development Model

As shown in Figure 4.12, the Software Development Life Cycle (SDLC) model used in this study is the Rapid Application Development (RAD). RAD is known for its incremental and time-constrained software development approach, enabling the rapid creation of applications. Planning requirements, system design, and implementation are needed for this methodology to ensure that the application is developed quickly and accurately to sustain the user's needs. [46].

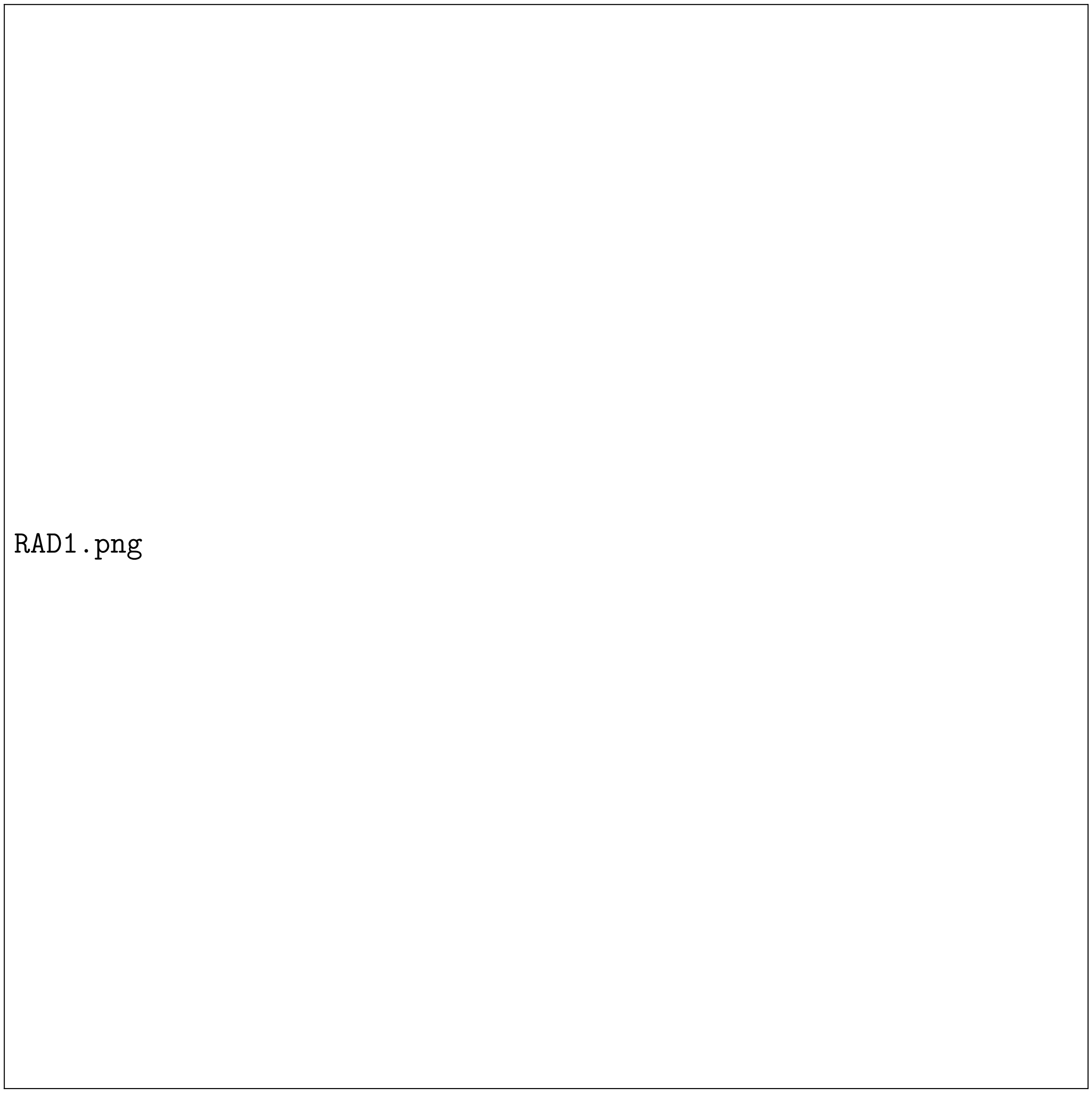

Figure 4.10. Rapid Application Development Diagram (RAD)

The following are the phases of Rapid Application Methodology:

1. **Requirements Planning -** During the requirements planning phase, the users and analysts meet to identify the application or system's objectives and the information requirements that derive from those objectives. This phase requires active engagement from all parties; it is not merely approving a plan or document. The target client was orientated throughout this phase, and data was gathered to build solutions to the problem. The developers spoke with the 2nd Provincial Mobile Force's Supply Accountable Officer about the present problems.
2. **User Design -** This is where the RAD methodology emphasizes and distinguishes itself from other project management methodologies. During this phase, developers work with clients to ensure that their demands are met at every stage of the design process. Users can evaluate each product prototype at each stage to ensure it meets their needs, similar to how customized software development works. The researchers choose to create an application platform system. Because the application system is available offline or without an internet connection, data can be transferred in a short amount of time.
3. **Construction -** It converts the prototype developed during the design phase into a working model during this step. Researchers built the final working model in less time than they would have in a traditional project, and improvements were made during the iterative design phase. The phase is divided into smaller sections: preparing for quick construction, developing programs and applications, coding, and system, unit, and integration testing. The System Engineer and System Analyst interact during this stage to ensure that everything is functioning smoothly and that the final product satisfies the client's expectations and objectives.
4. **Cutover -** This is the point at which the completed work is ready for distribution. User training, data conversion, testing, and switchover to the new system are all covered. All final improvements are made as the developers and clients continue to look for flaws in the system.

The developers of the proposed system follow a process that includes planning and requirements, analysis and user design (prototype, test, and refine), construction, and cutover.

This strategy greatly aids the project's proponents in being more flexible to changes to improve or optimize the design until they are satisfied with the expected result. Finding functional or design flaws early on in the development process enables corrective action to be taken on a limited budget and time frame.

Early in the development stage, the RAD method produces a workable model for the system. The RAD process can reduce project risk and lengthen development periods since they can find and fix problems early in the process.

4.4 Development Approach

The system design strategy used in this study was the bottom-up development approach. According to [47], the bottom-up approach involves breaking down a system into modular components when designing it. The process begins by identifying modules at the lowest level and grouping them based on their functions to create higher-level modules. This continues until all components and subsystems are integrated into a complete system. As the design progresses to higher levels, the level of abstraction increases significantly.

This study utilized the bottom-up development approach due to its compatibility with the Object-Oriented approach that is also applied here. Furthermore, as the study follows a RAD process based on the RAD Model in SDLC, the researchers determined that the bottom-up approach was the most suitable for the project [48].

4.5 Software Development Tools

The following software development tools and applications were used to develop the system:

- **Front-End Development**
  - **HTML 5 -** HTML 5 is the latest standard of Hypertext Markup Language, the code that describes the structure and presentation of the web pages [49]. Since the current project

was implemented with the use of the web and to experience the new features of the latest version, HTML5 was used in this project.

- **CSS 3** - Cascading Style Sheets (CSS 3) is a style sheet language used for specifying the presentation and styling of a document. It is also signed to enable the separation of content and presentation, including layout, colors, and fonts [50]. Together with HTML, along with other core technologies, is essential for constructing web pages and enhancing their visual appeal. Therefore, it was crucial to incorporate this markup language into the project.
- **Tailwind** - Tailwind CSS is a utility-first CSS framework designed to enable users to create applications faster and easier[51]. The proponent used Tailwind CSS because it can be a beneficial choice for building web-based management systems due to its focus on speed and customization. Tailwind's utility-first approach with pre-built classes lets developers quickly style components without writing a lot of CSS.
- **Bootstrap version 5.0** - Bootstrap v5.0 is a popular open-source CSS framework that provides pre-built, responsive CSS components and JavaScript plugins to accelerate web development. It offers a wide range of customizable components, such as buttons, forms, navigation bars, and more, to create visually appealing and functional websites [52]. The proponents used Bootstrap for a web-based mental health management system due to its numerous advantages. Bootstrap accelerates development with pre-built components, ensures responsive design for various devices, offers cross-browser compatibility, benefits from a large community and support, and allows for customization. By utilizing Bootstrap v5.0, the proponent can efficiently create a robust, user-friendly, and visually appealing mental health management system.
- **JavaScript** - JavaScript, is a programming language and core technology of the Web, alongside HTML and CSS [53]. The proponents used this tool because of its capability of offering faster user experiences, user interface interactivity, good responsive web design, ease of learning, and popularity.

- **Back-End Development**

  - **MYSQL** - MySQL is a widely used relational database management system (RDBMS). MySQL is free, open-source, and ideal for both small and large applications [54]. The proponents chose MySQL because it is a popular choice for building web-based management systems because it ticks many boxes for developers, it is also open-source and free to use keeps costs down, while a large community ensures easy access to help.

- **PHP** - A popular general-purpose scripting language that is especially suited to web development. Fast, flexible, and pragmatic, PHP powers everything from your blog to the most popular websites in the world [55]. The researchers chose PHP because it is a great choice for web management systems. It's easy to learn, has tons of resources, and works great with databases, all essential for building the system of the study efficiently.

- **LARAVEL** - Laravel is a free and open-source PHP-based web framework for building high-end web applications [56]. The researchers chose Laravel as the PHP framework because it has a strong contender for building the back-end of the project which is a web-based management system due to its emphasis on developer experience and security. Its clean syntax and features like Blade templating make coding faster and more readable.

- **Xampp** - XAMPP is a popular open-source software package that simplifies the process of setting up a local web development environment on your computer. It bundles together essential tools like Apache HTTP Server, MySQL database, PHP scripting language, and Perl programming language. By installing XAMPP, you can create and test dynamic websites and web applications locally without the need for a live web server. This is invaluable for web developers as it allows them to work offline, experiment with code changes, and debug issues efficiently before deploying their projects to a live server. XAMPP's user-friendly interface and straightforward installation process make it accessible to both beginners and experienced developers, making it a widely used choice for web development.[57].

- **Apache HTTP Server** - Apache HTTP Server is a powerful and versatile open-source web server that plays a crucial role in serving web content on the internet. It's renowned for its flexibility, reliability, and security, making it a popular choice for both small-scale websites and large-scale web applications [58]. Apache's modular design allows for customization and extension, enabling it to handle a wide range of web traffic and dynamic content. It supports various programming languages like PHP, Python, and Perl, and can be configured to work with different database systems. By using Apache, web developers can create robust and efficient websites that can deliver content to millions of users worldwide.

- **Text Editor and IDE**
  - **Visual Studio -** Visual Studio Code, also commonly referred to as VS Code, is a source-code editor developed by Microsoft for Windows, Linux, macOS, and web browsers. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded version control with Git [59].
- **Version Control System**
  - **Git -** Git is a distributed version control system that tracks changes in any set of computer files, usually used for coordinating work among programmers who are collaboratively developing source code during software development [60].

4.6 Schedule and Timeline

The project's timeline, as shown in Figure 4.13, are composed of two academic semesters – the 2nd semester of the Academic Year 2023-2024 and the 1st semester of the Academic Year 2024-2025. The project started from the idealization of the title on the 2nd week of February and ended in December.

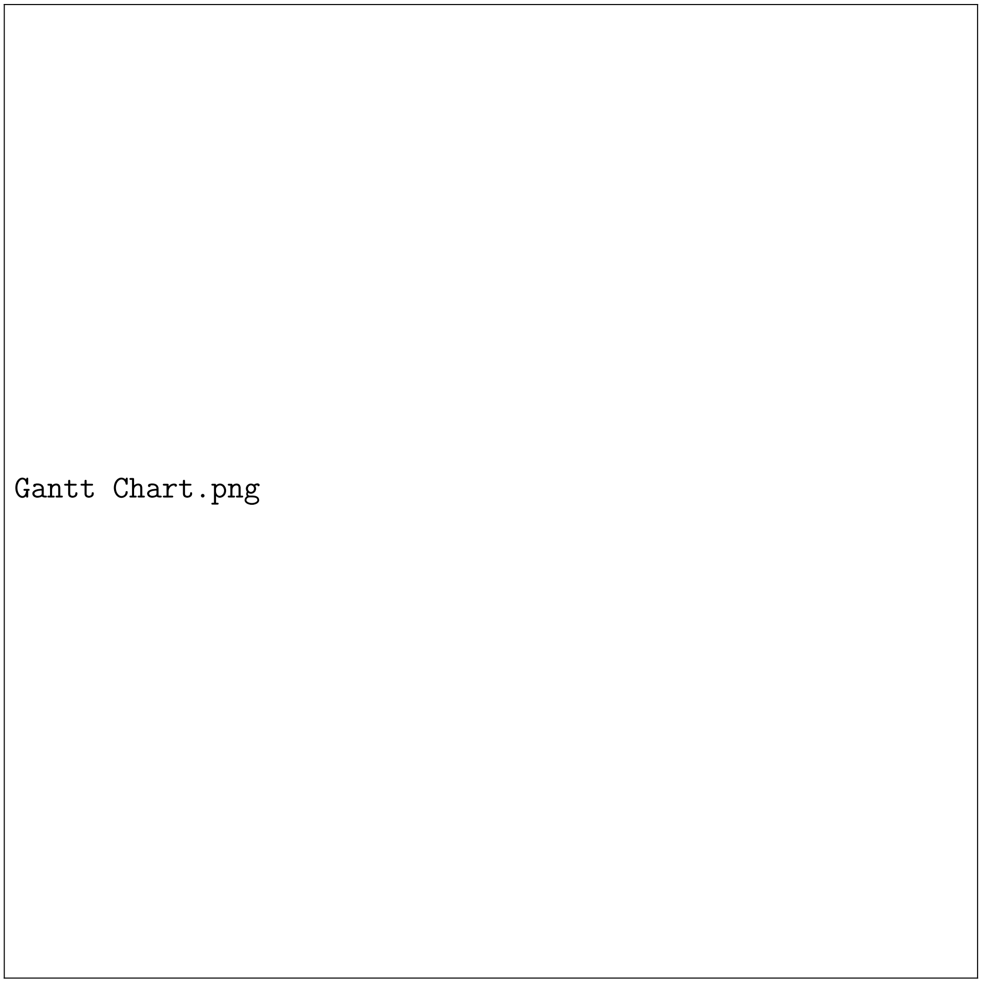

Figure 4.12. Gantt Chart

4.7 Responsibilities

The proponents of this study consisted of four BSIT students and an adviser. Below are their names, roles, and responsibilities. While specific roles were assigned to each member, all were also responsible for supervising the various modules of the system.

- **Homer G. Fuentes - Capstone Adviser** - Adviser plays a crucial role in the Project by providing expert guidance and strategic insights. Adviser help identify organizational needs, analyze existing processes, and recommend improvements aligned with best practices. Adviser facilitate the development of clear objectives, performance metrics, and effective communication channels. Also, support the implementation of the system, monitor its progress, and ensure ongoing compliance with relevant standards and regulations. And lastly, adviser's contribution is vital in creating a robust and efficient management system that drives organizational success.

- **Kathleen Joy J. Jetajobe - Developer/Project Leader** - The Developer is responsible for developing the Front-end by implementing visual elements that users see and interact with. They designed the system's interfaces and other visual aspects of the project including graphical elements in the manuscript. Also, developing the Back-end by handling the server-side development of the system. They are in charge of making sure that the features and functionalities of the system are aligned to the satisfaction of the client. Kathleen Joy Jetajobe, the project leader is the project team's management figure. The Project Leader is in charge of directing team members towards the project objectives. Also, in charge of driving and controlling the colleagues, and should act as a glue between the group. Leader also notify the participation of the colleagues because the leader should be responsible for providing the team's effort with significance and consistency.
- **Catherine S. Granado - Technical Writer** - A Technical Writer plays a pivotal role in the project by creating clear and concise documentation. Technical Writer translate complex technical information into easily understandable language, ensuring that all stakeholders can comprehend and utilize the system effectively. And also develop user manual, training materials, and other essential documentation that guides users through the system's functionalities and processes. By crafting well-structured and informative documents, they contribute to the system's overall efficiency and user adoption.+ The Technical Writer is responsible for preparing, reviewing, revising, and maintaining technical papers and the documentation of this project. Catherine was also responsible for the decisions required to create the features needed for users, specifically, the adviser and panelists.
- **Melody A. Escobedo - System Analysts** - The System Analysts is in charge of formulating and defining the project's scope and objectives, conducting complex research and analysis to define solutions, supporting design and development, and leading system integration testing, user acceptance testing, and implementation phases.
- **Gizelle De los Santos - Quality Assurance** - The Quality Assurance is the gurdian of quality in management systems development. QA sets the stage by defining standards and goals, then meticulously reviews documentation to ensure accuracy and alignment. through

audits and inspections, QA rigorously assesses compliance and identifies areas for improvement. By monitoring performance and fostering continuous improvement, QA ensures the system's effectiveness and customer satisfaction.

4.8 Budget and Cost Management

This project needed a budget for acquiring materials and services, it needed to create the documentation and the whole system. The approximate budget and cost allocation for this project and its operational cost when deployed were indicated below:

Proponents' Budget and Costs Management

The materials in Table 7 were needed for printing the documentation of the study. Printed copies of the manuscript were needed for the proposal, final defense, and the safekeeping of the deliverables. The approximate total amount of the materials, supplies and other expenses was Php 15, 656.00.

ITEM	QUANTITY	PRICE	AMOUNT
Bond Paper	4 Ream	Php 210.00	Php 840.00
Print	4 times	Php 380.00	Php 380.00
White Folder	4 pcs.	Php 8.00	Php 32.00
Internet Load	17 times	Php 102.00	Php 1,804.00
Snacks for Panel	2 times	Php 500.00	Php 1,000.00
Payments for Panels (Overall)	2 times	Php 4,800.00	Php 9, 600.00
Transportation	(overall)	Php 2,000	Php 2,000
		TOTAL AMOUNT:	**Php 15, 656**

Table 7: Materials and Supplies Budget

In Table 8, Internet Service was needed for the learning, searching for information, hosting, and communicating with the proponents and their adviser, professor, and panelists. The researchers will avail of web hosting monthly online. The amount needed for the services and hosting expenses is Php 6,228.00

SERVICE	MONTHLY FEE	AMOUNT
Internet Service (PLDTHOME)	Php 600.00	Php 4,800.00
Web and Cloud Hosting (Hostinger)	Php 476.00	Php 1, 428.00
	TOTAL AMOUNT:	**Php 6, 228.00**

Table 8: Service and Hosting Budget

SERVICE	MONTHLY & YEARLY FEE	AMOUNT
Web Hosting (Hostinger)	Php 476.00/per month Php 5, 712.00/per year	Php 476.00 Php 5, 712.00
	TOTAL AMOUNT:	**Php 6, 188.00**

Table 9: Service Operation Costs

Overall, the approximate total expenses of this project were Php 21, 884.00. However, it did not include the personal expenses of the proponents like food, transportation, and rent. The source of the fund was from the proponents' contribution.

Systems' Operation Costs in Deployment

Table 9 presents the minimum budget of the operational cost of the system when deployed using the following service: Hostinger for hosting the system in the cloud.

hostinger.png

Figure 4.13. Hostinger

Hostinger costs Php 467.00 a month and Php 5, 712.00 a year for their Basic Plan. This is the premium offer of the site as of the writing of this study.

Hostinger's Premium Plan, while offering a foundational suite of features, presents limitations for sophisticated web applications or eCommerce ventures. Its provision of 100GB SSD storage, a complimentary domain, and free migration services, coupled with 24/7 support and a three-month introductory period, constitutes a compelling value proposition for basic websites. However, its absence of advanced functionalities such as a CDN, dedicated IP, WooCommerce integration, and AI-driven tools renders it less suitable for complex or commerce-oriented applications. For organizations requiring such specialized capabilities, a more comprehensive plan or alternative provider may be necessary.

4.9 Verification and Validation

Verification and validation were conducted to ensure that the developed system functioned correctly and met the client's requirements. A combination of quantitative and qualitative methods was used to assess the system's performance.

Verification

Following a Rapid Application Development (RAD) process, the system underwent continuous verification and client checks throughout development. This RAD approach involved frequent walk-through, inspections, and reviews of the design and system processes. To ensure that the system is still aligned with the specification requirements, the team would ensure to get feedback on the completed modules from the client. During the review phases, the project adviser and the panelists would review the representation of the target clients and they would verify if the overall features of the system meet the actual needs and proposed requirements. This ongoing verification within the RAD process helped guarantee the final application would effectively serve the client's needs.

Validation

To check the actual system's effectiveness and if it meets the requirements and expectations of the client, the system undergoes validation. Validation is done by letting the client use the system with actual patients to assess the performance of the system.

The system underwent validation where the actual system was checked if it met the requirements and expectations of the users. To validate the system's effectiveness in a mental health management context for RHU in Bulan, Sorsogon, real-world testing can involve healthcare professionals using the system with actual patients to assess its ability to improve patient outcomes.

The 5-point Likert Scale is employed in this system to serve as a metric for the attributes inside the system's validation that is shown in Table 10. The table would display the overall rating with its verbal interpretation and mean range.

Rating	Mean Range	Verbal Interpretation
5	4.21-5.00	Strongly Agree
4	3.31-4.20	Agree
3	2.61-3.30	Neither Agree nor Disagree
2	1.81-2.60	Disagree
1	1.00 - 1.80	Strongly Disagree

Table 10: Likert Scale for System Validation

Table 11 to Table 18 displayed the questions used in validating the system which is adapted from ISO/IEC 9126-1 software quality standard model. The tables presented the key characteristics (Functional Suitability, Efficiency, etc.), sub-characteristics (Functional Completeness, Correctness, etc.), and the indicators or descriptions of the sub-characteristics.

SUB-CHRACTERISTICS	DESCRIPTION
Functional Completeness	It is comprehensive, addressing all mental health aspects. It ensures accessibility, confidentiality, timeliness, effectiveness, equity, and integration of mental health support within the organization's overall health and safety management system.
Functional Correctness	It is the accuracy and reliability of the system's processes and procedures. It ensures that the MHMS delivers the intended outcomes and aligns with evidence-based practices. This includes accurate diagnosis, appropriate treatment plans, effect effective monitoring, timely evaluation. By prioritizing functional correctness, organization can enhance the quality of mental health services and improve patient well-being.
Functional Appropriateness	It ensures the system's suitability to the system's specific needs, considering factors like size, industry, workforce demographics, and culture. This tailored approach enhances the effectiveness of mental health support.

Table 11: Functional Characteristics

SUB-CHRACTERISTICS	DESCRIPTION
Learnability	It is the ease with which users can learn to interact effectively with the system. This includes factors like intuitive design, clear instructions, and consistent user interface elements. It enables users to quickly navigate the system, access necessary information, and utilize available resources without significant training or support.
Efficiency	It is the speed and accuracy with which users can complete tasks within the system. This includes factors like fast response times, minimal clicks, and efficient navigation. It minimizes user effort and maximizes productivity, allowing users to quickly access and utilize the system's features.
Understandability	Refers to the clarity and comprehensibility of the system's information and interface. This includes factors like clear and concise language, logical information hierarchy, and consistent visual design. It minimizes user confusion and frustration, ensuring that users can easily interpret and interact with the system.
Attractiveness	It refers to the aesthetic appeal of the system's interface. This includes factors like pleasing color schemes, visually appealing layout, and clear typography. It creates a positive user experience, making the system more engaging and enjoyable to use.
Accessibility	The system was designed to be usable by people with diverse abilities. This includes features like screen reader compatibility, readable font sizes, clear and consistent visual design, etc. By prioritizing accessibility, it can ensure that it is inclusive and usable by a wide range of individuals.

Table 12: Usability Characteristics

4.10 Testing

Testing is done to check if the system is working correctly without bugs and errors. It also accepts inputs and overall processes inside the system. This is terminated to examine if the system follows the functional requirements that are being proposed.

The project employed a black box testing strategy, specifically cause-effect analysis, to validate the system's functional requirements. Initial testing was

SUB-CHARACTERISTICS	DESCRIPTION
Adaptability	It ensures the system's ability to adjust to organizational and user needs. This includes integration with other systems, adaptation to new technologies, and modification of features and functionalities. It is longevity and relevance, supporting the organization's mental health initiatives.
Installability	It refers to the ease with which the system can be installed and deployed in different environments. This includes factors such as compatibility with various operating systems, browsers, and hardware configurations. It minimizes setup time and effort, ensuring a smooth and efficient deployment process.
Replaceability	It refers to the ability to replace components of the system with equivalent or upgraded components without compromising the system's functionality or performance. This includes the ability to replace hardware, software, or specific modules within the system. It ensures flexibility and adaptability, allowing for easy maintenance, upgrades, and future-proofing of the system.

Table 13: Portability Characteristics

conducted manually by the development team. Subsequently, during the alpha testing phase, both the advisor and panelists participated in the testing process, utilizing a combination of black-box and white-box techniques to ensure comprehensive evaluation.

Black Box testing examines the function and ability to interpret inputs accurately. It covers positive and negative frameworks where the inputs in the system were tested consistently in the latter. Through the system's ability, the black box can handle valid and invalid inputs through its test outcomes. In the process of executing cause and effect testing it also involved the determining of the intended outcome to produce frameworks that can cause the effects to happen.

PRE-CONDITION			
Test Case/Scenario or Area Name	**Expected Result**	**Actual Result Code**	**Status**
Interoperability			
POST-CONDITION			

Table 14: Test Case Template

To describe the system tests, Table 4.15 was utilized. Preconditions are the conditions or scenarios that need to be met for the test case to be executed. The executed procedures, scenarios, or data used to test the system are referred to as the test case or scenario. The terms ”expected result” and ”actual result” relate to the anticipated and actual results of the test execution, respectively. The status might be ”Fail” if the test cases failed the testing, or ”Pass” if the scenario or case passed the testing or the predicted result and the actual result were the same.

5 Results and Discussions

This chapter aimed to presents a detailed analysis of the proposed system performance and user impact. It delves into the system's efficacy in addressing mental health issues, user satisfaction, and overall system impact. Key findings and insights are supported by data analysis and statistical methods. This chapter also identifies limitations and proposes future enhancements to further optimize the system's contribution to mental health support.

5.1 Development and Testing

During the development phase, the developers and researchers ensures that all system goals and client expectation are met and guided by the (RAD) developmental model that has Requirements Planning, User Design , Constructions, Cut Over and the testing to achieved the objective of the study.

Requirements Planning

During the Requirements Planning Phase of the RAD methodology, the researchers meticulously defined the scope and objectives of the Mindcare system. Through in-depth user needs analysis, including interviews with RHU Bulan staff and potential users, they identified the specific mental health services and features required. Key functionalities such as user registration, monitoring, tracking, recording, appointment booking, medicine inventory management, report generation, and resource access were outlined. Additionally, non-functional requirements, including functionality, usability, and portability, were specified. A detailed project plan was developed, encompassing timelines, resource allocation, and risk management strategies. By effectively executing these activities, the researchers laid a strong foundation for the subsequent development phases, ensuring that the Mindcare system would meet the specific needs of RHU Bulan and its users.

User Design

The second phase delved into the detailed design of the proposed system, outlining its overall functionality and operational mechanisms. Drawing upon the requirements identified in the preceding phase, the system and software designs were meticulously developed. To ensure quality and functionality, multiple prototypes were implemented and rigorously reviewed.

Constructions

During the third phase, the development team constructed the proposed system, adhering to the Rapid Application Development (RAD) model. This iterative approach prioritized rapid prototyping and quick feedback cycles over extended development and testing periods. By employing RAD, the team was able to produce multiple iterations and updates to the software efficiently, without requiring a complete restart of the development process. Throughout the development process, the team actively sought feedback and comments from stakeholders, incorporating suggestions for improvements and modifications. This iterative refinement process ensured that all identified bugs were resolved and the system was continuously enhanced to meet the evolving needs of the users.

