DESIGN AND IMPLEMENTATION OFA MEDICATION REMINDER SYSTEM FOR THE ELDERLY

BY

OSAIGBOVO ESEOSA

PSC1606872

DEPARTMENT OF COMPUTER SCIENCE,

FACULTY OF PHYSICAL SCIENCES,

UNIVERSITY OF BENIN,

BENIN CITY,

EDO STATE, NIGERIA.

FEBRUARY 2024

DESIGN AND IMPLEMENTATION OFA MEDICATION REMINDER SYSTEM FOR THE ELDERLY

BY

OSAIGBOVO ESEOSA

PSC1606872

A PROJECT REPORT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF PHYSICAL SCIENCES, UNIVERSITY OF BENIN, BENIN CITY

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF A BACHELOR OF SCIENCE (B.Sc.) DEGREE IN COMPUTER SCIENCE

FEBRUARY 2024

CERTIFICATION

This is to certify that this project work was carried out by **OSAIGBOVOESEOSA** with Matriculation Number **PSC1606872** under my supervision. It is adequate and satisfactory, both in scope and content, for the award of Bachelor of Science (B.sc) Degree in Computer Science of the University of Benin

PROF.V.V.N AKWUKWUMA
Project Supervisor

DATE

APPROVAL

This project work is hereby approved in partial fulfilment of the requirements for the award of Bachelor of Science (B.Sc.) Degree in Computer Science from the University of Benin.

PROF.V.V.NANKWUKWUMA **DATE**

Head of Department

DEDICATION

This project is dedicated to God Almighty for giving me the strength and wisdom to see it through to completion, and even throughout my stay in the UniversityofBenin (UNIBEN). It is also dedicated to my parents; MR and MRS PHILIP IKHUENBOR for their love, support and guidance throughout my academic journey.

ACKNOWLEDGEMENT

My utmost acknowledgement goes to God Almighty for giving me the strength, wisdom and direction throughout my academic journey. I would like to express my gratitude to my project supervisor PROFF V.V.N ANKWUKWUMA for making this project smooth and easy for me.

I would like to express my sincere gratitude to everyone who contributed to the successful completion of this project. First and foremost, I am deeply indebted to my parents, Mr. and Mrs. Philip Ikhuenbor, for their unwavering love, support, and guidance throughout this journey. Their belief in me has been a constant source of strength.

I am also incredibly grateful to my brother, Peter Osaigbovo, for his generous financial assistance and invaluable life advice. His support has been instrumental in overcoming challenges and keeping me focused.

To my extended family and friends, both near and far, thank you for your incredible support, both in cash and in kind. Your encouragement and assistance have been invaluable, and I am truly blessed to have you all in my life. I am deeply appreciative of your belief in me and your contributions to this project.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Many elderly individuals face challenges in remembering to take their medications as prescribed,which often leads to health problems.

This paper presents the design and implementation of a medical reminder system tailored specifically for the elderly. This system aims to address the challenges faced by the elderly population in adhering to complex medication regimens and timely appointments. The proposed system leverages a combination of hardware and software components to provide timely reminders, medication management features, and emergency contact capabilities. The hardware components include a user-friendly interface, a robust communication module, and sensors for monitoring vital signs. The software component encompasses a user-friendly interface, a comprehensive medication management system, and a secure communication platform. The system is designed to be easily customizable to accommodate individual needs and preferences, ensuring its adaptability to diverse elderly populations. The evaluation of the system demonstrates its effectiveness in improving medication adherence, reducing medication errors, and enhancing overall healthcare management for the elderly.

CHAPTER ONE

INTRODUCTION

1.0 Introduction

Most people today require medicines that were not available a few years ago, and the reason for this is that diseases are becoming more prevalent. Some diseases are only temporary, while others are chronic and life-threatening, integrating with the human body in such a way that they cannot leave the body and multiply rapidly. Many people miss their medication, with more than 80% of patients occasionally forgetting to take a dose, leading to consequences down the road. Almost 25% of heart fatalities can be avoided by taking medicine as directed. Work-related obligations, forgetfulness, or confusion can lead to patients or their loved ones failing to take prescribed medication. For instance, administering medication can take around 2 minutes, but people may forget the dosage or the medication's name. The COVID vaccination provides a recent example where some individuals failed to receive the second dosage of the vaccine at the scheduled time. The COVID pandemic has also led to hospitals being busy treating patients, which could result in other patients using the wrong medications. Hospital care itself can be dangerous, with nearly ten percent of patients experiencing an injury and 1.4 million people worldwide contracting illnesses from hospitals. The prevalence of diseases and the consequences of not taking medication have led to a decrease in human life expectancy. To overcome this, medicine must be taken on a regular and large scale, and patients must seek the advice of a doctor for instructions on how to take the medication and how to remember the new schedule of medicine when the prescription changes. To address this issue, my project aims to build a medication reminder app,the app will help to remind the patients to always take their medications.