Cut Over

The system's user interface underwent rigorous testing to ensure its usability and functionality. Once approved by the development team, the system was deemed ready for implementation, incorporating all necessary components to address potential factors affecting its performance. Subsequently, the system was delivered and presented to the respondents, leading to the successful resolution of the identified problem.

Prototype Checking

- 1st Iteration

  During the initial iteration of system development, the proponents created a prototype that achieved 33 percent of the planned functionalities. This prototype included essential features such as user login for three distinct user roles and a comprehensive dashboard that integrated Patient Record Management, Health Personnel Record Management, User Management, and Report Generator functionalities.

- 2nd Iteration

  The second iteration of the prototype achieved 66 percent of the planned functionalities. Key enhancements included the addition of user registration capabilities, allowing both establishments and individuals to create accounts under administrative oversight. Building upon the first iteration's foundation of Patient Record Management, Health Personnel Record Management, User Management, and Report Generation, the second iteration successfully implemented features related to the second and third objectives of the study: Appointment Booking, Calendar Viewing, All Scheduled Appointment Table List, and Medicine Inventory.

- 3rd Iteration

The final iteration of the prototype achieved 100 percent completion, fully implementing all functionalities outlined in the study's objectives. This iteration incorporated the fourth objective, evaluating system performance using the ISO/IEC 9126-1 standard, which assessed the system's functionality, usability, and portability. By adhering to this standard, the system was optimized to meet client expectations and needs. Minor faults identified during testing were promptly addressed and resolved.

5.2 Description of the Prototype

This section discusses the different components that form the system. Figures that illustrate the design of the web pages per component, their functionality, and how the components communicate with each other are also discussed in this chapter.

The features of the system explaining the needed data when using the system and functionalities to support the operation of the system.

log in1.png

Figure 5.1. Log in Page

In figure 5.1, present the login form page of the system that facilitates user access to the admin's personalized mental health resources and services. To utilize the system, users must provide their unique username and password. Additionally, the platform offers a convenient "Remember Me" feature, allowing users to remain logged in for future sessions. For users who have forgotten their password, a "Forgot Password?" link is provided to initiate the password recovery process. The system also includes a "Create an Account" option for new users to establish their accounts and gain access to the platform's services.

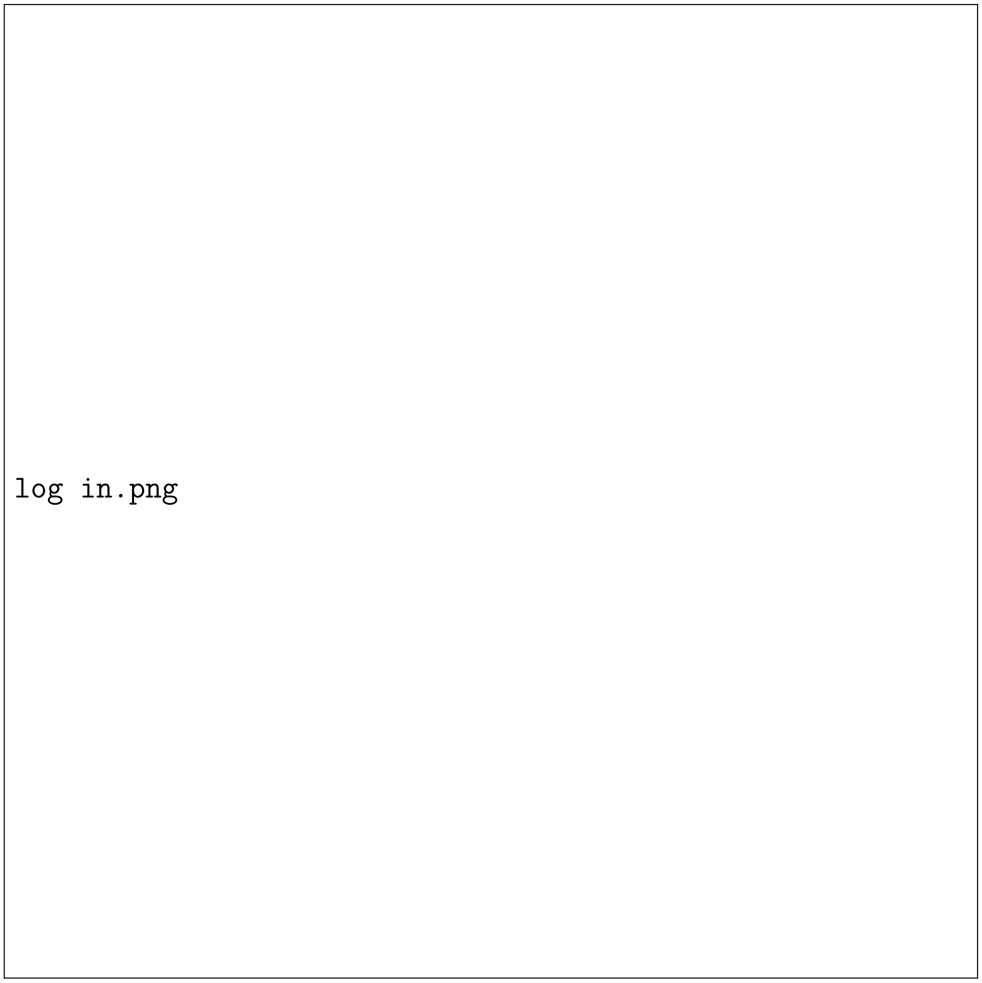

Figure 5.2. User Registration Page

Figure 5.2, shows a password reset feature integrated into a Rural Health Unit Management System. This feature enables users to recover user's forgotten passwords and regain access to their accounts. To initiate the process, users must input the user's registered email address into the designated field and click the "Send Password Reset Link" button. The system subsequently sends a password reset link to the provided email address. Upon receiving the email, users can click the link to access a password reset page where they can input and confirm a new password. The system then validates the new password and updates the user's account with the new credentials. This password reset feature ensures user convenience and system security by allowing users to recover user's accounts in case of forgotten passwords.

password reset email.png

Figure 5.3. Reset Password

Figure 5.3, shows the user registration form for the Mindcare mental health platform. This form enables new users to create an account and access the platform's services. To register, users must provide essential information, including a unique username, their full name, a valid email address, and a strong password. The system requires users to confirm user's password by entering it twice to ensure accuracy. Once all the necessary fields are filled out, users can click the "Register Account" button to submit their information. Upon successful registration, the user will be granted access to the user's personalized account and the platform’s various mental health resources.

```
Admin Dashboard Page 1.png
```

Figure 5.4. Administrator Dashboard Page 1

Figure 5.4, shows the system’s administration dashboard for a healthcare facility. Where admin can track patients, health personnel, and medicine inventory. Also, the admin can manage the key features including patient management, appointment scheduling, inventory management, and generating reports.

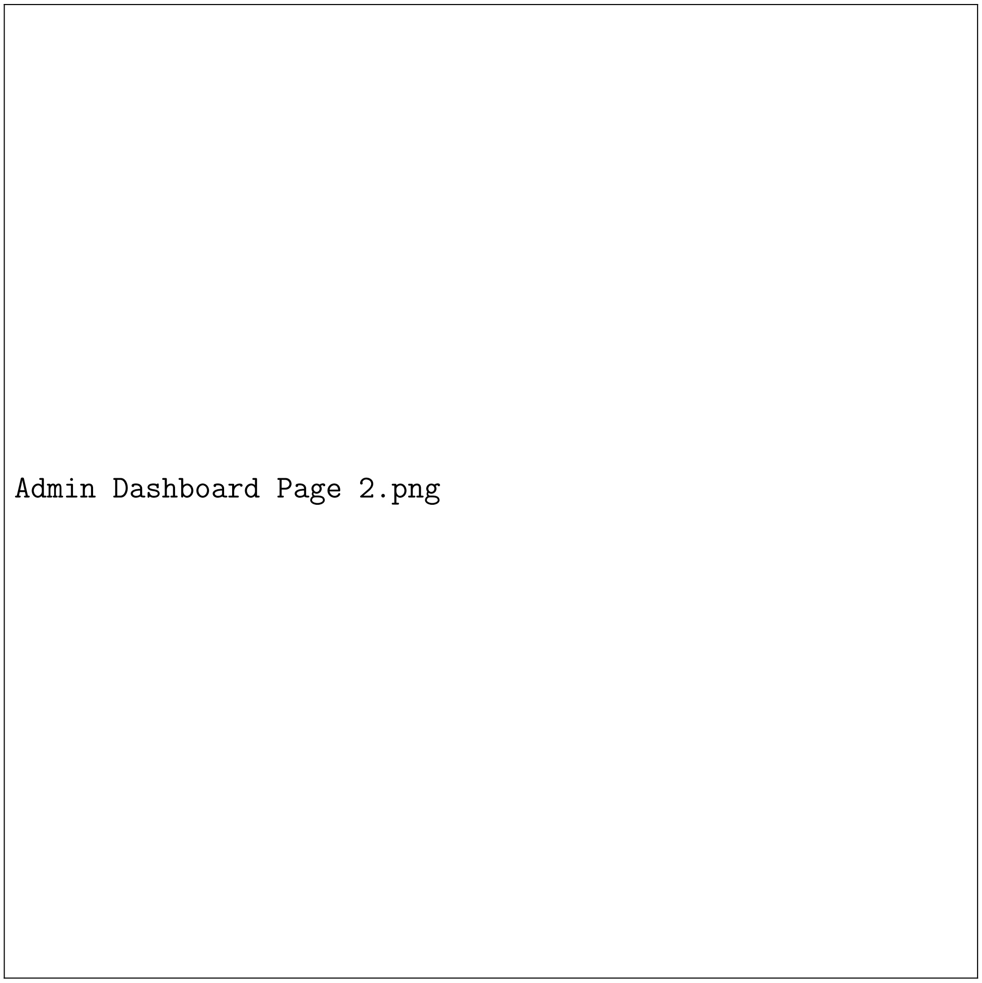

Figure 5.5. Administrator Dashboard Page 2

Figure 5.5, shows the system's admin monitoring dashboard. It is where the admin can track medication distribution, patient assessment status, and patient demographics. Also, key features include data entry, reporting, and patient notification alert systems. It helps the admin to monitor the patient's progress and allocate resources effectively.

Admin Dashboard Page 3.png

Figure 5.6. Administrator Dashboard Page 3

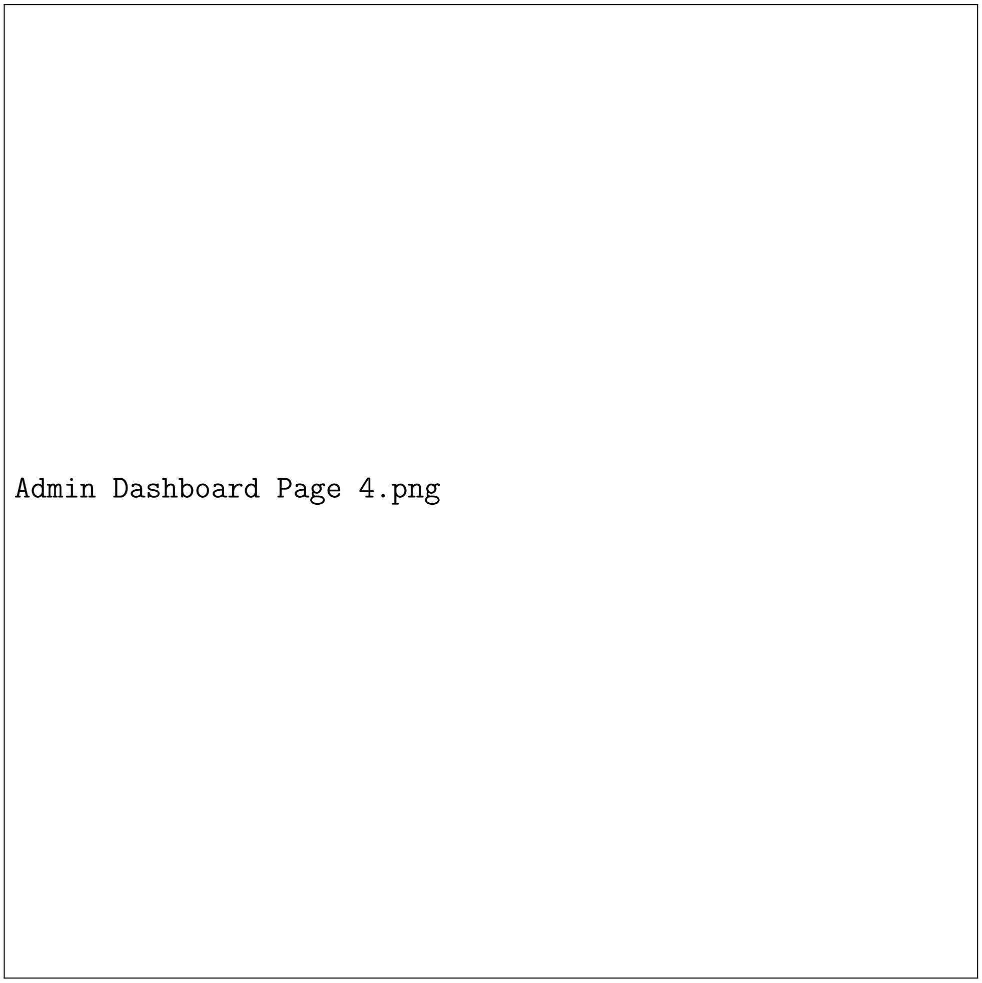

Figure 5.7. Administrator Dashboard Page 4

Figure 5.6 and 5.7, show the system's admins monitoring dashboard where the patient-approved and pending appointment scheduling system is displayed. It is where admin can track pending and approved appointments, including patient information and appointment details. Also manages key features including appointment scheduling, confirmation, reminders, and potential integration with EHR systems. It helps streamline the appointment process and improve patient communication.

User Management.png

Figure 5.8. Amin-User Management

This figure 5.8 shows the user management feature within the Mindcare mental health platform. This feature allows administrators to oversee and manage user accounts. Key functionalities include displaying user information, categorizing users into roles, indicating user status, and providing actions to edit or archive user accounts. This feature ensures the platform's security and integrity by managing user access and monitoring user activity.

Figure 5.9. Admin-User Management (Edit User)

Figure 5.9, showcases a user editing feature within the Mindcare mental health platform. This feature empowers administrators to modify existing user information, guaranteeing accurate and up-to-date records. By accessing a user's profile and making necessary changes to fields like name, username, role, and email address, administrators can ensure the platform's smooth operation and effective user management.

User Management (Archive User).png

Figure 5.10. Admin-User Management (Archive)

This figure 5.10, showcases an archived user feature within the Mindcare mental health platform. This feature enables administrators to manage and restore previously archived user accounts. By displaying a list of archived users and providing a "Restore" button, administrators can maintain a record of past users and reactivate them if needed.

Health Personnel Management Records.png

Figure 5.11. Admin-Health Personnel Management Records

The image in Figure 5.11, shows a component of a mental health management system. It specifically focuses on admin managing health personnel records. Key features where admin can practice adding new personnel, searching for existing records, and viewing/editing/archiving individual personnel details. This interface streamlines the organization and management of healthcare professionals within the system.

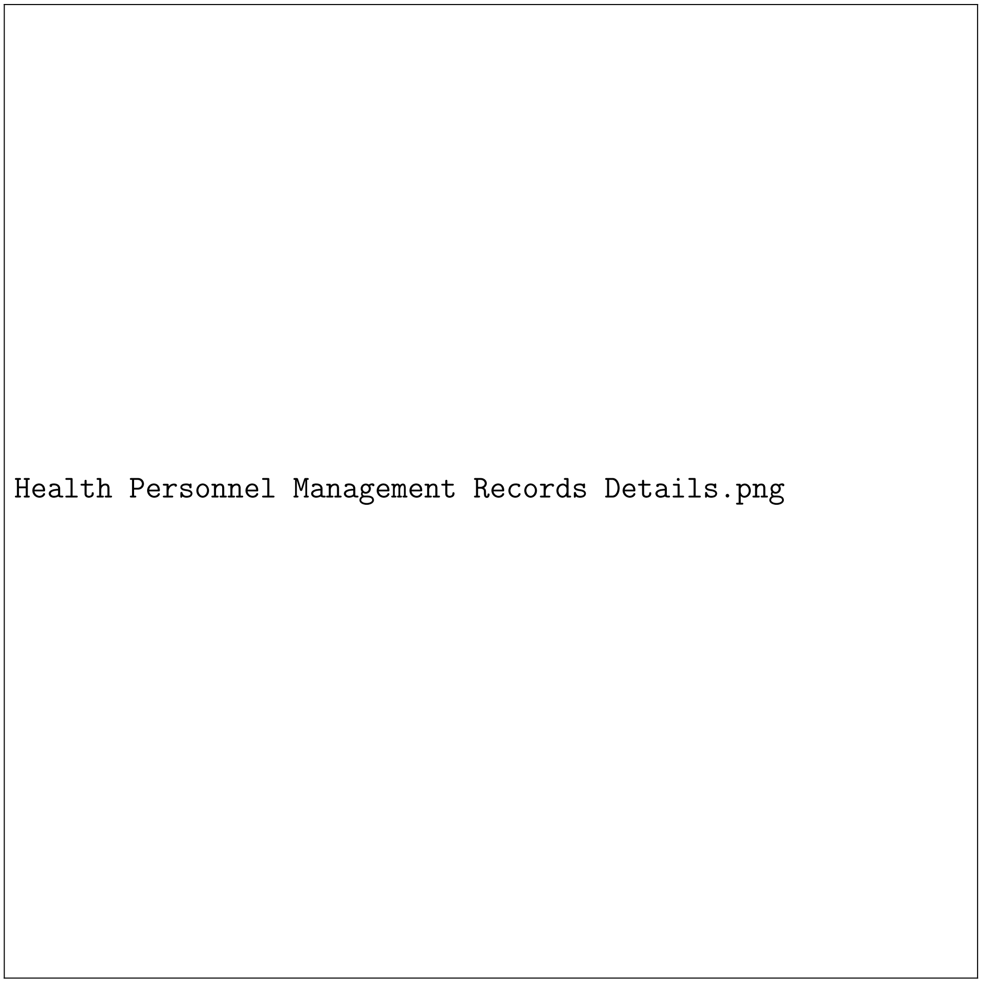

Figure 5.12. Admin-Health Personnel Management Records Details

Figure 5.12, shows a detailed view of a specific health personnel record within a mental health management system. This interface likely serves as a centralized repository for managing information about healthcare professionals who are assigned to the mental health field.

The interface displays detailed information about specific health personnel, including their name, contact information, specializations, and place of assignment. This level of detail allows for efficient management and tracking of individual healthcare professionals within the system.

Health Personnel Management Records (Edit Details).png

Figure 5.13. Admin-Health Personnel Management Records (Edit Details)

The image in Figure 5.13 shows, that the "Edit Health Personnel Record" interface in Figure 5.13 allows admins to manage healthcare professionals' information, including name, contact details, specializations, and assignments. The "Specializations" section enables granular categorization for efficient resource allocation. This interface ensures accurate and up-to-date records, aiding in efficient staffing, patient care, and overall healthcare facility management.

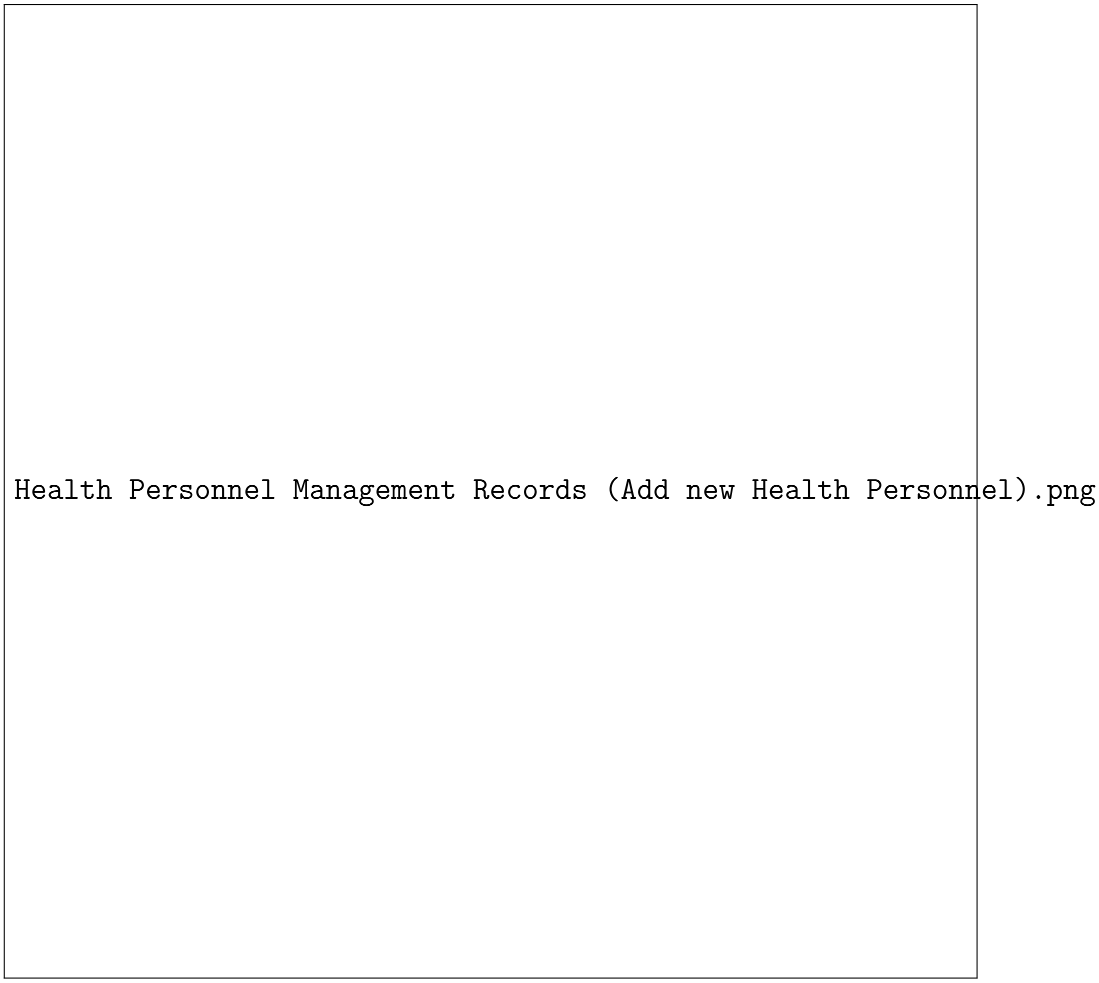
Health Personnel Management Records (Add new Health Personnel).png

Figure 5.14. Admin-Health Personnel Management Records (Add new Health Personnel)

Figure 5.14 shows, an ”Add New Health Personnel” interface that allows admins to input information about new healthcare professionals, including name, contact details, specializations, and assignments. The ”Specializations” section enables granular categorization. Once submitted, the system adds the new personnel to its database, streamlining onboarding and ensuring accurate records for efficient management.

Admin (Notifications).png

Figure 5.15. Admin (Notifications)

The Image in Figure 5.15 shows, that the notification center serves as a centralized hub for administrators to receive and manage critical updates related to patient care. This feature presents a chronological list of notifications, each categorized by time and type. For instance, the system may alert administrators about medication record updates, specifying the added medications, or inform them about newly booked appointments, providing details such as appointment ID, patient name, and current status. By offering a clear and organized presentation of notifications, this feature empowers administrators to stay informed, respond promptly to urgent matters, and ensure efficient management of patient care.

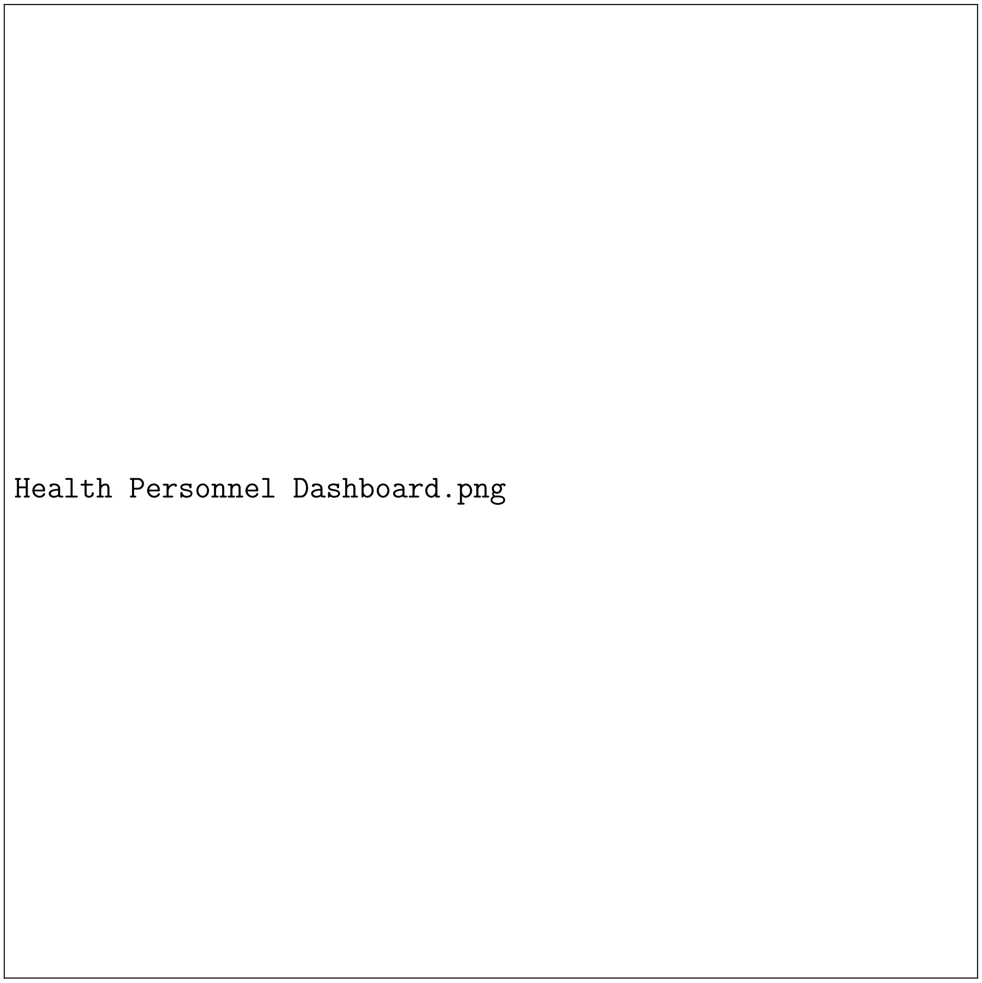

Figure 5.16. Health Personnel Dashboard

The image in Figure 5.16 shows, a Health Personnel Dashboard within the Mindcare mental health platform. This dashboard provides a centralized view of key metrics and functionalities essential for health personnel in managing the platform's operations. Also, the key features include tracking the total number of patients, health personnel, pending appointments, and medicine inventory. Additionally, the dashboard facilitates patient management, appointment scheduling, and medicine inventory management. By providing real-time data and insights, this dashboard empowers healthcare personnel to efficiently manage the platform and deliver quality mental health services.

Health Personnel-Patient Details.png

Figure 5.17. Health Personnel-Patient Details

The image in Figure 5.17 shows, a patient details interface within a mental health management system, likely designed for healthcare professionals and also for admin. This interface serves as a centralized repository that enables health personnel and admin to manage information about individual patients, facilitating efficient tracking and treatment.

The dashboard displays a variety of patient information, including personal details like name, contact information, and demographic data. It also includes clinical information such as mental disorder diagnosis, PhilHealth number, and disability status. Importantly, it tracks assessment dates and status, indicating the patient's current level of care and the need for further evaluation.

The "Impressions" section provides a space for health personnel to document their observations and insights about the patient, potentially aiding in treatment planning and decision-making. This interface, therefore, plays a crucial role in organizing and maintaining patient records, enabling efficient and effective mental health care delivery.

Health Personnel-Edit Patient.png

Figure 5.18. Health Personnel-Edit Patient

Figure 5.18 shows, an "Edit Patient" interface within a mental health management system, that is used by health Personnel and admin also. This interface serves as a centralized repository for health personnel to manage patient information.

The interface allows for editing a specific patient's details, including personal information, contact details, medical history, and assessment data. The "Patient Assessment" section provides a space for documenting the patient's current status and any necessary interventions.

This interface streamlines the process of updating patient records, ensuring accurate and up-to-date information for efficient patient care and treatment planning.

Health Personnel-Add Patient.png

Figure 5.19. Health Personnel-Add Patient

Figure 5.19 shows, an ”Add Patient” interface within the Mindcare mental health management system. This interface facilitates the creation of new patient records.

Healthcare Personnel and also admin can input essential patient information such as name, contact details, demographic data, mental disorder diagnosis, PhilHealth number, disability status, and assessment dates. The ”Patient Assessment” section allows for initial assessment categorization, potentially indicating the urgency of care. This interface streamlines the patient onboarding process, ensuring accurate and comprehensive records for efficient patient care and management.

Health Personnel-Generate Report.png

Figure 5.20. Health Personnel-Generate Report

the image in Figure 5.20 displays, a portion of the "Patient Record List" interface within the Mindcare mental health management system. This interface serves as a centralized repository for health personnel and also admin in managing patient records, facilitating efficient tracking and monitoring.

The list displays essential patient information such as ID, name, age, gender, barangay, mental disorder, last assessment date, medication, and action buttons. The "Generate Report" button likely generates a detailed report of patient records based on selected criteria, such as barangay. This interface streamlines the process of accessing and analyzing patient information, aiding in efficient patient care and management.

Health Personnel-Appointment Booking.png

Figure 5.21. Health Personnel-Appointment Booking

Figure 5.21 shows, a mental health management system's appointment booking interface. Users can input their names, select required services, add comments, and choose available time slots from a calendar view. The system likely integrates with a scheduling tool to manage appointments efficiently. The "pending" label suggests a verification process for bookings and the "approved label" means that health personnel and also admin accept patient appointments and be given a date for assessment.

Health Personnel-Booked Appointment.png

Figure 5.22. Health Personnel-Booked Appointment

The image in Figure 5.22 displays, a mental health management system's appointment record list. The user, which is the health personnel and admin, can view scheduled appointments, including patient names, services, comments, dates, times, and status. The system likely integrates with a calendar tool for scheduling and offers options to cancel appointments. The "Go to Calendar" button suggests a visual interface for managing appointments.

```
Health Personnel (Notifications).png
```

Figure 5.23. Health Personnel (Notifications)

Figure 5.23 shows, that the notification center serves as a centralized hub for healthcare providers to receive and manage critical updates related to patient care. This feature presents a chronological list of notifications, each categorized by time and type. For instance, the system may alert providers about medication record updates, specifying the added medications, or inform them about newly booked appointments, providing details such as appointment ID, patient name, and current status. By offering a clear and organized presentation of notifications, this feature empowers healthcare providers to stay informed, respond promptly to urgent matters, and ensure efficient management of patient care.

Medicine Inventory Records.png

Figure 5.23. Medicine Inventory Records

Figure 5.2 displays, a mental health management system's medicine inventory record. The health personnel and admin can view a list of medicines, including name, batch number, expiry date, batch supply, date received, milligram, dosage, and quantity. The system likely integrates with an inventory management tool to track stock levels. The "Add Medicine Stock," "Download

Report," and "Release Medicine" buttons suggest features for managing inventory and dispensing medication.

Medicine Inventory (Medicine Details).png

Figure 5.24. Medicine Inventory (Medicine Details)

Figure 5.24 shows, a mental health management system's medicine detail view. The health personnel and admin can view specific information about a particular medicine, including its name, expiry date, milligram dosage, quantity, batch number, batch supply, and batch date received. The system likely integrates with an inventory management tool to provide detailed information about each medicine. The "Cancel" button suggests a feature to cancel or modify the medicine record.

Medicine Inventory (Edit Medicine).png

Figure 5.25. Medicine Inventory (Edit Medicine)

The image in Figure 5.25 shows a mental health management system's medicine information editing screen. Where the health personnel and medicine, can modify details about a specific medicine, including its name, expiry date, milligram dosage, quantity, batch number, batch supply, and batch date received. The system likely integrates with an inventory management tool to allow for updates to medicine records. The "Update Medicine" and "Cancel" buttons suggest options to save changes or discard them.

Medicine Inventory (Add Medicine).png

Figure 5.26. Medicine Inventory (Add Medicine)

Figure 5.26 shows a mental health management system's "Add New Medicine" screen. The health personnel and admin can input details for a new medicine, including its name, expiry date, milligram dosage, batch supply, batch number, and batch date received. The system likely integrates with an inventory management tool to add new medicines to the stock. The "Add Medicine" and "Cancel" buttons suggest options to save the new medicine or discard the changes.

Medicine Inventory (Release Medicine).png

Figure 5.27. Medicine Inventory (Release Medicine)

Figure 5.27 shows a mental health management system's "Release Medicine to Patient" interface. This feature allows health personnel and admin to dispense prescribed medications to patients. The health personnel, can select a patient from a dropdown list and choose the specific medicine to be released. The system likely integrates with a patient management system to access patient information and a medicine inventory system to track medication stock. The "Dosage (times per day)" and "Quantity" fields allow for specifying the prescribed dosage and the amount of medication to be released. The "Close" button closes the dialog box, while the "Prescribe" button finalizes the medication release process and updates the patient's medical record and the inventory.

Medicine Inventory (Generated Reports).png

Figure 5.28. Medicine Inventory (Generated Reports)

Figure 5.28 displays a mental health management system's inventory report. The report provides a detailed overview of the current stock levels for various medications. Key information includes medicine names, batch numbers, expiry dates, batch supply, batch date received, milligram dosage, dosage frequency, and quantity. The system likely integrates with an inventory management tool to generate these reports. The report can be used to monitor stock levels, identify low-stock items, and plan for future orders or replenishments.

Patient Dashboard (Awareness).png

Figure 5.29. Patient Dashboard (Awareness)

The image in figure 5.29 displays, a patient dashboard within the Mindcare mental healthcare management system. The dashboard provides a centralized hub for patients to access and manage their healthcare information. The primary features visible on the dashboard include a ”Dashboard” tab, likely providing an overview of the patient’s health information, and an ”Appointment Booking” tab, allowing patients to schedule and manage their appointments. The prominent red banner at the top draws attention to a ”Suicide Awareness” message, emphasizing the system’s commitment to mental health support and encouraging patients to be aware of suicide warning signs. The overall design of the dashboard is clean and intuitive, with clear navigation options and a focus on providing essential information to patients.

Patient Dashboard (Patient Info.).png

Figure 5.30. Patient Dashboard (Patient Info.)

Figure 5.30 which was also part of the patient dashboard presents, a detailed profile of a patient's demographic and health information. It includes essential fields like personal details, contact information, and medical history. Notably, the system tracks the patient's mental disorder, PhilHealth number, and disability status. The interface also records crucial assessment data, including the date of the first assessment, the scheduled date for the next assessment, and a critical assessment status, visually highlighted in red. This comprehensive record empowers healthcare providers with the necessary insights to deliver targeted and timely care, ensuring the optimal well-being of the patient.

Patient Dashboard (Booked Appointment).png

Figure 5.31. Patient Dashboard (Booked Appointment)

Figure 5.31 shows, a user-friendly interface for patients to visualize and manage their scheduled appointments. This interactive calendar displays a month-long view, and color-coding appointments based on their status: pending or approved. This visual representation provides a clear overview of upcoming appointments, allowing patients to easily identify and plan for their healthcare needs. Additionally, the calendar's navigation buttons enable seamless exploration of past and future months, empowering patients to take proactive control of their healthcare journey.

Patient Dashboard (Appointment Booking).png

Figure 5.32. Patient Dashboard (Appointment Booking)

The Mindcare system facilitates a seamless appointment booking experience for patients. The intuitive interface guides users through a straightforward process. Patients begin by selecting a preferred date from the calendar, specifying the required service, and optionally adding any relevant comments. Subsequently, they choose an available time slot from the displayed options. Upon confirmation, the appointment is securely added to the patient's calendar, streamlining the scheduling process and ensuring timely access to essential healthcare services.

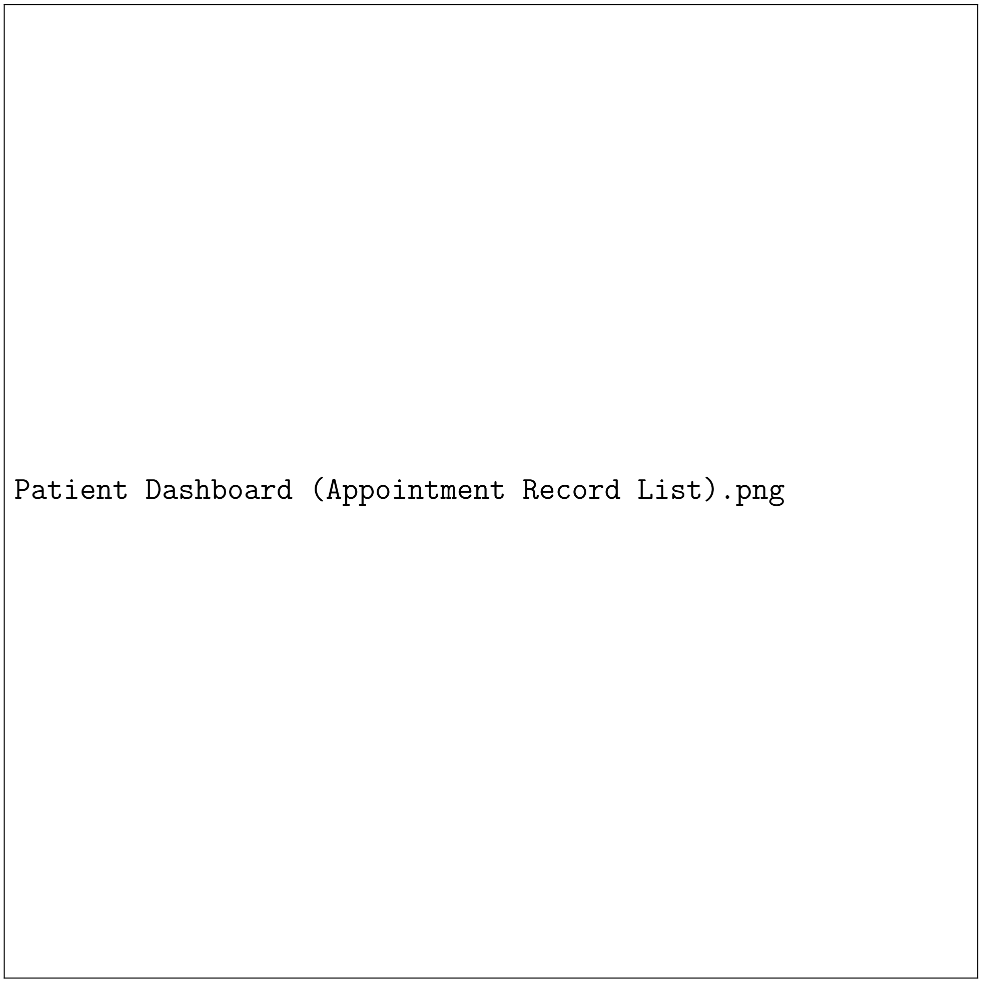

Figure 5.33. Patient Dashboard (Appointment Record List)

The image in Figure 5.33 shows, that the appointments record list provides a centralized repository of a patient's scheduled appointments. This comprehensive overview displays essential details such as appointment ID, service provider, type of service, any specific comments, date, time, current status, and available actions. By offering a clear and organized presentation of appointment information, this feature empowers patients to efficiently manage their healthcare schedule, ensuring timely access to necessary services.

Patient Dashboard (Notifications).png

Figure 5.34. Patient Dashboard (Notifications)

Figure 5.34 employs, a notification system to keep patients informed about important updates, such as appointment cancellations. The notification interface displays pertinent details including the appointment ID, name, and status. The "Mark as Read" feature allows patients to acknowledge and dismiss notifications, streamlining the communication process. This notification system ensures that patients remain updated on any changes to their appointments, fostering transparency and effective communication between patients and healthcare providers.

Figure 5.35. User Profile Information

Figure 5.35 shows, a dedicated profile page where users can access and manage users personal information. This centralized hub displays essential details such as username, full name, and verified email address. Furthermore, the page offers the functionality to update the password, ensuring the security of the user's account. By granting users control over their personal information, Mindcare prioritizes privacy and empowers individuals to maintain a secure and personalized experience within the system.

5.3 Implementation Plan

The requirements analysis using structured system analysis and design was described in the previous chapter, and the functionality requirements to specify the system capabilities and function that it must layout for its user interface (UI) will be done using CSS and Bootstrap and the connection between the database and the system is established using MYSQL and PHP language along with its free and open source web framework called LARAVEL.

The system was deployed on a cloud environment and made sure that the quality of the system was still the same when it was on localhost. The proponents initially deployed the system through Hostinger. Since the system can be successfully deployed on a cloud environment, the clients, if to implement the system, can also deploy the web system. To do so, the following are the requirements for implementing the project:

- The source code can be referred to in Appendix A or the backup external storage device where the proponents included the source code.
- A Version control system and platform, preferably Git or Github.
- An Account to Hostinger
- A computer that has at least the minimum requirement of the software and hardware specification specified in Chapter 3.

Assistance about the system specifically on how to use it is also required. The clients or the developer can refer to the prototype user's manual located in Chapter 5.2 to guide them on how to use the system. This can also be helpful to the users to understand and know how to operate the system with minimal assistance from the developer.

5.4 Implementation Result

The system was implemented, deployed, and reviewed successfully. During the implementation, the proponents started implementing the smallest features of the components and integrating them until the whole system was successful. The system was deployed through a cloud servicing platform, especially through Hostinger. The system was reviewed by the clients and they were able to create suggestions and recommendations to further enhance the system. The proponents were able to implement some of the suggestions and recommendations successfully. Some were outside of the scope of the study but will be able to enhance the functionality of the system if implemented. Those limitations will be discussed further in this paper.

RESPONDENTS	TOTAL NUMBER OF RESPONDENTS
Administrator	1
Health Personnel	3
Patients	30

Table 15: Summary Of Respondents

The above table (Table 15) shows, the total number of respondents as the system evaluators. It consists of 1 Administrator, 3 Health Personnel, and 30 Patients who are considered as the users of the system.

The following were the results of the implementation of the system in accordance with its objective.

1. Features and modules related to monitoring, tracking, booking, recording, and generating reports were successfully implemented in the system. The features and modules were the following: Patient Record Management, Health Personnel Record Management, User Management, Report Generator, Appointment Booking, Calendar Viewing, All Scheduled Appointment Table List, and Medicine Inventory.
2. The system was successfully tested and evaluated based on ISO/IEC 9126-1 Software Product Quality. Discussed below were the results of the evaluations of the respondents. The

respondents were composed of 1 Administrator, 3 Health Personnel, and 30 Patients of Rural Health Unit in Bulan Sorsogon.

QUESTIONNAIRE	MEAN	VERBAL INTERPRETATION
The system allows me to add, edit, and delete patient records effectively.	5	Strongly Agree
I can easily manage medicine inventory using the system.	5	Strongly Agree
The system enables me to monitor patient progress and history.	5	Strongly Agree
The features of the system meet the requirements of my tasks.	5	Strongly Agree
The system accurately display appointments slots for specific services.	5	Strongly Agree
I can generate accurate reports when needed.	5	Strongly Agree
Systems updates improve the functionality of the system.	5	Strongly Agree
The system helps reduce manual process effectively.	5	Strongly Agree
AVERAGE	**5**	**Strongly Agree**

Table 16: Functionality of the System (Admin Response)

Table 16 showed the result of the evaluation of the system that was anwered by the administrator in terms of its functionality. Functionality has the following 8 questions that was mentioned in the table above. All the 8 questions under the Functionality received a weighted mean of 5 or “Strongly Agree“. Overall, Functionality has an average of 5 which can be interpreted as Strongly Agree.

6 Conclusion and Recommendation

This chapter aimed to present the conclusions drawn and recommendations made on the conduct of this study. The challenges the proponents encountered and the limitations of the system were also discussed in this chapter.

6.1 Conclusion

Based on the findings and implementations of this study, the following are formulated.

1. The implementation of the Mental Health Record Management system at RHU Bulan has significantly impacted the delivery of mental healthcare services. By digitizing patient records, the system has enhanced data accuracy and accessibility, streamlining information retrieval and reducing the potential for errors. The centralized database has facilitated efficient tracking of patient history, treatment plans, and progress, empowering healthcare professionals to make informed decisions.

   Moreover, the system's robust reporting capabilities have provided valuable insights into patient demographics, treatment trends, and resource utilization. These insights have informed evidence-based practices, identified areas for improvement, and optimized resource allocation. By automating routine tasks and minimizing paperwork, the system has alleviated administrative burdens, enabling healthcare providers to prioritize patient care.

   Ultimately, the successful implementation of Mindcare has contributed to the overall enhancement of mental healthcare services at RHU Bulan, leading to improved patient outcomes and optimized resource utilization.

2. The integration of the Appointment Scheduling module within Mindcare has significantly enhanced the efficiency and organization of mental healthcare services at RHU Bulan. By automating the appointment booking process, the system eliminated manual scheduling errors and minimized patient wait times. The intuitive calendar viewing feature empowered healthcare providers to visualize their schedules, identify potential conflicts, and optimize time management.

   Moreover, the centralized appointment database facilitated efficient tracking of patient appointments, reducing the likelihood of missed or rescheduled appointments. By generating detailed reports on appointment trends, utilization rates, and no-show rates, the system provided valuable insights to optimize scheduling practices and resource allocation. Ultimately, the implementation of the Appointment Scheduling module contributed to a more efficient and patient-centered healthcare delivery system at RHU Bulan.

3. The implementation of the Medicine Inventory module, a critical component of Mindcare, has significantly enhanced medication management practices at RHU Bulan. By incorporating features such as stock-in and stock-out tracking, release management, dosage information, batch number tracking, batch supply details, and date of receipt, the system has provided a robust solution for optimizing medication management.

   Real-time monitoring of stock levels has enabled proactive alerts for potential shortages, allowing for timely replenishment and minimizing disruptions to patient care. Tracking medication expiration dates has ensured timely disposal of expired drugs, promoting patient safety and regulatory compliance. The integration of dosage information, batch numbers, and batch supply details has facilitated accurate medication administration and traceability.

   By automating inventory tasks and generating insightful reports on medication usage patterns, the Medicine Inventory module has empowered healthcare professionals to make informed decisions regarding medication procurement, storage, and administration. Ultimately, the implementation of this module has contributed to a more efficient, safe, and cost-effective medication management system at RHU Bulan.

4. The evaluation of Mindcare using ISO/IEC 9126-1 standards provided valuable insights into the system's performance and identify areas for improvement. By assessing the system's functionality, usability, and portability, the research team ensured that Mindcare met the specific needs of RHU Bulan's healthcare providers and patients.

   The functionality evaluation assessed the system's ability to perform its intended tasks, such as managing patient records, scheduling appointments, and tracking medication inventory. The usability evaluation focused on the system's ease of use, user interface design, and overall user experience. The portability evaluation assessed the system's ability to operate on different devices and platforms, ensuring flexibility and accessibility for healthcare providers.

   By addressing identified shortcomings and implementing necessary improvements, the research team enhanced the overall quality and user experience of Mindcare, ultimately contributing to the optimization of mental healthcare services at RHU Bulan.

6.2 Recommendation

Based on the conclusions, the following recommendations are hereby offered:

1. Implement a Chatbot: Develop a chatbot to provide immediate assistance to patients and healthcare personnel, answering common queries and offering support.

2. Notification Alerts: Incorporate a notification system to alert users about approved appointments, upcoming assessments, and overdue assessments.

3. User Customization: Allow users to personalize their system settings, such as preferred language, theme, and notification preferences.

References

[1] Sharma, N.: The presistent of manual processes in healthcare. Mental Health Manual Processes (2023)

[2] Taylor, C.B., Fitzsimmons-Craft, E.E., Graham, A.K.: Digital technology can revolutionize mental health services delivery: The covid-19 crisis as a catalyst for change. International Journal of Eating Disorders **53**(7), 1155–1157 (2020)

[3] Aburayya, A., Al Marzouqi, A., Al Ayadeh, I., Albqaeen, A., Mubarak, S.: Evolving a hybrid appointment system for patient scheduling in primary healthcare centres in dubai: Perceptions of patients and healthcare provider. International Journal on Emerging Technologies **11**(2), 251–260 (2020)

[4] Drossman, D.A., Ruddy, J.: Improving patient-provider relationships to improve health care. Clinical Gastroenterology and Hepatology **18**(7), 1417–1426 (2020)

[5] Callaly, T., Faulkner, P., Hollis, G.: The development of a mental health service patient information management system. Australian Health Review **21**(3), 182–193 (1998)

[6] Zala, K., Thakkar, H.K., Jadeja, R., Singh, P., Kotecha, K., Shukla, M.: Prms: Design and development of patients' e-healthcare records management system for privacy preservation in third party cloud platforms. IEEE Access **10**, 85777–85791 (2022) https://doi.org/10.1109/ACCESS.2022.3198094

[7] Kaur, J., Verma, V.C., Kumar, V., Singh, R., Bhatia, T., Sahu, S., Manak, M., Buttolia, H.K., Choudhary, B., Sharma, Y.S., *et al.*: i-mann: a web-based system for data management

of mental health research in india. Indian Journal of Psychological Medicine **42**(6_suppl), 15–22 (2020)

[8] Assis, A.G., Dos Santos, A.F.A., Dos Santos, L.A., Costa, J.F., Cabral, M.A.L., Souza, R.P.: Classification of medicines and materials in hospital inventory management: a multi-criteria analysis. BMC Medical Informatics and Decision Making **22**(1), 325 (2022)

[9] S, K., Refonaa, J., Jany Shabu, S.L., Dion Paul, K., Dhamodaran, S., Mary, V.A.: Medistock: Medical inventory management system. In: 2023 4th International Conference on Electronics and Sustainable Communication Systems (ICESC), pp. 738–743 (2023). https://doi.org/10.1109/ICESC57686.2023.10193177

[10] Refonaa, J., Shabu, S.J., Paul, K.D., Dhamodaran, S., Mary, V.A., *et al.*: Medistock: Medical inventory management system. In: 2023 4th International Conference on Electronics and Sustainable Communication Systems (ICESC), pp. 738–743 (2023). IEEE

[11] Anif, M., Putra, A.S., Ernawati, D., Prabuwono, A.S., *et al.*: Hometrack: Rfid-based localization for hospital medicine tracking system. In: 2015 2nd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), pp. 449–453 (2019). IEEE

[12] Adi, S., Arief Hermawan, M.: Design and development of a web-based drug inventory system at janitra farma pharmacy. PhD thesis, Universitas Teknologi Yogyakarta (2021)

[13] Benjaafar, S., Chen, D., Wang, R., Yan, Z.: Appointment scheduling under a service-level constraint. Manufacturing & Service Operations Management **25**(1), 70–87 (2023)

[14] Hancerliogullari, K.O.: Appointment scheduling in healthcare systems: A scientometric review. In: Global Joint Conference on Industrial Engineering and Its Application Areas, pp. 333–341 (2021). Springer

[15] Samartzis, L., Talias, M.A.: Assessing and improving the quality in mental health services. International journal of environmental research and public health **17**(1), 249 (2020)

[16] Carruthers, A.: The snowflake data cloud. In: Building the Snowflake Data Cloud: Monetizing and Democratizing Your Data, pp. 3–27. Springer, ??? (2022)

[17] Brimelow, R.E., Amalathas, A., Beattie, E., Byrne, G., Dissanayaka, N.N.: The use of balanced scorecards in mental health services: an integrative review and thematic analysis. The Journal of Behavioral Health Services & Research **50**(1), 128–146 (2023)

[18] Stahl, B.: DeepSIP: Deep Learning of Supernova Ia Parameters, 0.42, Astrophysics Source Code Library (2020), ascl:2006.023

[19] MDsyncNET.com: Medical scheduling basics: What you need to know (2023)

[20] Dosage in Medicine Inventory. https://www.google.com/search?q=dosage+in+medicine+inventory (2023)

[21] Milligram in Medicine Inventory. https://www.google.com/search?q=milligram+in+medicine+inventory (2023)

[22] Expiry Date in Medicine Inventory. https://www.google.com/search?q=expiry+date+in+medicine+inventory (2023)

[23] Quantity in Medicine Inventory. https://www.semanticscholar.org/paper/f799737093e6913106795339097330 (2024)

[24] Batch Supply in Medicine Inventory. https://www.semanticscholar.org/paper/f799737093e6913106795339097330 (2024)

[25] Batch Number in Medicine Inventory. https://www.google.com/search?q=batch+number+in+medicine+inventory (2023)

[26] Prescription in Medicine Inventory. https://www.semanticscholar.org/paper/f799737093e6913106795339097330 (2024)

[27] Intellectsoft: Complete overview of medication tracking software. Intellectsoft Blog (2021)

[28] Stock-In in Medicine Inventory. https://www.semanticscholar.org/paper/f799737093e6913106795339097330 (2024)

[29] Stock-Out in Medicine Inventory. https://www.semanticscholar.org/paper/f799737093e6913106795339097330 (2024)

[30] Archive in Mental Health Management System. https://www.semanticscholar.org/paper/

f799737093e6913106795339097330 (2024)

[31] Restoring Archived Information in Mental Health Management Systems. https://www.semanticscholar.org/paper/f799737093e6913106795339097330 (2024)

[32] Calendar Viewing in Mental Health Management Systems. https://www.semanticscholar.org/paper/f799737093e6913106795339097330 (2024)

[33] Online Status Description for Mental Health Management System. This is a description of online status features for a mental health management system, designed to enhance real-time communication and collaboration between patients and healthcare providers. (2024)

[34] Offline Status Description for Mental Health Management System. This is a description of offline status features for a mental health management system, designed to enhance user experience and manage expectations. (2024)

[35] Email Verification for a Web-Based Mental Health Management System. A description of email verification for a web-based mental health management system, written with the readability of a university paper. (2024)

[36] Notification Alert Messages for a Web-Based Mental Health Management System. A description of notification alert messages for a web-based mental health management system, written with the readability of a university paper. (2024)

[37] Services, A.W.: What is Architecture Diagramming? - Software and System Architecture Diagramming Explained (2021). https://aws.amazon.com/what-is/architecture-diagramming/

[38] IBM: Use-case diagrams in UML modeling (2019). https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-use-case

[39] (OMG), O.M.G.: Unified modeling language™ (omg document number: formal/2017-10-02). Technical report, Development Organization (2017)

[40] Elmasri, R., Navathe, S., Elmasri, R., Navathe, S.: Fundamentals of database systems¡/title. In: Advances in Databases and Information Systems: 24th European Conference, ADBIS 2020, Lyon, France, August 25–27, 2020, Proceedings, vol. 12245, p. 139 (2020). Springer

Nature

[41] Lo, D.C.-T., Karam, O.: Enhance capstone projects with a new online collaboration system. In: 2013 IEEE 13th International Conference on Advanced Learning Technologies, pp. 217–218 (2013). IEEE

[42] GeeksforGeeks: Object-Oriented Analysis and Design (OOAD) (2024). https://www.geeksforgeeks.org/object-oriented-analysis-and-design/