1.1 Background study

Recent studies have investigated various approaches to improving medication adherence. Kamal et al (2015) looked at the effectiveness of mobile technology in the form of customized Short Messaging Service (SMS) reminders. The study was conducted in resource-poor areas in Pakistan and involved a sample of 200 participants. The results showed that the intervention group, who received personalized medication reminders

and health information by SMS, had a higher mean medication score and a lower mean diastolic blood pressure compared to the control group who only received usual care. Adler et Al (2017) also conducted a systematic review to assess the impact of text messaging on adherence to treatment for patients with established arterial occlusive events. The study included seven trials with 1310 participants but found insufficient evidence to draw conclusions on the effectiveness of text messaging interventions for medication adherence. The authors noted that the evidence was of very low quality and recommended the need for high-quality, randomized trials, particularly in low- and middle-income countries.

In a study conducted by Choudry et al (2017), the effects of low-cost medication reminder devices on medication adherence were compared. The study involved 53,480 participants who were taking 1 to 3 oral medications for long-term use and who were suboptimally adherent to their prescribed therapies. The participants were divided into two groups: those taking medications for cardiovascular or other non-depression chronic conditions, and those taking antidepressants. Participants were randomly assigned to receive either a pill bottle strip with toggles, digital timer cap, or standard pillbox, or to receive neither notification nor a device (control group). A formative study conducted by Dorothy et al aimed to investigate the use of Wisepill evriMED Medication Event Reminder Monitor (MERM) among outpatients with drug susceptible pulmonary tuberculosis in Thanh Hoa, a rural province in northern Viet Nam. The study enrolled 20 patients in a randomized controlled trial, where half of the participants received daily alerts and scheduled dosing history reviews while the other half received the device without an alert. The study was conducted through in-depth interviews with the participants. In a study by Mohamed et al (2016), the importance of the rapid growth of mobile phone usage in low and middle-income countries was acknowledged as a valuable tool for public health programs to access patients. The researchers aimed to evaluate the impact of a two-way SMS reminder system called Zindagi SMS on the treatment success of patients with drug-sensitive tuberculosis. The study was conducted as a randomized controlled trial in Karachi, Pakistan, with 2,207 participants being randomly assigned to either the Zindagi SMS group or the control group. The primary outcome measured was the clinically recorded treatment success. In a study by hayakawa et al the feasibility of a smartphone-based medication self-management

system (SMSS) was investigated. The aim of the study was to design, develop and demonstrate the feasibility of the SMSS with real-time medication monitoring to help patients manage their medication regimen. The study was based on the results of interviews with 116 patients and the SMSS was designed and developed to offer two main functions to patients through their smartphones: (1) storage and provision of accurate medication history and medication-taking records; and (2) provision of reminders to take medication when patients forget. Despite its effectiveness, the SMS-based medication time reminder system still requires some improvements. One of the key challenges is the cost associated with sending multiple SMS reminders every day. A patient who requires reminders three times a day could incur a significant monthly cost from the network provider. Additionally, if the credit on the device is depleted, the patient may miss notifications. Another issue with the SMS-based approach is that the patient may not be aware of the alert if they are not with their phone at the time of delivery.

In light of these limitations, this project proposes the use of a notification/alarm system.. This is an improvement upon the work done by others and it is expected to be more cost-effective as well as more effective, as the ringing period is usually longer than the message alert notification period. This builds upon the work done by Bhati et al (2017) in integrating prescription schedules into the medication time reminder system.

1.2 Problem of the study

The elderly population often struggles with medication adherence due to various factors such as cognitive decline, polypharmacy, and complex dosing regimens. This non-adherence can lead to serious health consequences, increased healthcare costs, and decreased quality of life. Traditional methods of medication reminders, such as pillboxes and paper calendars, are often insufficient in addressing the specific needs of the elderly.

Therefore, there is a need for a user-friendly and effective medical reminder application that can:

* Provide timely reminders: Deliver accurate and timely reminders for medication intake, appointments, and other health-related tasks.

* Track medication adherence: Monitor medication intake and generate reports to assess adherence patterns.

* Offer personalized features: Allow users to customize reminders, set alarms, and receive notifications based on their specific needs and preferences.

* Integrate with healthcare providers: Facilitate communication between patients and healthcare providers by sharing adherence data and generating reports.

* Ensure user privacy and security: Protect sensitive health information through robust security measures.

By addressing these challenges, the development of this medication reminder app can significantly improve medication adherence, reduce medication errors, and enhance the overall health and well-being of the elderly population.

1.3 Aim and Objectives

The aim of this research project is designing a medication reminder app for the sick and elderly. The project aims to achieve the following objectives:

i. To identify the specific needs and challenges faced by the elderly populations in managing medication regimens
ii. To develop an easy-to-use mobile application with features such as personalized medication reminders.