[43] simplilearn: What is Requirement Analysis (2023). https://www.simplilearn.com/what-is-requirement-analysis-article

[44] Business Analysis., I.I.: A Guide to the Business Analysis Body of Knowledge (BABOK). International Institute of Business Analysis, ??? (2015)

[45] Gorbachenko, P.: Functional vs Non-Functional Requirements (2024). https://enkonix.com/blog/functional-requirements-vs-non-functional/

[46] Awaliah, N., Hendra, A., Amiruddin, A., Daud, D., Iskandar, A.: Web-based rapid application development (rad) for marketing of ende lio traditional bond motif woven fabric. Ceddi Journal of Information System and Technology (JST) **2**(1), 38–43 (2023)

[47] Agarwal, A.: Software engineering — system design strategy. GeeksforGeeks. https://www.geeksforgeeks.org/software-engineering-systemdesign-strategy/

[48] Tutorialspoint: Object Oriented Approach. Tutorialspoint. https://www.tutorialspoint.com/system_analysis_and_design/system_-analysis_and_design_object_oriented_approach.htm

[49] Software, E.: What is HTML5. https://www.ericom.com/glossary/what-is-html5/

[50] Wikipedia: CSS - Wikipedia (2019). https://en.wikipedia.org/wiki/CSS

[51] Blog, H.: What is Tailwind Css. https://developers.hubspot.com/blog/creating-a-hubspot-website-with-tailwind-css

[52] Bootstrap v5.0. Accessed: 2023-11-10. https://getbootstrap.com/

[53] Wikipedia: JavaScript - Wikipedia (2024). https://en.m.wikipedia.org/wiki/JavaScript

[54] W3Schools: W3Schools MySQL Tutorial. https://www.w3schools.com/mysql/

[55] MyPHP.net: What is PHP. https://www.php.net/

[56] Otwell, T., Laravel community: Laravel - Wikipedia (2024). https://en.wikipedia.org/wiki/Laravel

[57] XAMPP. Accessed November 8, 2024. https://www.apachefriends.org/index.html

[58] Apache HTTP Server. Accessed November 8, 2024. https://httpd.apache.org/

[59] Wikipedia: Visual Studio Code (2024). https://en.wikipedia.org/wiki/Visual_Studio_Code

[60] Wikipedia: Git (2024). https://en.wikipedia.org/wiki/Git

Appendix A :Resource Code

LoginController.php

```
<?php
namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\AuthenticatesUsers;
use Illuminate\Http\Request;
use Illuminate\Http\RedirectResponse;
use Illuminate\Support\Facades\Auth;
use App\Models\User;
use Illuminate\Support\Facades\Log;

class LoginController extends Controller
{
    /*
```

```
|--------------------------------------------------------------------------
--------------------
| Login Controller
|--------------------------------------------------------------------------
--------------------
|
| This controller handles authenticating users for the
application and
| redirecting them to your home screen. The controller
uses a trait
| to conveniently provide its functionality to your
applications.
|
*/
use AuthenticatesUsers;
/**
 * Where to redirect users after login.
 *
 * @var string
 */
protected $redirectTo = '/home';
/**
 * Create a new controller instance.
 *
 * @return void
 */
public function __construct()
```

```
    {
        $this->middleware('guest')->except('logout');
        $this->middleware('auth')->only('logout');
    }
    public function login(Request $request): RedirectResponse
    {
        $input = $request->all();
        $this->validate($request, [
            'username' => 'required|string|max:255',
            'password' => 'required|string|min:8',
        ]);


        if (auth()->attempt(['name' => $input['username'],
        'password' => $input['password']]) ||

            auth()->attempt(['username' => $input['username'],
            'password' => $input['password']])) {

            $user = Auth::user();
            $user->status = true; // Online
            $user->save();

            session()->flash('success', 'Successfully␣logged
␣␣␣␣␣␣␣␣␣␣␣␣in.');

            if (auth()->user()->type == 'admin') {
                return redirect()->route('admin.home');
            }else if (auth()->user()->type == 'therapist') {
                return redirect()->route('therapist.home');
            }else{
                return redirect()->route('home');
            }
        }else{
            return redirect()->route('login')
                ->with('error','Email-Address And Password Are
                Wrong.');
        }

    }

    public function logout(Request $request): RedirectResponse
    {

        Log::info('User logged out.');
        $user = Auth::user();
        if ($user) {
            // Set user status to offline
            $user->status = false; // Offline
            $user->save();
        }


        Auth::logout();
        $request->session()->invalidate();
        $request->session()->regenerateToken();

        return redirect()->route('login');
    }

}
```

RegisterController.php

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Foundation\Auth\RegistersUsers;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
use App\Mail\VerificationEmail;
use Illuminate\Support\Facades\Mail;

class RegisterController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    ----------
    | Register Controller
```

```
|--------------------------------------------------------------------------------------------
---------------------------
|
| This controller handles the registration of new users as
well as their
| validation and creation. By default this controller uses
a trait to
| provide this functionality without requiring any
additional code.
|
*/

use RegistersUsers;

/**
 * Where to redirect users after registration.
 *
 * @var string
 */
protected $redirectTo = '/home';

/**
 * Create a new controller instance.
 *
 * @return void
 */
public function __construct()
```

```
    {
        $this->middleware('guest');
    }

    /**
     * Get a validator for an incoming registration request.
     *
     * @param  array  $data
     * @return \Illuminate\Contracts\Validation\Validator
     */
    protected function validator(array $data)
    {
        return Validator::make($data, [
            'username' => ['required', 'string', 'max:255',
            'unique:users'],
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email',
            'max:255', 'unique:users'],
            'password' => ['required', 'string', 'min:8',
            'confirmed'],
        ]);
    }

    /**
     * Create a new user instance after a valid registration.
     *
     * @param  array  $data
     * @return \App\Models\User
     */
    protected function create(array $data)
    {
        return User::create([
            'username' => $data['username'],
            'name' => $data['name'],
            'email' => $data['email'],
            'password' => Hash::make($data['password']),
        ]);
         // Send the verification email
        Mail::to($user->email)->send(new VerificationEmail());

        return $user;
    }
}
```

AppointmentController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\Appointment;
use App\Models\Services;
use App\Models\Therapist;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;
use Carbon\Carbon;
use Illuminate\Support\Facades\Auth;
use App\Notifications\AppointmentStatusChanged;

class AppointmentController extends Controller
{


    public function index()
    {
        $appointments = Appointment::with('services')-
        >orderBy('date', 'asc')->get();
    //  $therapists = User::where('type', 2)->get(); //'2' is
    the type for therapists
        $services = Services::all();

        return view('calendar.index', compact('appointments',
        'services'));
    }


    public function store(Request $request)
    {
        Log::info($request->all());

        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'services_id' => 'required|exists:services,id',
            'comments' => 'nullable|string',
            'date' => 'required|date',
            'time' => 'required'
        ]);

        // Check if the slot is already booked
        $existingAppointment = Appointment::where('date',
        $validated['date'])
            ->where('time', $validated['time'])
            ->first();

        if ($existingAppointment) {
            return redirect()->back()->withErrors(['message'
            => 'This␣time␣slot␣is␣already␣booked.']);
        }

        $appointment =  Appointment::create([
            'name' => $validated['name'],
            'services_id' => $validated['services_id'],
            'comments' => $validated['comments'],
            'date' => $validated['date'],
            'time' => $validated['time'], // Save the selected
            time
        ]);

```

```
// Notify the patient
        $patient = User::where('name', $appointment->name)-
        >first(); // assuming name matches the patient's name
        if ($patient) {
            $patient->notify(new AppointmentStatusChanged("You
            Booked an Appointment.", [
                'appointment' => $appointment
            ]));
        }
            // Notify the therapist and admin
        $usersToNotify = User::whereIn('type', [1, 2])->get();
        // 1=admin, 2=therapist
        foreach ($usersToNotify as $user) {
            $user->notify(new AppointmentStatusChanged("A new
            appointment has been booked.", [
                'appointment' => $appointment
            ]));
        }

        return redirect()->route('calendar.index')-
        >with('success', 'Appointment booked successfully.');
    }


    public function availableSlots($date)
    {

        $bookedSlots = Appointment::where('date', $date)
        ->where('status', '!=', 'cancel') // Exclude canceled
        appointments
        ->pluck('time')
        ->toArray();

        $allSlots = [
            '07:00 AM', '08:00 AM', '09:00 AM',
            '10:00 AM', '11:00 AM', '01:00 PM',
            '02:00 PM', '03:00 PM', '04:00 PM'
        ];

        $availableSlots = array_map(function ($slot) use
        ($bookedSlots) {
            return ['time' => $slot, 'booked' =>
            in_array($slot, $bookedSlots)];
        }, $allSlots);

        return response()->json($availableSlots);
    }


    public function approve($id)
    {

        $appointment = Appointment::find($id);

        if ($appointment) {
            $appointment->status = 'approved';
            $appointment->save();

        // Notify the patient
        $patient = User::where('name', $appointment->name)-
        >first(); // assuming name matches the patient's name
        if ($patient) {
            $patient->notify(new
            AppointmentStatusChanged("Your appointment has
            been approved.", [
                'appointment' => $appointment
            ]));
        }

        // Notify the therapist and admin
       $usersToNotify = User::whereIn('type', [1, 2])->get();
       // 1=admin, 2=therapist
        foreach ($usersToNotify as $user) {
            $user->notify(new AppointmentStatusChanged("A new
            appointment has been booked.", [
                'appointment' => $appointment
            ]));
        }


        return redirect()->back()->with('success',
        'Appointment approved successfully.');
    }


        return redirect()->back()->with('error', 'Appointment
        not found.');
    }


    public function cancel($id)
    {
        $appointment = Appointment::find($id);


        if ($appointment) {
            // Update the appointment status to 'cancel'
            $appointment->status = 'cancel';
            $appointment->save();  // Save the update to the
            database

        // Notify the patient
        $patient = User::where('name', $appointment->name)-
        >first(); // assuming name matches the patient's name
        if ($patient) {
            $patient->notify(new
            AppointmentStatusChanged("Your appointment has
            been canceled.", [
                'appointment' => $appointment
            ]));
        }

        return redirect()->back()->with('success',
        'Appointment canceled successfully.');
        }

        return redirect()->back()->with('error', 'Appointment
        not found.');
    }


    public function allBooked()
    {
        // Get the authenticated user
        $user = Auth::user();

        // Initialize appointments query with relationship
        loading
        $appointments = Appointment::with('services')-
        >orderBy('date', 'asc');


        // Check the user type to filter appointments
        if ($user->type == 'admin' || $user->type ==
        'therapist') {
            // Admin and Therapist both see all appointments
            $appointments = $appointments->get();
        } elseif ($user->type == 'patient') {
            // Patient sees only their own approved
            appointments
            $appointments = $appointments->where('name', $user-
            >name) // assuming name matches user for patient
            appointments
                ->where('status', 'approved')
                ->get();
        }

        return view('calendar.allBooked',
        compact('appointments'));
    }

    public function notifications()
    {
        $user = Auth::user();

        if ($user) {
            $notifications = $user->unreadNotifications; //
            Only unread notifications

        } else {
            $notifications = [];
        }


        return view('calendar.notification',
        compact('notifications'));
    }


    public function markAsRead($id)
    {
        $user = Auth::user();

        if (!$user) {
            return redirect()->route('login')->with('error',
            'Please log in to mark notifications as read.');
        }

       $notification = $user->unreadNotifications->find($id);

        $notification->markAsRead();

        return redirect()->route('notifications')-
        >with('success', 'Notification marked as read.');
    }



}
```

Controller.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
use Illuminate\Foundation\Validation\ValidatesRequests;
use Illuminate\Routing\Controller as BaseController;
use App\Models\Patient;

class Controller extends BaseController
{
    use AuthorizesRequests, ValidatesRequests;

}
```

HomeController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\View\View;
use App\Models\User;
use App\Notifications\UserActivityNotification;
use App\Models\Patient;
use App\Models\Inventory;
use App\Models\Therapist;
use App\Models\Appointment;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Log;

class HomeController extends Controller
{
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
    }

    /**
     * Show the application dashboard.
     *
     * @return \Illuminate\Contracts\Support\Renderable
     */
    public function index(): View
    {
        // Fetch inventory data

        // Fetch total patients (if needed elsewhere)
        $totalPatients = Patient::count();

        return view('home', compact('totalPatients'));
    }
     /**
     * Show the application dashboard.
     *
     * @return \Illuminate\Contracts\Support\Renderable
     */

     public function home(): View
     {
        // Get the currently authenticated user
        $user = Auth::user();

        // Ensure the user is authenticated
        if (!$user) {
            return redirect()->route('login'); // Redirect to
            login if not authenticated
        }

        // Fetch the patient record using the relationship
        based on email
        $patientRecord = $user->patient;

        $notifications = $user->unreadNotifications;

        // Log unread notifications
        Log::info($user->unreadNotifications);


        return view('home', compact('patientRecord',
        'notifications'));
    }


    public function adminHome(): View
    {
        // Fetch the count of patients with each mental
        disorder type
        $mentalDisorderData = DB::table('patients')
        ->select('mental_disorder_type', DB::raw('COUNT(*) as
        total'))
        ->groupBy('mental_disorder_type')
        ->get();
        // Fetch the total count of each medication per
        barangay
        $medicationData = DB::table('patients')
            ->join('patient_inventory', 'patients.id', '=',
            'patient_inventory.patient_id')
            ->join('inventory',
            'patient_inventory.inventory_id', '=',
            'inventory.id')
            ->select('patients.barangay', 'inventory.name as
            medication', DB::raw('count(*) as
            total_patients_with_medication'))
            ->groupBy('patients.barangay', 'inventory.name')
            ->get();

        // Fetch medication names and quantities
        $inventoryData = Inventory::select('name', 'quantity')-
        >get();

        // Calculate the total quantity of all inventory items
        $totalQuantity = Inventory::sum('quantity');

        $therapistCount = Therapist::count(); // Get the total
        number of therapists

        // Fetch all pending appointments
        $pendingAppointments = Appointment::where('status',
        'pending')->orderBy('date', 'asc')->get();

        // Count the number of patients with each assessment
        status
        $assessmentStatuses = ['critical', 'progressive',
        'almost_recovered', 'recovered'];
        // Fetch all approved appointments
        $approvedAppointments = Appointment::where('status',
        'approved')->orderBy('date', 'asc')->get();

        $patientCounts = [];
        foreach ($assessmentStatuses as $status) {
            $patientCounts[$status] =
            Patient::where('assessment_status', $status)-
           >count();
        }

            // Fetch the count of male and female patients
            $malePatients = Patient::where('gender', 'M')-
           >count();
            $femalePatients = Patient::where('gender', 'F')-
           >count();

        return view('adminHome',
        compact('medicationData','inventoryData',
        'therapistCount', 'pendingAppointments',
        'totalQuantity', 'patientCounts',
        'approvedAppointments', 'mentalDisorderData',
        'malePatients', 'femalePatients'));
    }

    /**
     * Show the application dashboard.
     *
     * @return \Illuminate\Contracts\Support\Renderable
     */
    public function therapistHome(): View
    {
        // Fetch the count of patients with each mental
        disorder type
        $mentalDisorderData = DB::table('patients')
        ->select('mental_disorder_type', DB::raw('COUNT(*) as
         total'))
        ->groupBy('mental_disorder_type')
        ->get();


          // Fetch medication names and quantities
          $inventoryData = Inventory::select('name',
          'quantity')->get();

          // Calculate the total quantity of all inventory
          items
          $totalQuantity = Inventory::sum('quantity');

          $therapistCount = Therapist::count(); // Get the
          total number of therapists

          // Fetch all pending appointments
          $pendingAppointments =
          Appointment::where('status', 'pending')-
          >orderBy('date', 'asc')->get();

```

```
            // Count the number of patients with each
            assessment status
            $assessmentStatuses = ['critical', 'progressive',
            'almost_recovered', 'recovered'];
            // Fetch all approved appointments
            $approvedAppointments =
            Appointment::where('status', 'approved')-
            >orderBy('date', 'asc')->get();

            $patientCounts = [];
            foreach ($assessmentStatuses as $status) {
                $patientCounts[$status] =
                Patient::where('assessment_status', $status)-
                >count();
            }


        return view('therapistHome',
        compact('mentalDisorderData','inventoryData',
        'therapistCount', 'pendingAppointments',
        'totalQuantity', 'patientCounts',
        'approvedAppointments'));
    }


}
```

InventoryController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\Inventory;
use App\Models\Patient;
use Illuminate\Http\Request;
use App\Models\StockOut;
use App\Models\Prescription;
use Illuminate\Support\Facades\DB;
use Barryvdh\DomPDF\Facade\Pdf;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Log;
class InventoryController extends Controller
{
    public function index(Request $request)
{
    $search = $request->get('search');

    // Find all expired items
    $expiredItems = Inventory::where('expiry_date', '<',
    now())->get();

    // Loop through expired items and move each one to StockOut
    foreach ($expiredItems as $expiredItem) {
        // Create a new entry in the StockOut table with
        expired item data
        StockOut::create($expiredItem->only([
            'name', 'expiry_date', 'miligram', 'dosage',
            'quantity', 'batch_number',
        ]));

        // Delete the expired item from Inventory
        $expiredItem->delete();
    }

    // Fetch inventory items that are not expired
    $inventory = Inventory::where('name', 'like', "%
    {$search}%")
        ->where('expiry_date', '>', now())
        ->get();

    // Fetch other data for the view as needed
    $prescriptions = Prescription::with(['patient',
    'medicine'])->get();
    $stockOutItems = StockOut::all();
    $patients = DB::table('patients')
        ->select('id', DB::raw("CONCAT(firstname, ' ',
        lastname, ' ', middlename) AS full_name"))
        ->get();

    // Update stock status
    foreach ($inventory as $item) {
        if ($item->quantity == 0) {
            $item->stock_status = 'out';
        } elseif ($item->quantity <= 100) {
            $item->stock_status = 'critical';
        } elseif ($item->quantity <= 200) {
            $item->stock_status = 'warning';
        } else {
            $item->stock_status = 'normal';
        }
    }

    // Redirect to the inventory view with a success message
    if items were moved to StockOut
    $expiredCount = $expiredItems->count();
    $message = $expiredCount > 0 ? "$expiredCount␣expired
␣␣␣␣items␣moved␣to␣Stock␣Out." : null;

    return view('inventory.index', compact('inventory',
    'search', 'stockOutItems', 'patients', 'prescriptions'))
        ->with('success', $message);
}

    public function create()
    {
        return view('inventory.create');
    }

    public function store(Request $request)
    {
        $request->validate([
            'name' => 'required|string',
            'expiry_date' => 'required|date',
            'miligram' => 'required|integer',
            'dosage' => 'required|string',
            'quantity' => 'nullable|integer',  // Make
            quantity nullable if it's being set by batch_supply
            'batch_supply' => 'required|integer|min:1',  //
            Ensures batch supply is greater than 0
            'batch_number' => 'nullable|string',
            'batch_date_received' => 'nullable|date',
        ]);

        // Ensure quantity is set to batch_supply
        $data = $request->all();
        $data['quantity'] = $data['batch_supply'];  // Set
        quantity to match batch supply

        Inventory::create($data);  // Create the inventory
        record with validated data

        return redirect()->route('inventory.index')-
        >with('success', 'Medicine added successfully.');
    }


    public function edit(Inventory $inventory)
    {
        return view('inventory.edit', compact('inventory'));
    }

    public function update(Request $request, Inventory
    $inventory)
    {
        $request->validate([
            'name' => 'required|string',
            'expiry_date' => 'required|date',
            'miligram' => 'required|integer',
            'dosage' => 'required|string',
            'quantity' => 'required|integer',
            'batch_supply' => 'required|string',
            'batch_number' => 'nullable|string',
            'batch_date_received' => 'nullable|date',
        ]);

        $inventory->update($request->all());

        return redirect()->route('inventory.index')-
        >with('success', 'Medicine updated successfully.');
    }

    // public function destroy(Inventory $inventory)
    // {
```

```
    //     $inventory->delete();

    //     return redirect()->route('inventory.index')-
    >with('success', 'Medicine deleted successfully.');
    // }
    public function show(string $id)
    {
        $inventory = Inventory::findOrFail($id);
            return view('inventory.show',
            compact('inventory'));
    }

    public function moveToStockOut(Inventory $inventory)
    {
        if ($inventory->quantity == 0) {
            StockOut::create($inventory->only([
                'name', 'expiry_date', 'miligram', 'dosage',
                'quantity', 'batch_number'
            ]));

            // Delete the inventory item only, prescriptions
            remain unaffected
            $inventory->delete();

            return redirect()->route('inventory.index')-
            >with('success', 'Medicine moved to stock out.');
        }

        return redirect()->route('inventory.index')-
        >with('error', 'Medicine is not out of stock.');
    }


    public function stockOutIndex()
    {
        $stockOutItems = StockOut::all();
        return view('inventory.stockOut.index',
        compact('stockOutItems'));
    }

    public function showPrescriptionModal()
    {
        // Fetch all patients
        $patients = Patient::all();

        // Fetch all inventory items (medicines)
        $inventory = Inventory::all();

        // Pass the data to the view
        return view('inventory.prescription',
        compact('patients', 'inventory'));
    }

    public function prescribe(Request $request)
    {
        $request->validate([
            'patient_id' =>
            'required|integer|exists:patients,id',
            'medicine_id' =>
            'required|integer|exists:inventory,id',
            'dosage' => 'required|integer|min:1',
            'quantity' => 'required|integer|min:1',
        ]);

        // Find the inventory item
        $inventory = Inventory::findOrFail($request-
        >medicine_id);

        if ($inventory->quantity < $request->quantity) {
            return back()->withErrors(['quantity' =>
            'Insufficient stock for the selected medicine.']);
        }

        // Update inventory quantity
        $inventory->quantity -= $request->quantity;
        $inventory->save();

        // Log prescription
        Prescription::create([
            'patient_id' => $request->patient_id,
            'medicine_id' => $inventory->id,
            'dosage' => $request->dosage,
            'milligram' => $inventory->miligram,
            'quantity' => $request->quantity,
            'batch_number' => $inventory->batch_number,
        ]);

        return redirect()->route('inventory.index')-
        >with('success', 'Medicine prescribed successfully.');
    }

    public function generateReport()
    {
        // Fetch all inventory items
        $inventory = Inventory::all();

        // Fetch all stock out items
        $stockOutItems = StockOut::all();

        // Fetch the currently logged-in user
        $user = Auth::user();

        // Generate PDF using Blade template
        $pdf = Pdf::loadView('reports.inventory_report',
        compact('inventory', 'stockOutItems', 'user'))
            ->setPaper('a4', 'landscape'); // Set landscape
            orientation

        // Return the PDF for download
        return $pdf->download('inventory_report.pdf');
    }


}
```

InventoryReportController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class InventoryReportController extends Controller
{
    //
}
```

InvoiceController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\User;
use App\Models\Invoice;
use Illuminate\Support\Facades\Notification;
use App\Notifications\InvoicePaid;

class InvoiceController extends Controller
{
    public function index()
    {
        $invoices = Invoice::where('user_id', auth()->user()-
        >id)->get();

        return view('invoices', compact('invoices'));
    }

    public function show(Invoice $invoice)
    {
        return view('invoices', compact('invoice'));
    }

    public function sendInvoicePaidNotification(Request
    $request)
    {
        $request->validate([
            'invoice_id'=>'required|exists:invoices,id',
        ]);

        $user = auth()->user();

        $invoice = Invoice::find($request->invoice_id)-
        >first();

        $invoice['buttonText'] = 'View Invoice';
        $invoice['invoiceUrl'] = route('show.invoice');
        $invoice['thanks'] = 'Your thank you message';

        Notification::send($user, new InvoicePaid($invoice));

        return back()->with('You have successfully paid the
        invoice');
    }

    public function markInvoiceAsPaid($invoiceId)
    {
        // Assuming you have an invoice model and you're
        fetching the invoice
        $invoice = Invoice::findOrFail($invoiceId);

        // Assuming you are notifying the user associated
        with this invoice
        $user = $invoice->user; // Or fetch the user however
        you need

        // Send the notification
        $user->notify(new InvoicePaid($invoice));

        // Further logic...
```

```
␣␣␣␣}
}
```

MedicationController.php

```php
<?php

namespace App\Http\Controllers;
use App\Models\Medication;
use App\Models\Patient;
use Illuminate\Http\Request;

class MedicationController extends Controller
{
    public function index()
    {
        $medications = Medication::all();
        return view('medications.index',
        compact('medications'));
    }

    public function create()
    {
        return view('medications.create');
    }

    public function store(Request $request)
    {
        $medication = new Medication();
        $medication->name = $request->name;
        $medication->milligrams = $request->milligrams;
        $medication->dosage = $request->dosage;
        $medication->save();

        return redirect()->route('medications.index')-
        >with('success', 'Medication added successfully.');
    }

    public function edit($id)
    {
        $medication = Medication::findOrFail($id);
        return view('medications.edit',
        compact('medication'));
    }

    public function update(Request $request, $id)
    {
        $medication = Medication::findOrFail($id);
        $medication->name = $request->name;
        $medication->milligrams = $request->milligrams;
        $medication->dosage = $request->dosage;
        $medication->save();

        return redirect()->route('medications.index')-
        >with('success', 'Medication updated successfully.');
    }

    public function destroy($id)
    {
        Medication::destroy($id);
        return redirect()->route('medications.index')-
        >with('success', 'Medication␣deleted␣successfully.');
    }
}
```

NotificationController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;


class NotificationController extends Controller
{

    public function markAsRead()
{
    Auth::user()->unreadNotifications->markAsRead();
    return response()->json(['message' => 'Notifications
␣␣␣␣marked␣as␣read']);
}
}
```

PatientController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Http\Requests\StorePatientRequest;
use App\Models\Patient;
use App\Models\Inventory;
use App\Models\ArchivedPatient;
use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\DB;
use App\Models\User;
use App\Notifications\PatientActivityNotification;
use Illuminate\Support\Facades\Notification;
use App\Models\Appointment;
use Illuminate\Support\Carbon;
use App\Notifications\AppointmentReminderNotification;

class PatientController extends Controller
{

    public function dashboard()
    {
        // Get the currently authenticated user
        $user = Auth::user();

        // Ensure the user is authenticated
        if (!$user) {
            return redirect()->route('login'); // Redirect to
            login if not authenticated
        }

        // Fetch the patient record based on the user's email
        $patient = Patient::where('user_id', $user->user_id)-
        >first();

        // Check if the patient exists
        if (!$patient) {
            return redirect()->route('home')->with('error',
            'Patient not found.');
        }

         // Fetch notifications for the authenticated user
         $notifications = $user->notifications; // Or you can
         use unreadNotifications() if you want only unread ones

         // Pass the patient record to the view
         return view('home', compact('patient',
         'notifications'));
    }

    private function sendAppointmentReminderIfToday($patients)
    {
        $today = Carbon::today(); // Get today's date
        if (Carbon::parse($patients->next_appointment)-
        >isToday()) {
            // Send notifications
            $this->sendAppointmentReminder($patients);
        }
    }

    public function sendAppointmentReminder($patient)
    {
        // Notify the patient
        $patientUser = User::where('email', $patient->email)-
        >first();

        if ($patientUser && $patient->next_appointment) { //
        Check if next_appointment is not null
            $appointmentDate = \Carbon\Carbon::parse($patient-
            >next_appointment); // Convert to Carbon instance
            Notification::send($patientUser, new
            AppointmentReminderNotification(
                "Your scheduled next assessment is due today.
                Name: {$patient->firstname} {$patient-
                ->lastname}, Next-Assessment Date:
```

```
                {$appointmentDate->format('Y-m-d')}"
            ));
        }

        // Notify admin and therapist
        $usersToNotify = User::whereIn('type', [1, 2])-
        >get(); // 1 = Admin, 2 = Therapist

        foreach ($usersToNotify as $user) {
            $appointmentDate = \Carbon\Carbon::parse($patient-
            >next_appointment);
            $user->notify(new AppointmentReminderNotification(
                "A patient's next assessment is due today.
                Patient: {$patient->firstname} {$patient-
                >lastname}, Next-Assessment Date:
                {$appointmentDate->format('Y-m-d')}"
            ));
        }
    }
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        // Fetch patients along with 'barangay'
        $patients = Patient::with('inventory')
            ->where('is_active', true)
            ->orderBy('created_at', 'DESC')
            ->get();

        // Fetch barangays or define merged barangays if
        necessary
        $mergedBarangays = $patients->pluck('barangay')-
        >unique();

        // Pass the patients and barangays to the view
        return view('patients.index', compact('patients',
        'mergedBarangays'));
    }

    /**
     * Show the form for creating a new resource.
     */
    public function create()
    {
        $inventory = Inventory::all(); // Fetch all inventory
        items
        return view('patients.create', compact('inventory'));
    }


    /**
     * Store a newly created resource in storage.
     */
    public function store(StorePatientRequest $request)
    {

        $patients = Patient::create($request->validated());
        // Attach selected inventory items (medications)
        if ($request->has('inventory')) {
            $patients->inventory()->attach($request-
            >input('inventory'));
        }
            // Handle assessment fields (only if they're
            provided)
        // Update assessment and other fields
        $patients->fill($request->only([
            'assessment_status', 'impressions',
            'recommendations', 'philhealth_no', 'civil_status'
        ]))->save();

        // Check if next appointment is today
        $this->sendAppointmentReminderIfToday($patients);

        return redirect()->route('patients')->with('success',
        'Patient added successfully');
    }


    /**
     * Display the specified resource.
     */
    public function show(string $id)
    {
        $patients = Patient::findOrFail($id);

        return view('patients.show', compact('patients'));
    }

    /**
     * Show the form for editing the specified resource.
     */
    public function edit(string $id)
    {
        $patients= Patient::with('inventory')-
        >findOrFail($id);
        $inventory = Inventory::all(); // Fetch all inventory
        items
        $selectedMedications = $patients->inventory-
        >pluck('id')->toArray(); // Get selected medications
        (inventory IDs)

        return view('patients.edit', compact('patients',
        'inventory', 'selectedMedications'));
    }


    /**
```

```
     * Update the specified resource in storage.
     */
    public function update(StorePatientRequest $request,
    string $id)
    {
        $patients = Patient::findOrFail($id);

        // Get current medications (inventory IDs)
        $currentMedications = $patients->inventory-
        >pluck('id')->toArray();

        // Get updated medications from request
        $updatedMedications = $request->input('inventory') ??
        [];

        // Detect changes in medications
        $addedMedications = array_diff($updatedMedications,
        $currentMedications);
        $removedMedications = array_diff($currentMedications,
        $updatedMedications);

        // Update the existing patient record with validated
        data
        $patients->update($request->validated());

         // Handle other fields
        $patients->fill($request->only([
            'assessment_status', 'impressions',
            'recommendations', 'philhealth_no', 'civil_status'
        ]))->save();


        // Sync the medications to update the patient's
        medications list
        if ($request->has('inventory') && !empty($request-
        >input('inventory'))) {
            // Sync only if inventory input is provided and
            is not empty
            $patients->inventory()->sync($request-
            >input('inventory'));
        } else {
            // Detach all medications if inventory is empty
            $patients->inventory()->detach();
        }


        if (!empty($addedMedications) ||
        !empty($removedMedications)) {
            $medicationChanges = [
                'added' => Inventory::whereIn('id',
                $addedMedications)->pluck('name')->toArray(),
                'removed' => Inventory::whereIn('id',
                $removedMedications)->pluck('name')-
                >toArray(),
            ];

         // Notify the patient
          $patientUser = User::where('email', $patients-
          >email)->first();
          if ($patientUser) {
              Notification::send($patientUser, new
              PatientActivityNotification(
                  'Your␣medication␣records␣have␣been
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣updated.',
                  $medicationChanges
              ));
          }

          // Notify admins and therapists
          $usersToNotify = User::whereIn('type', [1, 2])-
          >get(); // 1=Admin, 2=Therapist
          foreach ($usersToNotify as $user) {
              $user->notify(new PatientActivityNotification(
                  'A␣patients␣medication␣records␣have␣been
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣updated.',
                  $medicationChanges
              ));
          }
      }
          // Check if next appointment is today
         $this->sendAppointmentReminderIfToday($patients);
```

```


        return redirect()->route('patients')->with('success',
        'Patient␣updated␣successfully');
    }

    /**
     * Remove the specified resource from storage.
     */


    public function archive($id)
    {
        $patients = Patient::findOrFail($id);
        // Retrieve medications associated with the patient
        from the pivot table
        // Retrieve the IDs of the medications associated
        with the patient
        $medications = $patients->inventory->pluck('id')-
        >toArray(); // Adjust if you need specific fields


        // Move patient to the archived_patients table
        ArchivedPatient::create([
            'first_appointment' => $patients-
            >first_appointment,
            'firstname' => $patients->firstname,
            'lastname' => $patients->lastname,
            'middlename' => $patients->middlename,
            'gender' => $patients->gender,
            'barangay' => $patients->barangay,
            'birthdate' => $patients->birthdate,
            'mental_disorder_type' => $patients-
           >mental_disorder_type,
            'disability' => $patients->disability,
            'next_appointment' => $patients->next_appointment,
            'medications' => json_encode($medications), //
            Using previously defined $medications as IDs
            'phoneNo' => $patients->phoneNo,

            'assessment_status' =>  $patients-
           >assessment_status,
            'impressions' =>  $patients->impressions,
            'recommendations' => $patients->recommendations,
            'philhealth_no' =>  $patients->philhealth_no,
            'civil_status' =>  $patients->civil_status,
        ]);

        // Update the patient status to inactive

        $patients->update(['is_active' => false]);
            // Delete the patient from the original table
        $patients->delete();


        // Redirect with success message
        return redirect()->route('patients')->with('success',
        'Patient␣archived␣successfully.');

    }
    // Method to display archived patients
    public function archiveIndex()
    {
        $archivedPatients = ArchivedPatient::all();
        return view('patients.archive',
        compact('archivedPatients'));
    }


    public function restore($id)
    {
        // Find archived patient
        $archivedPatient = ArchivedPatient::findOrFail($id);

        // Ensure the patient exists
        if (!$archivedPatient) {
            return redirect()->route('patients.archived')-
            >with('error', 'Patient␣not␣found.');
        }

        // Decode medications from JSON
        $medications = json_decode($archivedPatient-
        >medications, true);

        if (json_last_error() !== JSON_ERROR_NONE) {
            Log::error('JSON Decode Error: ' .
            json_last_error_msg());
            $medications = []; // Reset to an empty array if
            there's an error
        }

        // Filter out any non-numeric values from decoded
        medications
        $medicationsToAttach = array_filter($medications,
        'is_numeric');

        // Fetch valid inventory IDs from the database that
        match those in $medicationsToAttach
        $validInventoryIds =
        \App\Models\Inventory::whereIn('id',
        $medicationsToAttach)->pluck('id')->toArray();

        // Begin transaction
        DB::transaction(function () use ($archivedPatient,
        $validInventoryIds, &$restoredPatient) {
            // Restore the patient data back to the original
            table
            $restoredPatient = Patient::create([
                'first_appointment' => $archivedPatient-
                >first_appointment,
                'firstname' => $archivedPatient->firstname,
                'lastname' => $archivedPatient->lastname,
                'middlename' => $archivedPatient->middlename,
                'gender' => $archivedPatient->gender,
                'barangay' => $archivedPatient->barangay,
                'birthdate' => $archivedPatient->birthdate,
                'mental_disorder_type' => $archivedPatient-
                >mental_disorder_type,
                'phoneNo' => $archivedPatient->phoneNo,
                'next_appointment' => $archivedPatient-
                >next_appointment,
                'is_active' => true, // Mark the patient as
                active

                'assessment_status' => $archivedPatient-
                >assessment_status,
                'impressions' => $archivedPatient-
                >impressions,
                'recommendations' => $archivedPatient-
                >recommendations,
                'philhealth_no' =>  $archivedPatient-
                >philhealth_no,
                'civil_status' => $archivedPatient-
                >civil_status,
            ]);

            // Attach medications only if valid IDs are found
            if (!empty($validInventoryIds)) {
                Log::info('Medications to attach:',
                $validInventoryIds); // Log valid IDs
                $restoredPatient->inventory()-
                >attach($validInventoryIds);
            } else {
                Log::warning('No valid medication IDs to
                attach for restored patient.');
            }

            // Remove the patient from the archived table
            after successful restoration
            $archivedPatient->delete();
        });

        return redirect()->route('patients')->with('success',
        'Patient restored successfully.');
    }


    public function markAsRead($id)
    {
        $user = Auth::user();

        if (!$user) {
            return redirect()->route('login')->with('error',
            'Please log in to mark notifications as read.');
        }

        $notification = $user->unreadNotifications->find($id);


        Log::info('Before marking as read:', $notification-
        >toArray());

        $notification->markAsRead();

        Log::info('After marking as read:', $user-
        >unreadNotifications->toArray());

        return redirect()->route('notifications')-
        >with('success', 'Notification marked as read.');
    }
}
```

PatientReportController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\Patient;
use App\Models\User;
use Barryvdh\DomPDF\Facade\Pdf; // Optional, if you want to
generate PDFs without a library
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\View;

class PatientReportController extends Controller
{
    //

    public function showReportPage()
    {
        // Fetch distinct barangays from the database
        $barangaysFromDatabase = Patient::distinct()-
        >pluck('barangay')->toArray();

        // Predefined list of barangays
        $barangays = [
            'Abad␣Santos', 'Aguinaldo', 'Aquino', 'Bagacay',
            'Barangay␣Central', 'Beguin', 'Calomagon',
            'Cadandanan', 'Caruhayon', 'Danao', 'Del␣Pilar',
            'Fabrica', 'Gate', 'Jamorawon',
            'Lapinig', 'Managa-naga', 'Nasuje', 'Obrero',
            'Otavi', 'Quezon', 'San␣Francisco',
            'San␣Isidro', 'San␣Jose␣(Panumbagan)', 'San
␣␣␣␣␣␣␣␣␣␣␣␣Rafael', 'Santa␣Remedios', 'Taromata',
            'Zone␣I', 'Zone␣II', 'Zone␣III', 'Zone␣IV', 'Zone
␣␣␣␣␣␣␣␣␣␣␣␣V', 'Zone␣VI', 'Zone␣VII', 'Zone␣VIII'
        ];

        // Merge the database barangays with the predefined
        ones, making sure there are no duplicates
        $mergedBarangays =
        array_unique(array_merge($barangays,
        $barangaysFromDatabase));

        return view('patients.index',
        compact('mergedBarangays'));
    }



    public function generateReport(Request $request)
    {
           $barangay = $request->input('barangay');
    $sortByName = $request->has('sort_by_name') && $request-
    >input('sort_by_name') == 'on';

    $patients = Patient::with('inventory');

    // Apply barangay filter if selected
    if ($barangay) {
        $patients = $patients->where('barangay', $barangay);
    }

    // Apply sorting by name if selected
    if ($sortByName) {
        $patients = $patients->orderBy('firstname')-
        >orderBy('lastname');
    }

    $patients = $patients->get();


        $mentalDisorderTypes = [
            'Psychosis' ,
            'Depression' ,
            'Epilepsy',
            'Developmental␣Disorder',
            'Behavioral␣Disorder',
            'Dementia',
            'Bipolar',
            'Alcohol␣Use␣Disorder',
            'Drug␣Use␣Disorder',
            'Self␣Harm',
        ];

        $user = Auth::user();

        // Load view as HTML and generate PDF
        $pdf = Pdf::loadView('reports.patient_report',
        compact('patients', 'user', 'mentalDisorderTypes'))
            ->setPaper('legal', 'landscape'); // Set to A4
             landscape

        // Stream or download the generated PDF
        return $pdf->download('patient_report.pdf');
    }

}
```

ProfileController.php

```
<?php

namespace App\Http\Controllers;

use App\Http\Requests\ProfileUpdateRequest;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Redirect;
use Illuminate\View\View;

class ProfileController extends Controller
{
    /**
     * Display the user's profile form.
     */
    public function edit(Request $request): View
    {
          // Fetch the currently authenticated user's profile
            data
        $users = Auth::user();
        $users = $request->user();
        return view('profile.edit', [
            'users' => $request->user(),
        ], compact('users'));
    }

    /**
     * Update the user's profile information.
     */
    public function update(ProfileUpdateRequest $request):
    RedirectResponse
    {
        $user = Auth::user();

        // Fill in other profile fields
        $user->fill($request->except('password'));

        if ($request->filled('password')) {
            // Hash the new password and update
            $user->password = bcrypt($request->password);
        }

        // Check if the email has changed, if so, reset email
        verification
        if ($user->isDirty('email')) {
            $user->email_verified_at = null;
        }

        $user->save();

        return Redirect::route('profile.edit')-
        >with('status', 'profile-updated');
    }



    /**
     * Delete the user's account.
     */
    public function destroy(Request $request):
    RedirectResponse
    {
        $request->validateWithBag('userDeletion', [
            'password' => ['required', 'current_password'],
        ]);

        $user = $request->user();

```

```
            Auth::logout();

            $user->delete();

            $request->session()->invalidate();
            $request->session()->regenerateToken();

            return Redirect::to('/');
        }
}
```

TherapistController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Therapist;
use App\Models\Services;
use App\Http\Requests\StoreTherapistRequest;
use App\Http\Requests\UpdateTherapistRequest;
use App\Models\User;
use Illuminate\Support\Facades\DB;
class TherapistController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        $therapists = Therapist::orderBy('created_at',
        'DESC')->get();
            return view('therapists.index',
            compact('therapists'));
    }

    /**
     * Show the form for creating a new resource.
     */
    public function create()
    {

        return view('therapists.create');
    }

    /**
     * Store a newly created resource in storage.
    */
    public function store(StoreTherapistRequest $request)
    {
         // Extract specializations
        $specializations = $request->input('specializations',
        []);

        // Check for "Other" input
        if ($request->has('specializations.other') &&
        !empty($request->input('specializations.other'))) {
            $specializations[] = $request-
            >input('specializations.other'); // Add custom
            specialization
        }

        // Validate and store the therapist details
        Therapist::create([
            'firstname' => $request->input('firstname'),
            'lastname' => $request->input('lastname'),
            'middlename' => $request->input('middlename'),
            'email' => $request->input('email'),
            'phoneNo' => $request->input('phoneNo'),
            'place_of_assignment' => $request-
            >input('place_of_assignment'),
            'specializations' =>
            json_encode($specializations), // Store as JSON


        ]);

            return redirect()->route('therapists')-
            >with('success', 'Therapist␣created
␣␣␣␣␣␣␣␣␣␣␣␣successfully.');
    }


    /**
     * Display the specified resource.
     */
    public function show($id){
        $therapists = Therapist::findOrFail($id);
            return view('therapists.show',
            compact('therapists'));
    }


    /**
     * Show the form for editing the specified resource.
     */
    public function edit(string $id)
    {
        $therapists = Therapist::findOrFail($id);
            return view('therapists.edit',
            compact('therapists'));
    }

    /**
     * Update the specified resource in storage.
     */
   // TherapistController.php

```

```
    public function update(UpdateTherapistRequest $request,
    string $id)
    {
        $therapist = Therapist::findOrFail($id);

        // Extract specializations
         $specializations = $request->input('specializations',
         []);

         // Check for ”Other” input
         if ($request->has('specializations.other') &&
         !empty($request->input('specializations.other'))) {
             $specializations[] = $request-
             >input('specializations.other'); // Add custom
             specialization
         }

        $therapist->update([
            'firstname' => $request->input('firstname'),
            'lastname' => $request->input('lastname'),
            'middlename' => $request->input('middlename'),
            'email' => $request->input('email'),
            'phoneNo' => $request->input('phoneNo'),
            'place_of_assignment' => $request-
           >input('place_of_assignment'),
            'specializations' =>
            json_encode($specializations), // Update
            specializations as JSON

        ]);

         return redirect()->route('therapists')-
         >with('success', 'Therapist␣updated␣successfully.');
    }


     /**
      * Remove the specified resource from storage.
      */
     public function destroy(string $id){

         $therapists = Therapist::findOrFail($id);
         $therapists->delete();
             return redirect()->route('therapists')-
             >with('success', 'Therapist␣deleted
  ␣␣␣␣␣␣␣␣␣␣␣␣successfully');
     }

     // Show all archived therapists
     public function archived()
     {
     // Fetch only soft-deleted (archived) therapists
         $archivedTherapists = Therapist::onlyTrashed()-
         >orderBy('deleted_at', 'DESC')->get();

         return view('therapists.archived',
         compact('archivedTherapists'));
     }

     // Restore a specific therapist
     public function restore($id)
     {
         $therapist = Therapist::onlyTrashed()-
         >findOrFail($id);
         $therapist->restore();

         return redirect()->route('therapists')-
         >with('success', 'Therapist␣restored␣successfully.');

     }

     // Restore all archived therapists
     public function restoreAll()
     {
         Therapist::onlyTrashed()->restore();

         return redirect()->route('therapists.archived')-
         >with('success', 'All␣therapists␣restored
         ␣␣␣␣␣␣␣␣successfully');
     }

```

```
}
```

UserController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\User;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Auth\Events\Registered;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Mail;
use App\Models\ArchivedUser;
use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\Validator;
use Illuminate\Validation\Rule;
use App\Notifications\UserActivityNotification;
use Illuminate\Support\Facades\Auth;

class UserController extends Controller
{
    public function register(Request $request)
    {
        // Map integer types to strings if necessary
        $typeMapping = [0 => 'patient', 1 => 'admin', 2 =>
        'therapist'];
        $request->merge([
            'type' => $typeMapping[$request->input('type')]
            ?? $request->input('type')
        ]);

        $request->validate([
            'username' =>
            'required|string|max:255|unique:users',
            'name' => 'required|string|max:255',
            'email' => 'required|email|max:255|unique:users',
            'password' => 'required|string|min:8|confirmed',
            'type' => ['required', Rule::in(['patient',
            'admin', 'therapist'])],
        ]);

        $user = User::create([
            'username' => $request->username,
            'name' => $request->name,
            'email' => $request->email,
            'password' => bcrypt($request->password),
            'type' => $request->type,
        ]);

        event(new Registered($user));

        return redirect()->route('login')->with('message',
        'Verification␣link␣sent␣to␣your␣email!');
```

```
    }

    public function index()
    {
        $users = User::all();
        return view ('users.index', compact('users'));
    }

    public function edit(string $id)
    {
        $users = User::findOrFail($id);

            return view('users.edit', compact('users'));
    }

    public function update(Request $request, $id)
    {
        Log::info('Updating␣User␣ID:␣' . $id);
        Log::info('Request␣Data:␣', $request->all());

        // Validate the input
        $request->validate([
            'username' =>
            'required|string|max:255|unique:users',
            'name' => 'required|string|max:255',
            'email' =>
            'required|email|max:255|unique:users,email,' .
            $id,
            'type' => ['required', Rule::in(['patient',
            'admin', 'therapist'])],
        ]);

        // Find the user
        $user = User::findOrFail($id);

        // Update user details
        $user->username = $request->input('username');
        $user->name = $request->input('name');
        $user->email = $request->input('email');
        $user->type = $request->input('type');

        // Save changes
        if ($user->save()) {
            Log::info('User␣updated␣successfully:', $user-
            >toArray());
            return redirect()->route('users')-
            >with('success', 'User␣updated␣successfully');
        } else {
            Log::error('Failed␣to␣update␣user.');
            return redirect()->back()->withErrors(['error' =>
            'Failed␣to␣update␣user.'])->withInput();
        }
    }

    public function archivedIndex()
    {
        // Fetch all archived users with necessary attributes
        $archivedUsers = ArchivedUser::all(); // Ensure this
        includes the `type` field

        return view('users.archive',
        compact('archivedUsers'));
    }

    public function archive(Request $request, $userId)
    {
        // Find the user to be archived
        $user = User::findOrFail($userId);

        // Log user type for debugging
        Log::info("Archiving user with ID: {$user->id}, Name:
        {$user->name}, Type: {$user->type}");

        // Create a new ArchivedUser instance
        $archivedUser = new ArchivedUser();
        $archivedUser->name = $user->name;
        $archivedUser->email = $user->email;

        // Use switch-case to set the type
        switch ($user->type) {
            case 'patient':
                $archivedUser->type = 0; // Patient
                break;
            case 'admin':
                $archivedUser->type = 1; // Admin
                break;
            case 'therapist':
                $archivedUser->type = 2; // Therapist
                break;
            default:
                $archivedUser->type = 0; // Default to
                patient or any default case
                break;
        }

        $archivedUser->archived_at = now();

        // Save the archived user
        $archivedUser->save();

        // Optionally delete the original user
        $user->delete();

        return redirect()->route('users')->with('success',
        'User archived successfully.');
    }


    public function restore($id)
    {
        // Find the archived user
        $archivedUser = ArchivedUser::findOrFail($id);

        // Check if a user with the same email already exists
        if (User::where('email', $archivedUser->email)-
        >exists()) {
            return redirect()->route('archived.users.index')-
            >with('error', 'User with the same email already
            exists. Restore failed.');
        }

        // Create a new active user based on archived user
        data
        $user = new User();
        $user->name = $archivedUser->name;
        $user->email = $archivedUser->email;

        // Use switch-case to map archived user type back to
        string
        switch ($archivedUser->type) {
            case 0:
                $user->type = 'patient'; // Patient
                break;
            case 1:
                $user->type = 'admin'; // Admin
                break;
            case 2:
                $user->type = 'therapist'; // Therapist
                break;
            default:
                $user->type = 'patient'; // Default case
                break;
        }

        // Ensure to set a password or handle it differently
        $user->password = bcrypt('defaultPassword');

        // Save the new user
        $user->save();

        // Delete the archived user record
        $archivedUser->delete();

        return redirect()->route('archived.users.index')-
        >with('success', 'User restored successfully.');
    }



}
```

login.blade.php

```
<!DOCTYPE html>
<html lang=”en”>
<head>
    <meta charset=”utf–8”>
    <meta http–equiv=”X–UA–Compatible” content=”IE=edge”>
    <meta name=”viewport” content=”width=device–width,
␣␣␣␣initial–scale=1,␣shrink–to–fit=no”>
    <meta name=”description” content=””>
    <meta name=”author” content=””>
    <title>Login</title>
    <!–– Favicon (Logo) ––>
    <link rel=”icon” href=”{{␣asset(’img/favicon.png’)␣}}”
    type=”image/x–icon”>
    <!–– Custom fonts for this template––>

    <link href=”https://cdnjs.cloudflare.com/ajax/libs/font–
␣␣␣␣awesome/6.0.0–beta3/css/all.min.css” rel=”stylesheet”
    type=”text/css”>

    <link href=”https://fonts.googleapis.com/css?
␣␣␣␣family=Nunito:200,200i,300,300i,400,400i,600,600i,700,700i
␣␣␣␣,800,800i,900,900i” rel=”stylesheet”>
    <!–– Custom styles for this template––>
    <link href=”{{␣asset(’sd–admin/css/sb–admin–2.min.css’)
␣␣␣␣}}” rel=”stylesheet”>
    <style>
```

```
        .bg-gradient-green {
            background: linear-gradient(to right, #42b883,
            #347474, #35495e);
        }
        .login-card {
            max-width: 400px;
            margin: auto;
        }


    </style>
</head>
<body class="bg-gradient-green">
  <div class="container">
    <!-- Outer Row -->
    <div class="row␣justify-content-center␣align-items-center
␣␣␣␣min-vh-100">
      <div class="col-xl-5␣col-lg-6␣col-md-8␣login-card">
        <div class="card␣o-hidden␣border-0␣shadow-lg␣my-5">
          <div class="card-body␣p-4">
            <!-- Card Body Content -->
            <div class="text-center␣mb-4">
              <img src="{{
␣␣␣␣␣␣␣␣␣␣␣␣␣␣asset('img/369166592_1042691620113380_1198365635
␣␣␣␣␣␣␣␣␣␣␣␣␣␣148980317_n_034718.png')␣}}" class="mb-4"
              style="width:␣100px;␣height:␣auto;"
              alt="Mindcare">
  <h1 class="h4 text-gray-900">Welcome to
  Mindcare</h1>
</div>
<form action="{{ route('login') }}" method="POST"
class="user">
  @csrf
  @if ($errors->any())
    <div class="alert alert-danger">
      <ul>
        @foreach ($errors->all() as $error)
          <li>{{ $error }}</li>
        @endforeach
      </ul>
    </div>
  @endif
  <div class="form-group">
    <input name="username" type="text"
    class="form-control form-control-user"
    placeholder="Enter Username " required>
    </div>

  <div class="form-group">
    <input name="password" type="password"
    class="form-control form-control-user"
    placeholder="Password" required>

  </div>
```

```

            <div class=”text-right mb-3”>
                <a class=”small” href=”{{
                route(’password.request’) }}”>Forgot Password?
                </a>
            </div>

              <div class=”form-group”>
                <div class=”custom-control custom-checkbox
                small”>
                  <input name=”remember” type=”checkbox”
                  class=”custom-control-input”
                  id=”customCheck”>
                  <label class=”custom-control-label”
                  for=”customCheck”>Remember Me</label>
                </div>
              </div>
              <button type=”submit” class=”btn btn-success
              btn-block btn-user”>Login</button>
            </form>
            <hr>
            <div class=”text-center”>
              <a class=”small” href=”{{ route(’register’)
              }}”>Create an Account!</a>
            </div>
          </div>
        </div>
```

```
            </div>
        </div>
    </div>
    <!-- Bootstrap core JavaScript-->
    <script src=”{{ asset(’sd-
  admin/vendor/jquery/jquery.min.js’) }}”></script>
    <script src=”{{ asset(’sd-
  admin/vendor/bootstrap/js/bootstrap.bundle.min.js’) }}”>
    </script>
    <!-- Core plugin JavaScript-->
    <script src=”{{ asset(’sd-admin/vendor/jquery-
  easing/jquery.easing.min.js’) }}”></script>
    <!-- Custom scripts for all pages-->
    <script src=”{{ asset(’sd-admin/js/sb-admin-2.min.js’) }}”>
    </script>

</body>
</html>
```

register.blade.php

```
<!DOCTYPE html>
<html lang=”en”>

<head>
    <meta charset=”utf-8”>
    <meta http-equiv=”X-UA-Compatible” content=”IE=edge”>
    <meta name=”viewport” content=”width=device-width, initial-
```

```
␣␣scale=1,␣shrink-to-fit=no">
  <meta name="description" content="">
  <meta name="author" content="">
  <title>Register</title>

    <!-- Favicon (Logo) -->
    <link rel="icon" href="{{␣asset('img/favicon.png')␣}}"
    type="image/x-icon">
  <!-- Custom fonts for this template-->
  <link href="{{␣asset('sd-admin/vendor/fontawesome-
␣␣free/css/all.min.css')␣}}" rel="stylesheet" type="text/css">
  <link href="https://fonts.googleapis.com/css?
␣␣family=Nunito:200,200i,300,300i,400,400i,600,600i,700,700i,8
␣␣00,800i,900,900i" rel="stylesheet">
  <!-- Custom styles for this template-->
  <link href="{{␣asset('sd-admin/css/sb-admin-2.min.css')␣}}"
   rel="stylesheet">
  <style>
    .bg-gradient-success {
      background: linear-gradient(to right, #42b883, #347474,
      #35495e);
    }
    .card {
      max-width: 500px; /* Adjusts the card width */
      margin: auto;
    }
  </style>
```

```
</head>

<body class="bg-gradient-success">
  <div class="container d-flex align-items-center min-vh-100">
    <div class="card o-hidden border-0 shadow-lg">
      <div class="card-body p-0">
        <!-- Nested Row within Card Body -->
        <div class="row">
          <div class="col-lg-12">
            <div class="p-5">
              <div class="text-center">
                <h1 class="h4 text-gray-900 mb-4">Create an
                Account!</h1>
              </div>
              <form action="{{ route('register') }}"
              method="POST" class="user">
                @csrf
                <div class="form-group">
                    <input name="username" type="text"
                    class="form-control form-control-user
                    @error('username')is-invalid @enderror"
                    id="username" placeholder="Username">
                    @error('username')
                    <span class="invalid-feedback">{{
                    $message }}</span>
                    @enderror
                </div>
```

```
                 <div class="form-group">

                  <input name="name" type="text" class="form-
                  control form-control-user @error('name')is-
                  invalid @enderror" id="name"
                  placeholder="name">
                  @error('name')
                  <span class="invalid-feedback">{{ $message
                  }}</span>
                  @enderror
                </div>
                <div class="form-group">

                  <input name="email" type="email"
                  class="form-control form-control-user
                  @error('email')is-invalid @enderror"
                  id="email" placeholder="example@gamil.com">
                  @error('email')
                  <span class="invalid-feedback">{{ $message
                  }}</span>
                  @enderror
                </div>
                <div class="form-group row">
                  <div class="col-sm-6 mb-3 mb-sm-0">

                    <input name="password" type="password"
                    class="form-control form-control-user
```

```
                            @error('password') is-invalid @enderror"
                            id="password" placeholder="Password">
                            @error('password')
                            <span class="invalid-feedback">{{
                            $message }}</span>
                            @enderror
                        </div>
                        <div class="col-sm-6">

                            <input name="password_confirmation"
                            type="password" class="form-control form-
                            control-user
                            @error('password_confirmation') is-invalid
                            @enderror" id="password"
                            placeholder="Repeat Password">
                            @error('password_confirmation')
                            <span class="invalid-feedback">{{
                            $message }}</span>
                            @enderror
                        </div>
                    </div>
                    <button type="submit" class="btn btn-success
                    btn-user btn-block">Register Account</button>
                </form>
                <hr>
                <div class="text-center">
                    <a class="small" href="{{ route('login')
                }}">Already have an account? Login!</a>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
  <!-- Bootstrap core JavaScript-->
  <script src="{{ asset('sd-
  admin/vendor/jquery/jquery.min.js') }}"></script>
  <script src="{{ asset('sd-
  admin/vendor/bootstrap/js/bootstrap.bundle.min.js') }}">
  </script>
  <!-- Core plugin JavaScript-->
  <script src="{{ asset('sd-admin/vendor/jquery-
  easing/jquery.easing.min.js') }}"></script>
  <!-- Custom scripts for all pages-->
  <script src="{{ asset('sd-admin/js/sb-admin-2.min.js') }}">
  </script>
</body>

</html>
```

allBooked.blade.php

```
@extends('layouts.app')
@section('content')
```

```

<div class="container-fluid p-4 text-sm max-w-7xl min-w-3">
    <div class="card shadow">
        <div class="card-header" style="background-color:
        #35495e;">
            <h6 class="text-white font-weight-
            bold">Appointments Record List</h6>
        </div>

        <div class="card-body">
            <div class="table-responsive">
                <div class="flex align-items-center py-2">
                    <a href="{{ route('calendar.index') }}"
                    class="text-primary text-decoration py-2
                    text-sm">
                        Go to Calendar
                        <svg
                        xmlns="http://www.w3.org/2000/svg"
                        width="16" height="16"
                        fill="currentColor" class="bi bi-
                        arrow-right-circle-fill" viewBox="0 0
                        16 16">
                            <path d="M8 0a8 8 0 1 1 0 16A8 8
                            0 0 1 8 0M4.5 7.5a.5.5 0 0 0 0
                            1h5.793l-2.147 2.146a.5.5 0 0 0
                            .708.708l3-3a.5.5 0 0 0 0-.708l-3-
                            3a.5.5 0 1 0-.708.708L10.293
                            7.5z”/>
                        </svg>
                    </a>
                </div>
                <hr />
                <br />

                <table class=”table table-bordered”
                id=”AppointmentsListTable” width=”100%”
                cellspacing=”0”>
                    <thead class=”table-success text-gray-
                    800”>
                        <tr>
                            <th>ID</th>
                            <th>Name</th>
                            <th>Service</th>
                            <th>Comments</th>
                            <th>Date</th>
                            <th>Time</th>
                            <th>Status</th>
                            <th>Actions</th>
                        </tr>
                    </thead>
                    <tbody>
                        @if($appointments->isEmpty())
                            <tr>
                                <td colspan=”8” class=”text-
                                center">No appointments
                                found</td>
                            </tr>
                        @else
                            @foreach($appointments as
                            $appointment)
                                <tr>
                                    <td class="align-middle">
                                    {{ $appointment->id }}
                                    </td>
                                    <td class="align-middle">
                                    {{ $appointment->name }}
                                    </td>
                                    <td class="align-middle">
                                    {{ $appointment->services
                                    ? $appointment->services-
                                    >name : 'No Services
                                    found' }}</td>
                                    <td class="align-middle">
                                    {{ $appointment->comments
                                    }}</td>
                                    <td class="align-middle">
                                    {{ $appointment->date }}
                                    </td>
                                    <td class="align-middle">
                                    {{ $appointment->time }}
                                    </td>
```

```
<td class="align-middle">
<span class="badge {{
        $appointment-
        >status ===
        'approved' ? 'bg-
        success' :
        ($appointment-
        >status ===
        'pending' ? 'bg-
        warning' : 'bg-
        danger')
    }}">
    {{
    ucfirst($appointment-
    >status) }}
</span>
</td>
<td class="align-middle">
<!-- Approve button -->
<!-- Therapist/Admin
Approve Button -->
@if(auth()->user()->role
!== 'patient' &&
$appointment->status ===
'pending')
        <form
        method="POST"
```

```
                  action="{{
                  route('appointment
                  s.approve',
                  $appointment->id)
                  }}"
                  style="display:inl
                  ine;">
                     @csrf

                     @method('PATCH
                     ')
                     <button
                     type="submit"
                     class="btn
                     btn-sm btn-
                     primary">Appro
                     ve</button>
                  </form>
            @elseif($appointment-
            >status === 'approved')
               <!-- Cancel button -->
               <form method="POST"
               action="{{
               route('appointments.ca
               ncel', $appointment-
               >id) }}"
               style="display:inline;
```

```
                                        ">
                                                @csrf
                                                @method('PATCH')
                                                <button
                                                type="submit"
                                                class="btn btn-sm
                                                btn-
                                                danger">Cancel</bu
                                                tton>
                                            </form>
                                        @endif

                                        @if(auth()->user()->role
                                        === 'patient' &&
                                        $appointment->status ===
                                        'approved')
                                            <form method="POST"
                                            action="{{
                                            route('appointments.ca
                                            ncel', $appointment-
                                            >id) }}"
                                            style="display:inline;
                                            ">
                                                @csrf
                                                @method('PATCH')
                                                <button
                                                type="submit"
                                                class="btn btn-sm
                                                btn-
                                                danger">Cancel</bu
                                                tton>
                                            </form>
                                        @endif

                                        <!-- Status button if
                                        neither pending nor
                                        approved -->
                                        @if($appointment->status
                                        !== 'pending' &&
                                        $appointment->status !==
                                        'approved')
                                            <button class="btn
                                            btn-sm btn-secondary"
                                            disabled>{{
                                            ucfirst($appointment-
                                            >status) }}</button>
                                        @endif
                                    </td>


                                    </tr>
                                @endforeach
                            @endif
                        </tbody>
```

```
                </table>
            </div>
        </div>
    </div>
</div>

<script src=”https://code.jquery.com/jquery−3.5.1.min.js”>
</script>
<link rel=”stylesheet”
 href=”https://cdn.datatables.net/1.13.5/css/jquery.dataTables.
min.css”>
<script
 src=”https://cdn.datatables.net/1.13.5/js/jquery.dataTables.mi
n.js”></script>
<!−− JavaScript for Calendar and Approval Handling −−>
<script>
    document.addEventListener(’DOMContentLoaded’, function() {
        var calendarEl = document.getElementById(’calendar’);

        var calendar = new FullCalendar.Calendar(calendarEl, {
            initialView: ’dayGridMonth’,
            dateClick: function(info) {
                // Open the modal and set the date in the form
                    document.getElementById(’appointmentDate’)
                    .value = info.dateStr;
                    var bookingModal = new
                    bootstrap.Modal(document.getElementById(’b
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ookingModal'));
                    bookingModal.show();

                     // Load available time slots for the
                     selected date
                loadTimeSlots(info.dateStr);
            },
        events: [
            @foreach($appointments as $appointment)
                {
                    title: '{{␣$appointment->name␣}}␣({{
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ucfirst($appointment->status)␣}})',
                    start: '{{␣$appointment->date␣}}',
                    backgroundColor: '{{␣$appointment->status
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣===␣'approved'␣?␣'green'␣:␣'red'␣}}',
                    borderColor: '{{␣$appointment->status␣===
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣'approved'␣?␣'green'␣:␣'red'␣}}'
                },
            @endforeach
        ],


    });

    calendar.render();

    // Function to reload calendar events after approval
    function reloadCalendarEvents() {
        calendar.refetchEvents();
    }
});

function loadTimeSlots(date) {
        // Clear previous time slots
        document.getElementById('timeSlots').innerHTML = '';

        // Fetch available time slots for the selected date
        fetch(`/available-slots/${date}`)
            .then(response => response.json())
            .then(slots => {
                slots.forEach(slot => {
                    let button =
                    document.createElement('button');
                    button.className = slot.booked ? 'btn␣btn-
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣danger' : 'btn␣btn-success';
                    button.disabled = slot.booked;
                    button.textContent = slot.booked ?
                    'Booked' : slot.time;
                    button.type = 'button';

                    if (!slot.booked) {
                        button.addEventListener('click',
                        function() {
                            document.getElementById('selectedT
                            ime').value = slot.time;
                            console.log('Selected time:',
                            slot.time); // Log the selected
                            time for debugging

                        });
                    }

                    document.getElementById('timeSlots').appen
                    dChild(button);
                });
            });
            function loadTimeSlots(date) {
        // Clear previous time slots
        document.getElementById('timeSlots').innerHTML = '';

        // Fetch available time slots for the selected date
        fetch(`/available-slots/${date}`)
            .then(response => response.json())
            .then(slots => {
                slots.forEach(slot => {
                    let button =
                    document.createElement('button');
                    button.className = slot.booked ? 'btn btn-
                    danger' : 'btn btn-primary';
                    button.disabled = slot.booked;
                    button.textContent = slot.booked ?
                    'Booked' : slot.time;
                    button.type = 'button';

                    if (!slot.booked) {
                    button.addEventListener('click',
                        function() {
                            document.getElementById('selectedT
                            ime').value = slot.time;
                            console.log('Selected time:',
                            slot.time); // Log the selected
                            time for debugging

                        });
                    }

                    document.getElementById('timeSlots').appen
                    dChild(button);
                });
            });
        }

        function cancelAppointment(id) {
        // Make an AJAX call to cancel the appointment
        fetch(`/appointments/cancel/${id}`, {
            method: 'POST', // Assuming you use POST for
            canceling
            headers: {
                'Content-Type': 'application/json',
                'X-CSRF-TOKEN':
                document.querySelector('meta[name="csrf-
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣token"]').getAttribute('content')
            },
            body: JSON.stringify({ id: id })
        })
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                // After canceling, reload the calendar
                events to reflect the changes
                reloadCalendarEvents();
            } else {
                alert('Failed␣to␣cancel␣the␣appointment');
            }
        });
    }

    // Call this function after your cancel button click
    handler
    cancelAppointment(appointmentId);

    function reloadCalendarEvents() {
    calendar.refetchEvents();
    }
}
```

```


</script>

<script>
    $(document).ready(function() {
        $('#AppointmentsListTable').DataTable({
            "order": [],
            "paging": true,
            "searching": true,
            "info": true,
            "lengthChange": true
        });
    });
</script>
@endsection
```

index.blade.php

```
@extends('layouts.app')
    @section('content')
        <!-- Begin Page Content -->
        <div class="container-fluid p-4 text-sm sm:px-4  ">
            <!-- DataTables -->
            <div class="card shadow mb-4">
                <div class="card-header" style="background-
                color: #35495e;">
                    <h6 class="m-0 text-white text-md font-
```

```
                   weight-bold”>Appointment Booking
                   Calendar</h6>
               </div>
               <div id=”calendar” class=”m-3 text-lg”
               style=”color: black;”> </div>


               <div class=”card-body”>
                   <!-- Modal for booking form -->
    <div class=”modal fade” id=”bookingModal” tabindex=”-1”
    aria-labelledby=”bookingModalLabel” aria-hidden=”true”>
        <div class=”modal-dialog”>
            <div class=”modal-content”>
               <div class=”modal-header” style=”background-
               color: #35495e;”>
                   <h5 class=”modal-title text-white”
                   id=”bookingModalLabel”>Book
                   Appointment</h5>
                   <button type=”button” class=”btn-close ”
                   data-bs-dismiss=”modal” aria-
                   label=”Close”></button>
               </div>
               <div class=”modal-body”>
                   <form id=”bookingForm” method=”POST”
                   action=”{{ route('appointments.store')
                   }}”>
                       @csrf
```

```
<input type=”hidden” name=”date”
id=”appointmentDate”>
<input type=”hidden”
id=”selectedTime” name=”time”
required>

<div class=”mb-3”>
    <label for=”name” class=”form-
    label”>Name</label>
    <input type=”text” class=”form-
    control” id=”name” name=”name”
    required>
</div>

<div class=”mb-3”>
    <label for=”services” class=”form-
    label”>Services Needed</label>
    <select name=”services_id”
    id=”service” class=”form-control”
    required>
        @foreach($services as
        $service)
            <option value=”{{
            $service->id }}”>{{
            $service->name }}</option>
        @endforeach
```

```
                            </select>
                        </div>

                        <div class=”mb–3”>
                            <label for=”comments” class=”form–
                            label”>Comments</label>
                            <textarea class=”form–control”
                            id=”comments” name=”comments”
                            maxlength=”60”></textarea>
                        </div>

                        <div class=”mb–3”>
                            <label for=”time” class=”form–
                            label”>Select Time Slot</label>
                            <div type=”hidden” id=”timeSlots”>
                                <!–– Time slots will be
                                dynamically added here by
                                JavaScript ––>
                            </div>
                        </div>

                        <button type=”submit” class=”btn btn–
                        primary”>Book Appointment</button>
                    </form>
                </div>
            </div>
        </div>
```

```
    </div>
</div>

                </div>
            </div>
        </div>
    </div>
     <!-- Scroll to Top Button -->
  <a class="scroll-to-top rounded" href="#page-top">
        <i class="fas fa-angle-up"></i>
    </a>
<script>
    document.addEventListener('DOMContentLoaded', function() {
        var calendarEl = document.getElementById('calendar');

        var calendar = new FullCalendar.Calendar(calendarEl, {
            initialView: 'dayGridMonth',
            dateClick: function(info) {
                // Open the modal and set the date in the form
                    document.getElementById('appointmentDate')
                    .value = info.dateStr;
                    var bookingModal = new
                    bootstrap.Modal(document.getElementById('b
                    ookingModal'));
                    bookingModal.show();

                     // Load available time slots for the
                    selected date
                loadTimeSlots(info.dateStr);
            },
            events: [
                @foreach($appointments as $appointment)

                        {
                            title: '{{ $appointment->status
                            }}',
                            start: '{{ $appointment->date }}',
                                // Check if status is
                                'cancel', don't display in
                                calendar
                            display: '{{ $appointment->status
                            === "cancel" ? "none" : "block"
                            }}',
                            backgroundColor:
                                '{{ $appointment->status ===
                                "approved" ? "green" :
                                ($appointment->status ===
                                "pending" ? "orange" : "red")
                                }}',
                            borderColor:
                                '{{ $appointment->status ===
                                "approved" ? "green" :
                                ($appointment->status ===
                                "pending" ? "orange" : "red")
                                }}',
                        },

                @endforeach
            ],

        });

        calendar.render();
    });

    function loadTimeSlots(date) {
        // Clear previous time slots
        document.getElementById('timeSlots').innerHTML = '';

        // Fetch available time slots for the selected date
        fetch(`/available-slots/${date}`)
            .then(response => response.json())
            .then(slots => {
                slots.forEach(slot => {
                    let button =
                    document.createElement('button');
                    button.className = slot.booked ? 'btn btn-
                    danger' : 'btn btn-primary';
                    button.disabled = slot.booked;
                    button.textContent = slot.booked ?
                    'Booked' : slot.time;
                    button.type = 'button';

                    if (!slot.booked) {
                        button.addEventListener('click',
                        function() {
                            document.getElementById('selectedT
                            ime').value = slot.time;
                            console.log('Selected time:',
                            slot.time); // Log the selected
                            time for debugging
                        });
                    }

                    document.getElementById('timeSlots').appen
                    dChild(button);
                });
            });
    }


    function cancelAppointment(id) {
        // Make an AJAX call to cancel the appointment
        fetch(`/appointments/cancel/${id}`, {
            method: 'POST', // Assuming you use POST for
            canceling
            headers: {
                'Content-Type': 'application/json',
                'X-CSRF-TOKEN':
                document.querySelector('meta[name="csrf-
                token"]').getAttribute('content')
            },
            body: JSON.stringify({ id: id })
        })
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                // After canceling, reload the calendar
                events to reflect the changes
                reloadCalendarEvents();
            } else {
                alert('Failed to cancel the appointment');
            }
        })
        .catch(error => {
            console.error('Error canceling appointment:',
            error);
            alert('An error occurred while canceling the
            appointment.');
        });
    }

    function reloadCalendarEvents() {
        calendar.refetchEvents();  // Ensure the calendar is
        reloaded to reflect the change`
```

```
}


</script>

@endsection
```

notification.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-md-8">
                <div class="card border-left-info shadow">
                    <div class="card-header">
                        <h6 class="m-0 font-weight-bold text-
                        primary">Notifications</h6>
                    </div>
                    <div class="card-body">
                        @if ($notifications->isEmpty())
                            <p class="text-center text-
                            muted">No unread notifications at
                            the moment.</p>
                        @else
                            <ul class="list-group">
                                @foreach ($notifications as
                            $notification)
                                <li class="list-group-
                                item d-flex justify-
                                content-between align-
                                items-center">
                                    <div>
                                        <strong>{{
                                        $notification-
                                        >data['message']
                                        }}</strong><br>
                                        <small
                                        class="text-
                                        muted">
                                            {{
                                            $notification-
                                            >created_at-
                                            >diffForHumans
                                            () }}
                                        </small>
                                    </div>
                                    <form action="{{
                                    route('notifications.m
                                    arkAsRead',
                                    $notification->id)
                                    }}" method="POST">
                                        @csrf
                                        <button
```

```

                                            type="submit"
                                            class="btn btn-sm
                                            btn-outline-
                                            success">
                                                Mark as Read
                                            </button>
                                        </form>
                                    </li>
                                @endforeach
                            </ul>
                        @endif
                    </div>
                </div>
            </div>
        </div>
    </div>
@endsection
```

application-log0.blade.php

auth-session-status.blade.php

danger-button.blade.php

```
<button {{ $attributes->merge(['type' => 'submit', 'class' =>
'inline-flex items-center px-4 py-2 bg-red-600 border border-
transparent rounded-md font-semibold text-xs text-white
uppercase tracking-widest hover:bg-red-500 active:bg-red-700
focus:outline-none focus:ring-2 focus:ring-red-500 focus:ring-
offset-2 transition ease-in-out duration-150']) }}>
    {{ $slot }}
</button>
```

dropdown-link.blade.php

```
<a {{ $attributes->merge(['class' => 'block w-full px-4 py-2
text-start text-sm leading-5 text-gray-700 hover:bg-gray-100
focus:outline-none focus:bg-gray-100 transition duration-150
ease-in-out']) }}>{{ $slot }}</a>
```

dropdown.blade.php

```
@props(['align' => 'right', 'width' => '48', 'contentClasses'
=> 'py-1 bg-white'])

@php
switch ($align) {
    case 'left':
        $alignmentClasses = 'ltr:origin-top-left rtl:origin-
        top-right start-0';
        break;
    case 'top':
        $alignmentClasses = 'origin-top';
        break;
    case 'right':
    default:
        $alignmentClasses = 'ltr:origin-top-right rtl:origin-
        top-left end-0';
        break;
}

switch ($width) {
    case ’48’:
        $width = ’w–48’;
        break;
}
@endphp

<div class=”relative” x–data=”{␣open:␣false␣}”
@click.outside=”open␣=␣false” @close.stop=”open␣=␣false”>
    <div @click=”open␣=␣!␣open”>
        {{ $trigger }}
    </div>

    <div x–show=”open”
            x–transition:enter=”transition␣ease–out␣duration–
␣␣␣␣␣␣␣␣␣␣␣␣200”
            x–transition:enter–start=”opacity–0␣scale–95”
            x–transition:enter–end=”opacity–100␣scale–100”
            x–transition:leave=”transition␣ease–in␣duration–
␣␣␣␣␣␣␣␣␣␣␣␣75”
            x–transition:leave–start=”opacity–100␣scale–100”
            x–transition:leave–end=”opacity–0␣scale–95”
            class=”absolute␣z–50␣mt–2␣{{␣$width␣}}␣rounded–md
␣␣␣␣␣␣␣␣␣␣␣␣shadow–lg␣{{␣$alignmentClasses␣}}”
            style=”display:␣none;”
            @click="open␣=␣false">
        <div class="rounded-md␣ring-1␣ring-black␣ring-opacity-
␣␣␣␣␣␣␣␣5␣{{␣$contentClasses␣}}">
            {{ $content }}
        </div>
    </div>
</div>
```

input-error.blade.php

```
@props(['messages'])

@if ($messages)
    <ul {{ $attributes->merge(['class' => 'text-sm␣text-red-
␣␣␣␣600␣space-y-1']) }}>
        @foreach ((array) $messages as $message)
            <li>{{ $message }}</li>
        @endforeach
    </ul>
@endif
```

input-label.blade.php

```
@props(['value'])

<label {{ $attributes->merge(['class' => 'block␣font-medium
text-sm␣text-gray-700']) }}>
    {{ $value ?? $slot }}
</label>
```

modal.blade.php

```
@props([
        'name',
        'show' => false,
        'maxWidth' => '2xl'
])

@php
$maxWidth = [
        'sm' => 'sm:max-w-sm',
        'md' => 'sm:max-w-md',
        'lg' => 'sm:max-w-lg',
        'xl' => 'sm:max-w-xl',
        '2xl' => 'sm:max-w-2xl',
][$maxWidth];
@endphp

<div
        x-data="{
        show: @js($show),
        focusables() {
            // All focusable element types...
            let selector = 'a, button,
            input:not([type=\'hidden\']), textarea, select,
            details, [tabindex]:not([tabindex=\'-1\'])'
            return [...$el.querySelectorAll(selector)]
                // All non-disabled elements...
                .filter(el => ! el.hasAttribute('disabled'))
        },
        firstFocusable() { return this.focusables()[0] },
        lastFocusable() { return this.focusables().slice(-1)
        [0] },
        nextFocusable() { return this.focusables()
        [this.nextFocusableIndex()] || this.firstFocusable()
        },
        prevFocusable() { return this.focusables()
        [this.prevFocusableIndex()] || this.lastFocusable() },
        nextFocusableIndex() { return
        (this.focusables().indexOf(document.activeElement) +
        1) % (this.focusables().length + 1) },
        prevFocusableIndex() { return Math.max(0,
        this.focusables().indexOf(document.activeElement)) -1
        },
    }"
    x-init="$watch('show', value => {
        if (value) {
            document.body.classList.add('overflow-y-hidden');
            {{ $attributes->has('focusable') ? 'setTimeout(()
            => firstFocusable().focus(), 100)' : '' }}
        } else {
            document.body.classList.remove('overflow-y-
            hidden');
        }
    })"
    x-on:open-modal.window="$event.detail == '{{ $name }}' ?
    show = true : null"
    x-on:close-modal.window="$event.detail === '{{ $name }}' ?
    show = false : null"
    x-on:close.stop="show = false"
    x-on:keydown.escape.window="show = false"
    x-on:keydown.tab.prevent="$event.shiftKey ||
    nextFocusable().focus()"
    x-on:keydown.shift.tab.prevent="prevFocusable().focus()"
    x-show="show"
    class="fixed inset-0 overflow-y-auto px-4 py-6 sm:px-0 z-
    50"
    style="display: block;"

>
    <div
        x-show="show"
        class="fixed inset-0 transform transition-all"
        x-on:click="show = false"
        x-transition:enter="ease-out duration-300"
        x-transition:enter-start="opacity-0"
        x-transition:enter-end="opacity-100"
        x-transition:leave="ease-in duration-200"
        x-transition:leave-start="opacity-100"
        x-transition:leave-end="opacity-0"
    >
        <div class="absolute inset-0 bg-gray-500 opacity-75">
        </div>
    </div>

    <div
        x-show="show"
        class="mb-6 bg-white rounded-lg overflow-hidden
        shadow-xl transform transition-all sm:w-full {{
        $maxWidth }} sm:mx-auto"
        x-transition:enter="ease-out duration-300"
        x-transition:enter-start="opacity-0 translate-y-4
        sm:translate-y-0 sm:scale-95"
        x-transition:enter-end="opacity-100 translate-y-0
        sm:scale-100"
        x-transition:leave="ease-in duration-200"
        x-transition:leave-start="opacity-100 translate-y-0
        sm:scale-100"
        x-transition:leave-end="opacity-0 translate-y-4
        sm:translate-y-0 sm:scale-95"
    >
        {{ $slot }}
    </div>
</div>
```

nav-link.blade.php

```
@props(['active'])

@php
$classes = ($active ?? false)
```

```
            ? 'inline-flex items-center px-1 pt-1 border-b-2
            border-indigo-400 text-sm font-medium leading-5
            text-gray-900 focus:outline-none focus:border-
            indigo-700 transition duration-150 ease-in-out'
            : 'inline-flex items-center px-1 pt-1 border-b-2
            border-transparent text-sm font-medium leading-5
            text-gray-500 hover:text-gray-700 hover:border-
            gray-300 focus:outline-none focus:text-gray-700
            focus:border-gray-300 transition duration-150
            ease-in-out';
@endphp

<a {{ $attributes->merge(['class' => $classes]) }}>
    {{ $slot }}
</a>
```

primary-button.blade.php

```
<button {{ $attributes->merge(['type' => 'submit', 'class' =>
'inline-flex items-center px-4 py-2 bg-gray-800 border border-
transparent rounded-md font-semibold text-xs text-white
uppercase tracking-widest hover:bg-gray-700 focus:bg-gray-700
active:bg-gray-900 focus:outline-none focus:ring-2 focus:ring-
indigo-500 focus:ring-offset-2 transition ease-in-out
duration-150']) }}>
    {{ $slot }}
</button>
```

responsive-nav-link.blade.php

```
@props(['active'])

@php
$classes = ($active ?? false)
            ? 'block w-full ps-3 pe-4 py-2 border-l-4 border-
            indigo-400 text-start text-base font-medium text-
            indigo-700 bg-indigo-50 focus:outline-none
            focus:text-indigo-800 focus:bg-indigo-100
            focus:border-indigo-700 transition duration-150
            ease-in-out'
            : 'block w-full ps-3 pe-4 py-2 border-l-4 border-
            transparent text-start text-base font-medium text-
            gray-600 hover:text-gray-800 hover:bg-gray-50
            hover:border-gray-300 focus:outline-none
            focus:text-gray-800 focus:bg-gray-50 focus:border-
            gray-300 transition duration-150 ease-in-out';
@endphp

<a {{ $attributes->merge(['class' => $classes]) }}>
    {{ $slot }}
</a>
```

secondary-button.blade.php

```
<button {{ $attributes->merge(['type' => 'button', 'class' =>
'inline-flex items-center px-4 py-2 bg-white border border-
gray-300 rounded-md font-semibold text-xs text-gray-700
uppercase tracking-widest shadow-sm hover:bg-gray-50
focus:outline-none␣focus:ring-2␣focus:ring-indigo-500
focus:ring-offset-2␣disabled:opacity-25␣transition␣ease-in-
out␣duration-150’]) }}>
    {{ $slot }}
</button>
```

text-input.blade.php

```
@props([’disabled’ => false])

<input {{ $disabled ? ’disabled’ : ’’ }} {!! $attributes-
>merge([’class’ => ’border-gray-300␣focus:border-indigo-500
focus:ring-indigo-500␣rounded-md␣shadow-sm’]) !!}>
```

create.blade.php

```
@extends(’layouts.app’)
    @section(’content’)

            <!-- Begin Page Content -->
        <div class=”container-fluid␣sm:px-4␣pt-4␣␣”>
            <div class=”card␣shadow␣mb-2”>
                <div class=”card-header”>
                    <h5  class=”m-0␣text-gray-800␣text-sm
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣font-weight-bold”>Add New Medicine</h5>
                <hr />
                <br />
                @if(Session::has(’success’))
                    <div class=”alert␣alert-success” role
                    =”alert”>
```

```
                        {{ Session::get('success') }}
                    </div>
                @endif

                @if (session('error'))
                        <div class="alert alert-danger">{{
                        session('error') }}</div>
                @endif
                        <!-- Form to create a new medicine -->
                        <form action="{{
                        route('inventory.store') }}"
                        method="POST" enctype="multipart/form-
                        data">

                            @csrf
                            <div class="row mb-3">
                                <div class="col">
                                    <label
                                    for="name">Medicine
                                    Name</label>
                                    <input type="text"
                                    name="name" id="name"
                                    class="form-control
                                    @error('name') is-invalid
                                    @enderror"  required>
                                    @error('name') <div
                                    class="invalid-feedback">
                                    {{ $message }}</div>
                                    @enderror
                                </div>

                                <div class="col">
                                    <label
                                    for="expiry_date">Expiry
                                    Date</label>
                                    <input type="date"
                                    name="expiry_date"
                                    id="expiry_date"
                                    class="form-control
                                    @error('expiry_date') is-
                                    invalid @enderror"
                                    required>
                                    @error('expiry_date')
                                    <div class="invalid-

                                    feedback">{{ $message }}
                                    </div> @enderror
                                </div>
                            </div>

                            <div class="row mb-3">
                                <div class="col">
                                    <label
                                    for="miligram">Milligram</
                                    label>
                                    <div class=”input−group
                                    mb−2”>
                                        <input type=”number”
                                        name=”miligram”
                                        class=”form−control
                                        @error('miligram') is−
                                        invalid @enderror”

                                        required>
                                        <div class=”input−
                                        group−append”>
                                            <span
                                            class=”input−
                                            group−
                                            text”>mg</span>
                                        </div>
                                    </div>
                                    @error('miligram') <div
                                    class=”invalid−feedback”>
                                    {{ $message }}</div>
                                    @enderror
                                </div>

                                <div class=”col”>
                                    <label
                                    for=”dosage”>Dosage</label
```

```
                                >
                                <input type="number"
                                name="dosage" id="dosage"
                                class="form-control
                                 @error('dosage') is-
                                invalid @enderror"
                                required>
                                @error('dosage') <div
                                class="invalid-feedback">
                                {{ $message }}</div>
                                @enderror
                            </div>
                        </div>

                        <div class="row mb-3">
                            <div class="col">
                                <label
                                for="batch_supply">Batch
                                Supply</label>
                                <input type="number"
                                name="batch_supply"
                                id="batch_supply"
                                class="form-control
                                 @error('batch_supply') is-
                                invalid @enderror"
                                placeholder="Total
                                Supply" required>
                                @error('batch_supply')
                                <div class="invalid-
                                feedback">{{ $message }}
                                </div> @enderror
                            </div>
                            <div class="col">
                                <label
                                for="batch_number">Batch
                                Number</label>
                                <input type="text"
                                name="batch_number"
                                id="batch_number"
                                class="form-control
                                @error('batch_number') is-
                                invalid @enderror">
                                @error('batch_number')
                                <div class="invalid-
                                feedback">{{ $message }}
                                </div> @enderror
                            </div>
                            <div class="col">
                                <label
                                for="batch_date_received">
                                Batch Date
                                Received</label>
                                <input type="date"
                                name="batch_date_received"
                                id="batch_date_received"
                                class="form-control
                                @error('batch_date_receive
                                d') is-invalid @enderror">
                                @error('batch_date_receive
                                d') <div class="invalid-
                                feedback">{{ $message }}
                                </div> @enderror
                            </div>
                        </div>

                        <hr />
                        <br />

                        <button type="submit" class="btn
                        btn-primary">Add Medicine</button>
                        <a href="{{
                        route('inventory.index') }}"
                        class="btn btn-
                        secondary">Cancel</a>
                    </form>
                </div>

            </div>
        </div>
    </div>
@endsection
```

edit.blade.php

```
@extends('layouts.app')

    @section('content')

    <!-- Begin Page Content -->
    <div class="container-fluid sm:px-4 pt-4  ">
            <div class="card shadow mb-2">
                <div class="card-header">
                    <h5  class="m-0 text-gray-800 text-sm
                    font-weight-bold">Edit Medicine
                    Information</h5>
                <hr />
                <br />
            @if(Session::has('success'))
                <div class="alert alert-success" role="alert">
                    {{ Session::get('success') }}
                </div>
            @endif

            @if (session('error'))
                <div class="alert alert-danger">{{
                session('error') }}</div>
            @endif
    <!-- Form to edit an existing medicine -->
    <form action="{{ route('inventory.update', $inventory)
    }}" method="POST">
```

```
            @csrf
            @method('PUT')
            <div class=”row mb-3”>

                <div class=”col”>
                    <label for=”name”>Name</label>
                    <input type=”text” name=”name” id=”name”
                    class=”form-control” value=”{{ $inventory-
                    >name }}” required>
                </div>

                <div class=”col”>
                    <label for=”expiry_date”>Expiry Date</label>
                    <input type=”date” name=”expiry_date”
                    id=”expiry_date” class=”form-control” value=”
                    {{ $inventory->expiry_date->format('Y-m-d')
                    }}” required>
                </div>

                <div class=”col”>
                    <label for=”miligram”>Miligram</label>
                    <div class=”input-group mb-2”>
                        <input type=”number” name=”miligram”
                        id=”miligram” step=”0.01” class=”form-
                        control” value=”{{ $inventory->miligram
                        }}” required>
                            <div class=”input-group-append”>
                            <span class="input-group-
                            text">mg</span>
                        </div>
                </div>
            </div>

        </div>

        <div class="row mb-3">
            <div class="col">
                <label for="dosage">Dosage (number of dosage
                per day)</label>
                <input type="number" name="dosage"
                id="dosage" class="form-control" value="{{
                $inventory->dosage }}" required>
            </div>

            <div class="col">
                <label for="quantity">Quantity</label>
                <input type="number" name="quantity"
                id="quantity" class="form-control" value="{{
                $inventory->quantity }}" required>
            </div>

            <div class="col">
                <label for="batch_supply">Batch Supply</label>
                <input type="text" name="batch_supply"
                id="batch_supply" class="form-control" value="
                {{ $inventory->batch_supply }}" required>
            </div>
        </div>

        <div class="row mb-2">
            <div class="col">
                <label for="batch_number">Batch Number</label>
                <input type="text" name="batch_number"
                id="batch_number" class="form-control" value="
                {{ $inventory->batch_number }}" required>
            </div>

            <div class="col">
                <label for="batch_date_received">Batch Date
                Received</label>
                <input type="date" name="batch_date_received"
                id="batch_date_received" class="form-control"
                value="{{  $inventory->batch_date_received
                }}" required>
            </div>

        </div>
        <hr />
        <br />

        <button type="submit" class="btn btn-primary">Update
        Medicine</button>
        <a href=”{{␣route(’inventory.index’)␣}}” class=”btn
␣␣␣␣␣␣␣␣btn-secondary”>Cancel</a>

    </form>


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

    @endsection
```

index.blade.php

```
@extends(’layouts.app’)
@section(’content’)
    <!-- Begin Page Content -->
    <div class=”container-fluid␣p-4␣text-sm␣max-w-7xl␣min-w-
␣␣␣␣3”>
        <!-- DataTables -->
        <div class=”card␣shadow␣mb-4”>
            <div class=”card-header” style=”background-color:
␣␣␣␣␣␣␣␣␣␣␣␣#35495e;”>
                <h6 class=”m-0␣text-white␣font-weight-
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣bold”>Medicine Inventory Records</h6>
            </div>

            <div class="card-body">
                <div class="table-responsive">
                    <div class="flex align-items-center py-2">
                        <a href="{{ route('inventory.create')
                        }}" class="btn btn-primary py-2 text-
                        sm">Add Medicine Stock</a>
                        <a href="{{
                        route('reports.inventory_report') }}"
                        class="btn btn-secondary py-2 text-sm
                        ml-2" target="_blank">Download Report
                        <i class="fas fa-download"></i> </a>
                        <!-- Prescription Button -->
                        <button type="button" class="btn btn-
                        success py-2 text-sm" data-
                        toggle="modal" data-
                        target="#prescribeModal">
                            Release Medicine
                        </button>

                    </div>
                    <hr />
                    <br />

                    @if(Session::has('success'))
                    <div class="alert alert-success"
                    role="alert">
```

```
    {{ Session::get(’success’) }}
</div>
@endif

@if (session(’error’))
    <div class=”alert alert-danger”>{{
    session(’error’) }}</div>
@endif

<table class=”table table-bordered”
id=”dataTable” width=”100%”
cellspacing=”0”>
    <thead class=”table-primary text-gray-
    800”>
        <tr>
            <th>Id</th>
            <th>Medicine Name</th>
            <th>Batch Number</th>
            <th>Expiry Date</th>
            <th>Batch Supply</th>
            <th>Batch Date Received</th>
            <th>Miligram</th>
            <th>Dosage</th>
            <th>Quantity</th>
            <th>Actions</th>
        </tr>
    </thead>
```

```
                  <tbody>
                     @forelse ($inventory as $item)
                     <tr>
                        <td class="align-middle">{{
                        $loop->iteration }}</td>
                        <td class="align-middle">{{
                        $item->name }}</td>
                        <td class="align-middle">{{
                        $item->batch_number }}</td>
                        <td class="align-middle">{{
                        $item->expiry_date->format('Y-
                        m-d') }}</td>
                        <td class="align-middle">{{
                        $item->batch_supply }}</td>
                        <td class="align-middle">{{
                        $item->batch_date_received }}
                        </td>
                        <td class="align-middle">{{
                        $item->miligram }} mg</td>
                        <td class="align-middle">{{
                        $item->dosage }} times a
                        day</td>
                        <td class="align-middle
                           @if($item->stock_status
                           == 'warning') bg-warning
                           text-white
                           @elseif($item-
                           >stock_status ==
                           ’critical’) bg-danger
                           text-white
                           @elseif($item-
                           >stock_status == ’out’)
                           bg-active text-white
                           @endif
                        ”>{{ $item->quantity }}</td>
                        <td class=”align-middle”>
                           <div class=”btn-group”
                           role=”btn-group” aria-
                           label=”Basic example”>
                               @if ($item->quantity
                               == 0)
                                   <form action=”{{
                                   route(’inventory.s
                                   tockOut’, $item-
                                   >id) }}”
                                   method=”POST”>
                                       @csrf
                                       <button
                                       type=”submit”
                                       class=”btn
                                       btn-
                                       danger”>Move
                                       to Stock
                                       Out</button>
```

```
                                        </form>
                                    @else
                                        <!-- Optional:
                                        Edit/Delete
                                        actions here -->
                                        <a href="{{
                                        route('inventory.s
                                        how', $item->id)
                                        }}" type="button"
                                        class="btn btn-
                                        success h-
                                        10">Detail</a>
                                        <a href="{{
                                        route('inventory.e
                                        dit', $item-
                                        >id)}}"
                                        ="button"
                                        class="btn btn-
                                        warning h-
                                        10">Edit</a>
                                    @endif
                                </div>
                            </td>
                        </tr>
                        @empty
                        <tr>
                            <td colspan="8" class="text-
```

```
                                center">No medicines
                                found</td>
                            </tr>
                            @endforelse
                        </tbody>
                    </table>


                    <!-- DataTables Scripts -->
                    <script
                    src="https://code.jquery.com/jquery-
                    3.5.1.min.js"></script>
                    <link rel="stylesheet"
                    href="https://cdn.datatables.net/1.13.5/cs
                    s/jquery.dataTables.min.css">
                    <script
                    src="https://cdn.datatables.net/1.13.5/js/
                    jquery.dataTables.min.js"></script>

                    <script>
                        $(document).ready(function() {
                            $('#prescriptionTable').DataTable(
                            {
                                "order": [], // Disable
                                initial sorting
                                "paging": true, // Enable
                                pagination
                                "searching": true, // Enable
                                search
                                "info": true, // Enable table
                                information display
                                "lengthChange": true // Allow
                                changing the number of
                                records per page
                            });
                        });
                    </script>


        </div>

        </div>
    </div>
</div>
<!--
**************************************************************
**************************************************************
**********************************************-->


<!-- Stock Out Table Included Here -->
@include('inventory.stock_out')

<!--
```

```
**************************************************************
**************************************************************
********************************************-->


        <!-- Prescription History -->
    <div class="container-fluid p-4 text-sm max-w-7xl min-w-
    3">
        <div class="card shadow mb-4">
            <div class="card-header" style="background-color:
            #35495e;">
                <h6 class="m-0 text-white font-weight-
                bold">Medicine Release History</h6>
            </div>

        <div class="card-body">
            <div class="table-responsive">

            <br />
                    @if(Session::has('success'))
                    <div class="alert alert-success"
                    role="alert">
                        {{ Session::get('success') }}
                    </div>
                    @endif

                    @if (session('error'))
                        <div class="alert alert-danger">{{
                        session('error') }}</div>
                    @endif
            <table class="table table-striped"
            id="prescriptionTable" width="100%"
            cellspacing="0">
                <thead class="table-primary text-gray-800">
                    <tr>

                        <th>Patient Name</th>
                        <th>Medicine Name</th>
                        <th>Dosage</th>
                        <th>Milligrams</th>
                        <th>Batch Number</th>
                        <th>Quantity</th>
                        <th>Date Prescribed</th>
                    </tr>
                </thead>
                <tbody>
                @forelse ($prescriptions as $prescription)
                    <tr>

                        <td class="align-middle">{{
                        $prescription->patient ?
                        $prescription->patient->full_name :
                        'No patient found' }}</td>
                        <td class="align-middle">{{
                        $prescription->medicine ?
                        $prescription->medicine->name : 'No
                        medicine found' }}</td>
                        <td class="align-middle">{{
                        $prescription->dosage }} times a
                        day</td>
                        <td class="align-middle">{{
                        $prescription->milligram }} mg</td>
                        <td class="align-middle">{{
                        $prescription->batch_number}}</td>
                        <td class="align-middle">{{
                        $prescription->quantity }}</td>
                        <td class="align-middle">{{
                        $prescription->created_at->format('Y-
                        m-d') }}</td>
                    </tr>
                @empty
                    <tr>
                        <td colspan="7" class="text-
                        center">No prescription history
                        found</td>
                    </tr>
                @endforelse

                </tbody>
            </table>
            </div>
```

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


<!-- Prescription Modal -->
<div class=”modal fade” id=”prescribeModal” tabindex=”-1”
aria-labelledby=”prescribeModalLabel” aria-hidden=”true”>
    <div class=”modal-dialog”>
        <div class=”modal-content”>
            <div class=”modal-header” style=”background-
            color: #35495e;”>
                <h5 class=”modal-title text-white”
                id=”prescribeModalLabel”>Release Medicine to
                Patient</h5>
                <button type=”button” class=”close” data-
                dismiss=”modal” aria-label=”Close”>
                    <span aria-hidden=”true”>&times;</span>
                </button>
            </div>
            <form action=”{{ route('inventory.prescribe') }}”
            method=”POST”>
                @csrf
                <div class=”modal-body”>
                    <div class=”form-group”>
                        <label for=”patient_id”>Select
                        Patient</label>
                        <select name="patient_id"
                        id="patient_id" class="form-control"
                        required>
                            <option value="">Choose
                            Patient</option>
                            @foreach($patients as $patient)
                                <option value="{{ $patient-
                                >id }}">{{ $patient-
                                >full_name }}</option>
                            @endforeach
                        </select>
                    </div>
                    <div class="form-group">
                        <label for="medicine_id">Select
                        Medicine</label>
                        <select name="medicine_id"
                        id="medicine_id" class="form-control"
                        required>
                            <option value="">Choose
                            Medicine</option>
                            @foreach($inventory as $item)
                                <option value="{{ $item->id
                                }}"
                                        data-milligram="{{
                                        $item->miligram }}"
                                        data-batch-number="{{
                                        $item->batch_number
```

```
                                            }}">
                                    {{ $item->name }} : ({{
                                    $item->miligram }} mg)
                                </option>
                            @endforeach
                        </select>
                    </div>

                    <div class="form-group">
                        <label for="dosage">Dosage (times per
                        day)</label>
                        <input type="number" name="dosage"
                        id="dosage" class="form-control"
                        min="1" required>
                    </div>
                    <div class="form-group">
                        <label for="quantity">Quantity</label>
                        <input type="number" name="quantity"
                        id="quantity" class="form-control"
                        min="1" required>
                    </div>
                </div>
                <div class="modal-footer">
                    <button type="button" class="btn btn-
                    secondary" data-
                    dismiss="modal">Close</button>
                    <button type="submit" class="btn btn-
                    primary">Prescribe</button>
                    </div>
                </form>
            </div>
        </div>
</div>

<!-- End of Prescription Modal -->


<!--
**************************************************************
**************************************************************
**********************************************-->


    <!-- Scroll to Top Button -->
    <a class="scroll-to-top rounded" href="#page-top">
        <i class="fas fa-angle-up"></i>
    </a>

    @push('scripts')
    <script>
        document.getElementById('medicine_id').addEventListene
        r('change', function () {
            const selectedOption =
            this.options[this.selectedIndex];
            const milligram =
            selectedOption.getAttribute('data-milligram');
            const batchNumber =
            selectedOption.getAttribute('data-batch-number');

            document.getElementById('milligram').value =
            milligram || ''; // Default to empty if null
            document.getElementById('batch_number').value =
            batchNumber || ''; // Default to empty if null
        });
    </script>
    @endpush

@endsection
```

show.blade.php

```
@extends('layouts.app')

    @section('content')

    <!-- Begin Page Content -->
        <div class="container-fluid sm:px-4 pt-4  ">
            <div class="card shadow mb-2">
                <div class="card-header">
                    <h6 class="m-0 text-gray text-sm font-
                    weight-bold">Medicine Details</h6>
                    <hr />
                <br />
                <div class="row mb-3">
                    <div class="col">
                    <label for="name">Medicine
                    Name</label>
                        <input type="text" name="name"
                        class="form-control"
                        placeholder="Name" value=" {{
                        $inventory->name }}" readonly>
                    </div>

                    <div class="col">
                    <label for="expiry_date">Expiry
                    </label>
                        <input type="date"
                        ="expiry_date" class="form-
                        control"
                        placeholder="Expiry_Date" value="
                        {{ $inventory->expiry_date-
                        >format('Y-m-d') }}" readonly>
                    </div>

                    <div class="col">
                    <label for="miligram">Miligram</label>
                        <input type="number"
                        name="miligram" class="form-
                        control" step="0.01"
                placeholder="Miligram" value="{{
                $inventory->miligram }}" readonly>
            </div>
        </div>
        <div class="row mb-3">
            <div class="col ">
            <label for="dosage">Dosage</label>
                <input type="text" name="dosage"
                class="form-control"
                placeholder="Dosage" value="{{
                $inventory->dosage }}" readonly>
            </div>

            <div class="col">
            <label for="quantity">Quantity</label>
                <input type="number"
                name="quantity" class="form-
                control" placeholder="Quantity"
                value="{{ $inventory->quantity
                }}" readonly>
            </div>

            <div class="col">
            <label for="batch_supply">Batch
            Supply</label>
                <input type="text"
                name="batch_supply" class="form-
```

```
                            control" placeholder="Batch
                            Supply" value="{{ $inventory-
                            >batch_supply }}" readonly>
                        </div>
                    </div>
                    <div class="row mb-3">
                        <div class="col ">
                        <label for="batch_number">Batch
                        Number</label>
                            <input type="text"
                            name="batch_number" class="form-
                            control" placeholder="Batch
                            Number" value="{{ $inventory-
                            >batch_number }}" readonly>
                        </div>

                        <div class="col">
                        <label
                        for="batch_date_received">Batch Date
                        Received</label>
                            <input type="date"
                            name="batch_date_received"
                            class="form-control"
                            placeholder="Batch Date Received"
                            value="{{ $inventory-
                            >batch_date_received }}" readonly>
                        </div>
                    </div>

                </div>
            </div>
            <a href="{{ route('inventory.index') }}"
            class="btn btn-secondary">Cancel</a>
        </div>

@endsection
```

stock$_o$*ut.blade.php*

```

<div class="container-fluid p-4 text-sm max-w-7xl min-w-3">
        <!-- DataTables -->
    <div class="card shadow mb-4">
        <div class="card-header" style="background-color: #35495e;">
            <h6 class="text-white ">Stock Out</h6>
        </div>

        <div class="card-body">
            <div class="table-responsive">

            @if(Session::has('success'))
                <div class="alert alert-success" role="alert">
                    {{ Session::get('success') }}
                </div>
            @endif

        @if (session('error'))
            <div class="alert alert-danger">{{
            session('error') }}</div>
        @endif


        <table class="table table-striped w-100"
        id="stockOutTable" cellspacing="0">
            <thead class="table-primary text-gray-800">
                <tr>
                    <th>Name</th>
                    <th>Expiry Date</th>
                    <th>Milligram</th>
                    <th>Dosage</th>
                    <th>Quantity</th>
                    <th>Batch Number</th>

                </tr>
            </thead>
            <tbody>
                @foreach ($stockOutItems as $item)
                    <tr class="{{ $item->expiry_date <
                    now() ? 'bg-danger text-white' : ''
                    }} {{ $item->quantity == 0 ? 'bg-
                    danger text-white' : '' }}">
                        <td>{{ $item->name }}</td>
```

```
                                <td>
                                    {{ $item->expiry_date }}
                                     @if ($item->expiry_date <
                                     now())
                                        <!-- Badge for expired
                                        items -->
                                        <span class="badge badge-
                                        danger">Expired</span>
                                    @endif
                                </td>
                                <td>{{ $item->miligram }} mg</td>
                                <td>{{ $item->dosage }} times a
                                day</td>
                                <td>{{ $item->quantity }}
                                    @if ($item->quantity == 0)
                                        <!-- Badge for expired
                                        items -->
                                        <span class="badge badge-
                                        danger">Out of
                                        Stock</span>
                                    @endif
                                </td>
                                <td>{{ $item->batch_number }}</td>
                            </tr>
                        @endforeach
                    </tbody>
                </table>
```

```



            <!-- JavaScript to handle modal form action
            dynamically -->
            <script>
                document.addEventListener('DOMContentLoaded',
                function () {
                        // Initialize DataTables
                        $('#stockOutTable').DataTable({
                            paging: true,
                            searching: true,
                            ordering: true,
                            lengthMenu: [10, 25, 50, 100]
                        });

                });
            </script>

            </div>
        </div>

    </div>
</div>
```

app.blade.php

```
<!DOCTYPE html>
```

```
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width,
        initial-scale=1">
        <meta name="csrf-token" content="{{ csrf_token() }}">

        <title>Mindcare</title>


            <!-- Favicon (Logo) -->
        <link rel="icon" href="{{ asset('img/favicon.png')
        }}" type="image/x-icon">
        <!-- Custom CSS -->
        <link href="{{ asset('css/app.css') }}"
        rel="stylesheet" />

        <!-- Custom fonts for this template-->
        <link href="https://fonts.googleapis.com/css?
        family=Nunito:200,200i,300,300i,400,400i,600,600i,700,
        700i,800,800i,900,900i" rel="stylesheet">
        <link rel="stylesheet"
        href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/
        /css/bootstrap.min.css">
        <link
        href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dis
        t/css/bootstrap.min.css" rel="stylesheet">

        <!-- Custom styles for this template-->
        <link href="{{ asset('sd-admin/css/sb-admin-
        2.min.css') }}" rel="stylesheet">

        <!-- Custom fonts for this template-->
        <link href="{{ asset('sd-admin/vendor/fontawesome-
        free/css/all.min.css') }}" rel="stylesheet"
        type="text/css">
        <link href="{{ asset('sd-
        admin/vendor/datatables/dataTables.bootstrap4.min.css'
        )}}" rel="stylesheet">

        <!-- Fonts -->
        <link rel="preconnect" href="https://fonts.bunny.net">
        <link href="https://fonts.bunny.net/css?
        family=figtree:400,500,600&display=swap"
        rel="stylesheet" />

         <!-- Custom fonts for this template-->
        <link rel="stylesheet"
        href="https://cdn.datatables.net/1.10.24/css/jquery.da
        taTables.min.css">
        <link rel="stylesheet"
        href="https://cdn.datatables.net/buttons/1.7.1/css/but
        tons.dataTables.min.css">
        <link href="https://fonts.googleapis.com/css?
        family=Nunito:200,200i,300,300i,400,400i,600,600i,700,
        700i,800,800i,900,900i" rel="stylesheet">
        <link
        href="https://cdnjs.cloudflare.com/ajax/libs/font-
        awesome/6.0.0-beta3/css/all.min.css" rel="stylesheet">
        <link
        href="https://cdnjs.cloudflare.com/ajax/libs/font-
        awesome/5.15.4/css/all.min.css" rel="stylesheet">


        <style>
        /* Add this in your CSS file */

            #sidebar {
                margin-left: 0;
                padding-left: 0;
            }
            @media (max-width: 768px) {

                #accordionSidebar {
                    width: 80px;
                }
                #accordionSidebar .nav-link .sidebar-text {
                    display: none; /* Hide text on smaller
                    screens */
                }
                #accordionSidebar .nav-link {
```

```
            justify-content: center; /* Center the
            icons */

        }

        /* Hide the text and brand name for icons-
        only view */
        #accordionSidebar .nav-link span,
        #accordionSidebar .sidebar-heading,
        #accordionSidebar .sidebar-brand {
            display: none;
        }
        /* Apply padding and font-size adjustments
        for smaller screens */
        .container-fluid {
            padding: 0.5rem;
        }

        .container-fluid .row {
            margin: 0; /* Reduce row margin on small
            screens */
        }

        .col-xl-3, .col-md-4, .col-xl-8, .col-lg-7,
        .col-lg-5 {
            flex: 0 0 100%; /* Stack columns
            vertically */
            max-width: 100%;
        }

        /* Adjust card padding and font sizes */
        .card {
            padding: 0.5rem;
        }

        .h3, .h5, .h1 , .h2{
            font-size: 1.25rem; /* Scale down headers
            */
        }

        .text-xs {
            font-size: 0.75rem;
        }

        /* Adjust canvas and charts */
        canvas {
            max-width: 100%;
        }

        /* Custom badge style */
        .badge-expired {
            background-color: #dc3545; /* Red color
            for expired items */
            color: white;
```

```
                }

                /* Custom row highlight for expired items */
                .table tr.bg-danger {
                    background-color: #f8d7da !important; /*
                    Light red background */
                }


            }
        </style>
    </head>

<body id="page-top" class="relative">

    <div class="max-h-screen md:h-full sm:h-full bg-gray-100">
        @include('layouts.navigation')
    </div>



    <div id="wrapper">

    @include('layouts.sidebar')
     <!-- Page Heading -->

        <!-- Begin Page Content -->
        <div class="container-fluid sm:px-4 text-sm mt-2 ">
```

```

            <main>
            @isset($header)
                <header class="bg-white text-gray-900 shadow">
                    <div class="mx-auto px-2 sm:px-2 md:px-4
                    lg:px-8">
                        {{ $header }}
                    </div>
                </header>
            @endisset
                @yield('content')
            </main>


        </div>
        <!-- End Page Content -->
    </div>

      <!-- Scroll to Top Button -->
    <a class="scroll-to-top rounded" href="#page-top">
        <i class="fas fa-angle-up"></i>
    </a>


    @stack('scripts') <!-- For pushing additional scripts
    from specific pages -->
                <!-- Bootstrap core JavaScript-->
                <script src="{{ asset('sd-
                admin/vendor/jquery/jquery.min.js') }}">
                </script>
                <!-- jQuery -->
                <script src="https://code.jquery.com/jquery-
                3.5.1.slim.min.js"></script>

                <script src="{{ asset('sd-
                admin/vendor/bootstrap/js/bootstrap.bundle.min
                .js') }}"></script>
                <!-- Core plugin JavaScript-->
                <script src="{{ asset('sd-admin/vendor/jquery-
                easing/jquery.easing.min.js') }}"></script>
                <!-- Custom scripts for all pages-->
                <script src="{{ asset('sd-admin/js/sb-admin-
                2.min.js') }}"></script>
                <!-- Page level plugins -->
                <script src="{{ asset('sd-
                admin/vendor/chart.js/Chart.min.js') }}">
                </script>
                <script src="{{ asset('sd-
                admin/js/demo/datatables-demo.js') }}">
                </script> <!-- DataTables scripts -->
                <script src="{{ asset('sd-
                admin/vendor/datatables/jquery.dataTables.min.
                js') }}"></script>
                <script src="{{ asset('sd-
                admin/vendor/datatables/dataTables.bootstrap4.
                min.js') }}"></script>

                <!-- Page level plugins -->
                <script src="{{ asset('sd-
                admin/vendor/chart.js/Chart.min.js')}}">
                </script>

                <!-- Page level custom scripts -->
                <script src="{{ asset('sd-admin/js/demo/chart-
                area-demo.js')}}"></script>
                <script src="{{ asset('sd-admin/js/demo/chart-
                pie-demo.js')}}"></script>

                <script src="{{ asset('sd-
                admin/vendor/bootstrap/js/bootstrap.bundle.min
                .js')}}"></script>
                <script src="//unpkg.com/alpinejs" defer>
                </script>
                <!-- jQuery -->
                <script src="https://code.jquery.com/jquery-
                3.6.0.min.js"></script>
                <!-- Bootstrap JS and Popper.js -->
                <script
                src="https://cdn.jsdelivr.net/npm/@popperjs/co
                re@2.10.2/dist/umd/popper.min.js"></script>
                <script
```

```
            src="https://cdn.jsdelivr.net/npm/bootstrap@5.
            1.3/dist/js/bootstrap.min.js"></script>

            <script src="{{ asset('sd-
            admin/vendor/bootstrap/js/bootstrap.min.js')}}
            "></script>


            <!-- DataTables JS -->
            <script src="https://code.jquery.com/jquery-
            3.6.0.min.js"></script>
            <script
            src="https://cdn.datatables.net/1.10.24/js/jqu
            ery.dataTables.min.js"></script>
            <script
            src="https://cdn.datatables.net/buttons/1.7.1/
            js/dataTables.buttons.min.js"></script>


            <script
            src="https://cdn.jsdelivr.net/npm/fullcalendar
            @6.1.0/index.global.min.js"></script>
            <!-- Bootstrap JS -->
            <script
            src="https://cdn.jsdelivr.net/npm/bootstrap@5.
            3.0-alpha1/dist/js/bootstrap.bundle.min.js">
            </script>
```

```


<script>
        document.getElementById('sidebarToggleTop').addEventLi
        stener('click', function() {
        const sidebar = document.getElementById('sidebar');
        sidebar.classList.toggle('hidden'); // This toggles
        the visibility of the sidebar
    });
    document.addEventListener("DOMContentLoaded", function ()
    {
        // Elements
        const sidebarToggleTop =
        document.getElementById('sidebarToggleTop');
        const accordionSidebar =
        document.getElementById('accordionSidebar');

        // Ensure both elements exist
        if (sidebarToggleTop && accordionSidebar) {
            // Toggle 'toggled' class on the sidebar when the
            button is clicked
            sidebarToggleTop.addEventListener('click',
            function () {
                accordionSidebar.classList.toggle('toggled');
            });
        } else {
            console.error("Sidebar␣or␣toggle␣button␣not
            found.");
        }
        });

        document.addEventListener("DOMContentLoaded",
        function () {
            const sidebarToggleTop =
            document.getElementById('sidebarToggleTop');
            const accordionSidebar =
            document.getElementById('accordionSidebar');

            if (sidebarToggleTop && accordionSidebar) {
                sidebarToggleTop.addEventListener('click',
                function () {
                    accordionSidebar.classList.toggle('toggled
                    ');
                    document.body.classList.toggle('sidebar-
                    toggled');
                });
            }
        });

    document.getElementById("sidebarToggle").addEventListener(
    "click", function () {
        document.getElementById("accordionSidebar").classList.
        toggle("hidden");
    });
```

```
</script>


    <script>
        $(document).ready(function() {
            $('#dataTable').DataTable({
                "order": [], // No default ordering"paging":
                true, // Enable pagination
                columnDefs: [
                { orderable: false, targets: -1 }, // Make
                the first column non-orderable for selection
                checkboxes
                { searchable: false, targets: -1 } // Last
                column is not searchable
                ],
                "pageLength": 100, // Default number of
                entries
                "searching": true, // Enable searching
                "info": true, // Enable table information
                display
                "lengthChange": true, // Allow user to change
                the number of records per page
                "destroy": true,

                });
        });
    </script>
```

```

<!-- Inline PHP Logic for Chart Data -->
@php
    $barangayCounts =
    \App\Models\Patient::select('barangay',
    \DB::raw('count(*)␣as␣total'))
        ->groupBy('barangay')
        ->get();

    $barangayNames = $barangayCounts->pluck('barangay');
    $patientCounts = $barangayCounts->pluck('total');
@endphp

<!-- Chart.js Initialization -->
<script>
    document.addEventListener('DOMContentLoaded',
    function() {
        var ctx =
        document.getElementById('barangayChart').getContex
        t('2d');
        var barangayChart = new Chart(ctx, {
            type: 'bar',
            data: {
                labels: @json($barangayNames),
                datasets: [{
                    label: 'Number␣of␣Patients',
                    data: @json($patientCounts),
                        backgroundColor: 'rgba(75, 192, 192,
                        0.2)',
                        borderColor: 'rgba(75, 192, 192, 1)',
                        borderWidth: 1
                    }]
                },
                options: {
                    scales: {
                        y: {
                            beginAtZero: true
                        }
                    }
                }
            });
        });
    </script>


    @include('layouts.footer')


</body>
</html>
```

footer.blade.php

```


```

```
<footer class="sticky-footer text-color-emerald-900">

  <div class="container my-auto">
    <div class="copyright text-center my-auto">
      <span>{{ date('Y') }} Copyright    Rural Health Unit
      Management System 2024</span>
    </div>
  </div>
</footer>
```

guest.blade.php

```
@extends('layouts.app')
@section('content')
    <div name="header">
        <h2 class="font-semibold text-xl leading-tight bg-
        white p-4 top-0 shadow">
            {{ __(' Dashboard') }}
        </h2>
    </div>

    <div class="pb-5 pt-3">
            <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
                <div class="bg-white overflow-hidden shadow-
                sm sm:rounded-lg">
                    <div class="p-3 text-gray-900">
                        {{ __("You're logged in!") }}
                    </div>
                    </div>
                </div>
            </div>
@endsection
```

header.blade.php

```
<head>
    <meta charset=”utf−8”>
    <meta name=”viewport” content=”width=device−width,
␣␣␣␣initial−scale=1”>
    <meta name=”csrf−token” content=”{{␣csrf_token()␣}}”>

    <title>Mindcare</title>

    <!−− Custom CSS −−>
    <link href=”{{␣asset(’css/style.css’)␣}}”
    rel=”stylesheet”>

    <!−− Bootstrap CSS −−>
    <link rel=”stylesheet”
    href=”https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/cs
␣␣␣␣s/bootstrap.min.css”>
    <link
    href=”https://cdn.jsdelivr.net/npm/flowbite@2.4.1/dist/flo
␣␣␣␣wbite.min.css” rel=”stylesheet”>
    <link rel=”stylesheet”
    href=”https://cdn.datatables.net/1.10.24/css/jquery.dataTa
```

```
␣␣␣␣bles.min.css">
    <link rel="stylesheet"
    href="https://cdn.datatables.net/buttons/2.2.2/css/buttons
␣␣␣␣.dataTables.min.css">
    <link rel="stylesheet" type="text/css"
    href="https://cdn.datatables.net/1.11.5/css/jquery.dataTab
␣␣␣␣les.min.css">

    <!-- Font Awesome -->
    <link href="{{␣asset('sd-admin/vendor/fontawesome-
␣␣␣␣free/css/all.min.css')␣}}" rel="stylesheet"
    type="text/css">
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-
␣␣␣␣awesome/6.0.0-beta3/css/all.min.css" rel="stylesheet">

    <!-- SB Admin CSS -->
    <link href="{{␣asset('sd-admin/css/sb-admin-2.min.css')
␣␣␣␣}}" rel="stylesheet">
    <link href="{{␣asset('sd-
␣␣␣␣admin/vendor/datatables/dataTables.bootstrap4.min.css')
␣␣␣␣}}" rel="stylesheet">

    <!-- Custom styles for this page -->
    <link href="{{␣asset('sd-
␣␣␣␣admin/vendor/datatables/dataTables.bootstrap4.min.css')}}"
    rel="stylesheet">
    <link href="{{␣asset('sd-admin/vendor/fontawesome-
    free/css/all.min.css') }}” rel=”stylesheet”
        type=”text/css”>

        <!-- Google Fonts -->
        <link rel=”preconnect” href=”https://fonts.bunny.net”>
        <link href=”https://fonts.bunny.net/css?
    family=figtree:400,500,600&display=swap” rel=”stylesheet”>
        <link href=”https://fonts.googleapis.com/css?
    family=Nunito:200,200i,300,300i,400,400i,600,600i,700,700i
    ,800,800i,900,900i” rel=”stylesheet”>
         <!-- DataTables CSS -->
         <link rel=”stylesheet”
          href=”https://cdn.datatables.net/1.10.24/css/jquery.dataT
     ables.min.css”>
        <link rel=”stylesheet”
        href=”https://cdn.datatables.net/buttons/1.7.1/css/buttons
    .dataTables.min.css”>

</head>
```

navigation.blade.php

```


            <!-- Content Wrapper -->
            <div id=”content-wrapper” class=”d-flex flex-column”>

                <!-- Main Content -->
            <div id=”content”>

                <!-- Topbar -->
                <nav class=”navbar navbar-expand navbar-light topbar s

                      <!-- Sidebar Toggle (Topbar) -->
                    <button id=”sidebarToggleTop” class=”btn btn-link
                        <i class=”fa fa-bars”></i>
                    </button>

                    <div class=”max-auto max-w-7xl px-2 sm:px-6 lg:px-
                        <div class=”relative flex h-16” >
                            <div class=”flex”>
                                <!-- Logo -->
                                <div class=”shrink-0 flex items-cente
                                    <img src=”{{ asset(’img/369166592.
al=”Mindcare”>
                                </div>
                            </div>
                        </div>
                    </div>

                    <!-- Topbar Navbar -->
                    <ul class=”navbar-nav ml-auto”>
                    <!-- Notification Bell -->
                        <li class=”nav-item dropdown no-arrow mx-1”>
                            <a class=”nav-link dropdown-toggle” href=”
```

```
    <i class="fas fa-bell fa-fw"></i>
    @if (Auth::check())
        <span class="badge badge-danger ba
    @endif

</a>

<div class="dropdown-list dropdown-menu dr
    <h6 class="dropdown-header">Notificati
    @auth
    @forelse(Auth::user()->unreadNotificat
        <div class="dropdown-item d-flex a
            <div class="mr-3">
                <div class="icon-circle bg
                    <i class="fas fa-file-
                </div>
            </div>
            <div>
                <div class="small text-gra
                <span class="font-weight-
                @if(isset($notification->
                    <ul>
                        @foreach($notifica
                            <li>Added: {{
                        @endforeach
                        @foreach($notifica
                            <li>Removed:
```

```
                            @endforeach
                            @if(isset($notific
                                <li>
                                    Appointme
                                    <br>Appoi
                                    <br>Statu
                                    <br> {{ $
                                </li>
                            @endif
                        </ul>
                    @endif
                    <form method="POST" actio
                        @csrf
                        <button type="submit"
                    </form>
                </div>
            </div>
        @empty
            <div class="dropdown-item text-ce
        @endforelse
        @endauth
    </div>

</li>


<div class="topbar-divider d-none d-sm-block">
```

```

                        <li class=”nav–item dropdown no–arrow”>
                            <a class=”nav–link dropdown–toggle text–w
                                data–toggle=”dropdown” aria–haspopup=”
                                <div>{{ auth()–>check() ? auth()–>user
                            <div class=”ms–1”>
                            <svg xmlns=”http://www.w3.org/2000/svg” w
                                <path d=”M0 2a2 2 0 0 1 2–

                                2h12a2 2 0 0 1 2 2v12a2 2 0 0
                                1–2 2H2a2 2 0 0 1–2–2zm4
                                4a.5.5 0 0 0 –.374.832l4
                                4.5a.5.5 0 0 0 .748 0l4–
                                4.5A.5.5 0 0 0 12 6z”/>
                            </svg>
                            </div>
                            </a>
                            <!–– Dropdown – User Information –
                            –>
                            <div class=”dropdown–menu
                            dropdown–menu–right shadow
                            animated––grow–in”
                                aria–
                                labelledby=”userDropdown”>
                                <a class=”dropdown–item”
                                href=”{{ rout
                                e(’profile.edit’) }}”>
```

```
                                   <i class="fas fa-user fa-
                                   sm fa-fw mr-2 text-gray-
                                   400"></i>
                                   Profile
                               </a>

                               <div class="dropdown-divider">
                               </div>

                                   <form method="POST"
                                   action="{{ rout
                                   e('logout') }}">
                                       @csrf
                                       <x-dropdown-link
                                       :href="route('logout')
                                       "

                                           onclick="event.pre
                                           ventDefault();
                                           this.closest('form
                                           ').submit();">
                                       <i class="fas fa-sign-
                                       out-alt fa-sm fa-fw
                                       mr-2 text-gray-400">
                                       </i>
                                           {{ __('Log Out')
                                           }}
```

```
                            </x-dropdown-link>
                        </form>
                already</div>
                </div>
            </li>

        </ul>

    </nav>
    <!-- End of Topbar -->

 @push('scripts')

    <script>
    document.querySelectorAll('#notification-dropdown a').forEach(item => {
            item.addEventListener('click',
            function (event) {
                event.preventDefault();
                fetch(this.href, {
                    method: 'POST',
                    headers: {
                        'X-CSRF-TOKEN': '{{ csrf_token() }}',
                    },
                })

```

```
                            .then(() => location.reload());
                            // Reload page to reflect changes
                        });
                    });

                    setInterval(() => {
                        fetch('/notifications/unread-count')
                            .then(response => response.json())
                            .then(data => {
                                const badge =
                                document.querySelector('.badge
                                -counter');
                                badge.textContent =
                                data.unreadCount > 99 ? '99+'
                                : data.unreadCount;
                            });
                    }, 5000);

                </script>
                <script>
                    function markNotificationAsRead(url) {
                        fetch(url, {
                            method: 'POST',
                            headers: {
                                'X-CSRF-TOKEN': '{{
                                csrf_token() }}',
                                'Content-Type':
                                'application/json'
                            }
                        }).then(response => {
                            if (response.ok) {
                                window.location.reload();
                            } else {
                                alert('Failed to mark
                                notification as read.');
                            }
                        }).catch(error =>
                        console.error('Error:', error));
                    }
                </script>



            @endpush
```

sidebar.blade.php

```

@if(Auth::check() && Auth::user()->type == 'admin')
 <div id="sidebar">

   <!-- Sidebar -->
     <ul class="navbar-nav sidebar sidebar-dark accordion sm:w-
     16 flex-shrink-0" style="height: 100%; background-color:
     #347474;" id="accordionSidebar">
```

```

        <!-- Sidebar - Brand -->
        <a class="sidebar-brand d-flex align-items-center
        justify-content-center" href="{{ route('admin.home')
        }}">
            <div class="sidebar-brand-text mx-
            3">Mindcare</div>
        </a>

        <!-- Divider -->
        <hr class="sidebar-divider ">

        <!-- Nav Item - Dashboard -->
        <li class="nav-item active">
            <a class="nav-link" href="{{ route('admin.home')
            }}">
                <i class="fas fa-fw fa-tachometer-alt"></i>
                <span>Dashboard</span>
            </a>
        </li>

        <!-- Divider -->
        <hr class="sidebar-divider">

        <!-- Heading -->
        <div class="sidebar-heading">
            Record Management
```

```
</div>

<!-- User -->

    <!-- User Management with Collapse Menu -->
    <li class="nav-item">
        <a class="nav-link collapsed" href="#" data-
        toggle="collapse" data-
        target="#collapseUsers" aria-expanded="false"
        aria-controls="collapseUsers">
            <i class="fas fa-users"></i>
            <span>User Management</span>
        </a>
        <div id="collapseUsers" class="collapse" aria-
        labelledby="headingUsers" data-
        parent="#accordionSidebar">
            <div class="bg-white py-2 collapse-inner
                                   rounded">
                <a class="collapse-item" href="{{
                                       route('users') }}"><i class="fas fa-
                                       user"></i> Active Users</a>
                <a class="collapse-item" href="{{
                                       route('archived.users.index') }}"><i
                 class="fas fa-archive"></i> Archived
                 Users</a>
            </div>
        </div>
        </li>


        <!-- Patient Management with Collapse Menu -->
        <li class="nav-item">
            <a class="nav-link collapsed" href="#" data-
            toggle="collapse" data-
            target="#collapsePatients" aria-
            expanded="false" aria-
            controls="collapsePatients">
                <i class="fas fa-user-injured"></i>
                <span>Patient Management</span>
            </a>
            <div id="collapsePatients" class="collapse"
            aria-labelledby="headingPatients" data-
            parent="#accordionSidebar">
                <div class="bg-white py-2 collapse-inner
                rounded">
                    <a class="collapse-item" href="{{
                    route('patients') }}"><i class="fas
                    fa-user"></i> Patients</a>
                    <a class="collapse-item" href="{{
                    route('patients.archive.index') }}">
                    <i class="fas fa-archive"></i>
                    Archived Patients</a>
                </div>
            </div>
        </li>


        <!-- Therapist -->
    <!-- Therapist Management with Collapse Menu -->
    <li class="nav-item">
        <a class="nav-link collapsed" href="#" data-
        toggle="collapse" data-
        target="#collapseTherapists" aria-
        expanded="false" aria-
        controls="collapseTherapists">
            <i class="fas fa-user-md"></i>
            <span>Health Personels</span>
        </a>
        <div id="collapseTherapists" class="collapse"
        aria-labelledby="headingTherapists" data-
        parent="#accordionSidebar">
            <div class="bg-white py-2 collapse-inner
            rounded">
                <a class="collapse-item" href="{{
                route('therapists') }}"><i class="fas fa-
                user-md"></i> Active Health Personal</a>
                <a class="collapse-item" href="{{
                route('therapists.archived') }}"><i
                class="fas fa-archive"></i> Archived
                Personel</a>
            </div>
        </div>
    </li>

    <!-- Divider -->
    <hr class="sidebar-divider">

    <!-- Appointments Booking -->
    <li class="nav-item">
        <a class="nav-link collapsed" href="#" data-
        toggle="collapse" data-
        target="#collapseAppointments" aria-
        expanded="false" aria-
        controls="collapseAppointments">
            <i class="fas fa-calendar-alt"></i>
            <span>Appointments Booking</span>
        </a>
        <div id="collapseAppointments" class="collapse"
        aria-labelledby="headingAppointments" data-
        parent="#accordionSidebar">
            <div class="bg-white py-2 collapse-inner
            rounded">
                <a class="collapse-item" href="{{
                route('calendar.index') }}"><i class="fas
                fa-calendar-day"></i> Appointment
                Booking</a>
                <a class="collapse-item" href="{{
                route('calendar.allBooked') }}"><i
                class="fas fa-calendar-plus"></i> Booked
                Appointment</a>
            </div>
        </div>
    </li>

        <!-- Medicine Inventory -->
    <li class="nav-item">
        <a class="nav-link" href="{{
        route('inventory.index') }}">
            <i class="fas fa-box"></i>
            <span>Medicine Inventory</span>
        </a>
    </li>


    <!-- Divider -->
    <hr class="sidebar-divider d-none d-md-block">
      <!-- Sidebar Toggler (Sidebar) -->
        <div class="text-center d-none d-md-inline">
            <button class="rounded-circle border-0
            md:hidden p-2 fixed top-0 left-0 z-10"
            id="sidebarToggle"></button>
        </div>


    </ul>
    <!–– End of Sidebar —>
</div>

@endif


@if(Auth::check() && Auth::user()–>type == 'therapist')

<!–– Page Wrapper —>
<div id=”wrapper”>

<!–– Sidebar —>
<ul class=”navbar–nav␣sidebar␣sidebar–dark␣accordion␣sm:w–
␣␣16␣flex–shrink–0” style=”height:␣100%;␣background–color:
␣␣#347474;” id=”accordionSidebar”>

<!–– Sidebar – Brand —>
<a class=”sidebar–brand␣d–flex␣align–items–center␣justify–
␣␣␣␣content–center” href=”{{␣route('therapist.home')␣}}”>
    <div class=”sidebar–brand–text␣mx–3”>Mindcare</div>
</a>

    <!–– Divider —>
    <hr class=”sidebar–divider␣my–0”>

    <!–– Nav Item – Dashboard —>
    <li class=”nav–item␣active”>
```

```
            <a class="nav-link" href="{{
            route('therapist.home') }}">
                <i class="fas fa-fw fa-tachometer-alt"></i>
                <span>Dashboard</span>
            </a>
        </li>

        <!-- Divider -->
        <hr class="sidebar-divider">

        <!-- Heading -->
        <div class="sidebar-heading">
            Record Management
        </div>

            <!-- Patient Management with Collapse Menu -->
            <li class="nav-item">
                <a class="nav-link collapsed" href="#" data-
                toggle="collapse" data-
                target="#collapsePatients" aria-
                expanded="false" aria-
                controls="collapsePatients">
                    <i class="fas fa-user-injured"></i>
                    <span>Patient Management</span>
                </a>
                <div id="collapsePatients" class="collapse"
                aria-labelledby="headingPatients" data-
                parent="#accordionSidebar">
                    <div class="bg-white py-2 collapse-inner
                    rounded">
                        <a class="collapse-item" href="{{
                        route('patients') }}"><i class="fas
                        fa-user"></i> Patients</a>
                        <a class="collapse-item" href="{{
                        route('patients.archive.index') }}">
                        <i class="fas fa-archive"></i>
                        Archived Patients</a>
                    </div>
                </div>
            </li>

        <!-- Divider -->
        <hr class="sidebar-divider">

         <!-- Appointments Booking -->
         <li class="nav-item">
            <a class="nav-link collapsed" href="#" data-
            toggle="collapse" data-
            target="#collapseAppointments" aria-
            expanded="false" aria-
            controls="collapseAppointments">
                <i class="fas fa-calendar-alt"></i>
                <span>Appointments Booking</span>
            </a>
            <div id="collapseAppointments" class="collapse"
            aria-labelledby="headingAppointments" data-
            parent="#accordionSidebar">
                <div class="bg-white py-2 collapse-inner
                rounded">
                    <a class="collapse-item" href="{{
                    route('calendar.index') }}"><i class="fas
                    fa-calendar-day"></i> Appointment
                    Booking</a>
                    <a class="collapse-item" href="{{
                    route('calendar.allBooked') }}"><i
                    class="fas fa-calendar-plus"></i> Booked
                    Appointment</a>
                </div>
            </div>
        </li>

            <!-- Medicine Inventory -->
        <li class="nav-item">
            <a class="nav-link" href="{{
            route('inventory.index') }}">
                <i class="fas fa-box"></i>
                <span>Medicine Inventory</span>
            </a>
        </li>


        <!-- Divider -->
        <hr class=”sidebar-divider␣d-none␣d-md-block”>
          <!-- Sidebar Toggler (Sidebar) -->
            <div class=”text-center␣d-none␣d-md-inline”>
                <button class=”rounded-circle␣border-0”
                id=”sidebarToggle”></button>
            </div>


        </ul>
        <!-- End of Sidebar -->
</div>


@endif
@if(Auth::check() && Auth::user()->type == 'patient')

<div id=”sidebar”>
    <!-- Sidebar -->
    <ul class=”navbar-nav␣sidebar␣sidebar-dark␣accordion␣sm:w-
␣␣␣␣16␣flex-shrink-0” style=”height:␣100%;␣background-color:
␣␣␣␣#347474;” id=”accordionSidebar”>

    <!-- Sidebar - Brand -->
    <a class=”sidebar-brand␣d-flex␣align-items-center␣justify-
␣␣␣␣content-center” href=”{{␣route('admin.home')␣}}”>
        <div class=”sidebar-brand-text␣mx-3”>Mindcare</div>
    </a>


        <!-- Divider -->
        <hr class="sidebar-divider">

            <!-- Nav Item - Dashboard -->
            <li class="nav-item">
                <a class="nav-link" href="{{ route('home')
                }}">
                    <i class="fas fa-fw fa-tachometer-alt">
                    </i>
                    <span>Dashboard</span>
                </a>
            </li>
             <!-- Divider -->
             <hr class="sidebar-divider">

             <!-- Appointments Booking -->
         <!-- Appointments Booking -->
         <li class="nav-item">
            <a class="nav-link collapsed" href="#" data-
            toggle="collapse" data-
            target="#collapseAppointments" aria-
            expanded="false" aria-
            controls="collapseAppointments">
                <i class="fas fa-calendar-alt"></i>
```

```
                    <span>Appointments Booking</span>
                </a>
                <div id="collapseAppointments" class="collapse"
                aria-labelledby="headingAppointments" data-
                parent="#accordionSidebar">
                    <div class="bg-white py-2 collapse-inner
                    rounded">
                        <a class="collapse-item" href="{{
                        route('calendar.index') }}"><i class="fas
                        fa-calendar-day"></i> Appointment
                        Booking</a>
                        <a class="collapse-item" href="{{
                        route('calendar.allBooked') }}"><i
                        class="fas fa-calendar-plus"></i> Booked
                        Appointments</a>
                    </div>
                </div>
            </li>

                        <!-- Divider -->
                <hr class="sidebar-divider d-none d-md-block">

                <!-- Sidebar Toggler (Sidebar) -->
                <div class="text-center d-none d-md-inline">
                    <button class="rounded-circle border-0"
                    id="sidebarToggle"></button>
                </div>
```

```

        </ul>
    </ul>
</div>
    <!-- End of Sidebar -->

@endif
```

Appendix A :Validation Questionnaires

Appendix A :Communication Letters