1.4 Significance of this study

It is important to know that the significance of this study lies in its potential to address a critical healthcare issue affecting the elderly population. By developing a user friendly and effective medical reminder app,this research aims to improve medication adherence,enhance quality of life, and reduce healthcare costs.

By addressing these significant issues,this study has the potential to make a positive impact on the health and well-being of the elderly population.

1.5 Scope of study

The scope of this study is focused on the design and development of a mobile application specifically tailored to the needs of the elderly population. The app will primarily focus on the following core functionalities:

1. Personalized Medication Reminders
2. User-Friendly Interface

By focusing on these key areas, the app aims to provide a comprehensive solution for improving medication adherence and enhancing the overall health and well-being of the elderly population.

CHAPTER TWO

LITERATURE REVIEW

This chapter gives an overview of the current state of research and development in medical reminder systems for the elderly, highlighting their potential benefits, limitations, and future directions.

Medication non-adherence is a significant public health problem with substantial consequences for the elderly population. Non-adherence can lead to poor health outcomes, increased hospitalization rates, and higher healthcare costs. Several factors contribute to medication non-adherence among the elderly, including:

2.1 The challenge of medication non-adherence in the elderly; An overview

Medication non-adherence is a significant public health problem with substantial consequences for the elderly population. Non-adherence can lead to poor health outcomes, increased hospitalization rates, and higher healthcare costs. Several factors contribute to medication non-adherence.

2.1.1 The prevalence of medication non-adherence among the elderly

Medication non-adherence is a significant public health concern, particularly among the elderly population. This issue is complex and multifaceted, influenced by various factors, including cognitive decline, polypharmacy, and socio-economic conditions.

Several studies have investigated the prevalence of medication non-adherence among the elderly. A systematic review by Sabaté et al. (2001) found that non-adherence rates can vary widely, ranging from 20% to 79%, depending on the specific medication and patient population.

More recent studies have reported similar findings. For example, a study by DiMatteo et al. (2000) found that non-adherence rates among older adults with chronic conditions can exceed 50%. Similarly, a study by Haynes et al. (2008) reported that non-adherence rates for hypertension and diabetes can range from 30% to 50%.

2.1.2 Factors Contributing to Non-Adherence

A number of factors contribute to medication non-adherence in the elderly population. These include:

1. **Cognitive impairment:** Age-related cognitive decline can impair memory and decision-making abilities, making it difficult for older adults to remember complex medication regimens.
2. **Polypharmacy:** The use of multiple medications increases the risk of medication errors, interactions, and adverse effects.

3. **Complex regimens:** Complicated dosing schedules, multiple daily doses, and specific timing requirements can be challenging to follow.
4. **Sensory impairments:** Visual and hearing impairments can hinder the ability of older adults to read labels, understand instructions, and hear alarms or reminders.
5. **Socioeconomic factors:** Low income, limited access to healthcare, and lack of health literacy can contribute to non-adherence.
6. **Adverse drug effects:** Side effects from medications can lead to reduced adherence.

2.1.3 Impact of non-adherence

Medication non-adherence can have serious consequences for the health and well-being of the elderly population. When people fail to adhere to their medications, it results to consequences. These consequences include:

1. **Increased morbidity and mortality**: Non-adherence can lead to worsening of chronic conditions, increased hospitalizations, and increased mortality rates.
2. **Increased healthcare costs**: Non-adherence can result in higher healthcare costs due to increased hospitalizations, emergency room visits, and other medical services.
3. **Reduced quality of life**: Non-adherence can negatively impact quality of life by causing physical symptoms, emotional distress, and functional limitations.

2.2 Traditional methods of medication adherence

Traditional methods for improving medication adherence have been employed for decades, often relying on patient education, counselling, and simple organizational tools. While these methods may not be as technologically advanced as contemporary solutions, they remain effective when implemented correctly.

2.2.1 Patient Education and Counselling

1. **Clear and Concise Instructions**: Healthcare providers should provide clear and concise instructions about medication regimens, including dosage, timing, and potential side effects.
2. **Tailored Education**: Tailoring education to the individual patient's needs, including their literacy level and cultural background, can enhance understanding and adherence.
3. **Regular Follow-up**: Regular follow-up appointments with healthcare providers allow for monitoring adherence, addressing any concerns, and reinforcing medication instructions.

2.2.2 Organizational Tools

1. **Pill Organizers**: A Traditional Tool for Medication Adherence

   Pill organizers are a common traditional tool used to improve medication adherence, particularly among the elderly. These devices come in various shapes and sizes,

often featuring compartments for each day of the week, divided into sections for morning, afternoon, and evening doses.

How Pill Organizers Work:

- Medication Sorting: Patients or caregivers can sort medications into the appropriate compartments based on the prescribed dosing schedule.
- Visual Reminder: The physical act of filling the pill organizer serves as a visual reminder of the medication regimen.
- Portable Convenience: Pill organizers are often compact and portable, making it easy for individuals to carry their medications with them.

Benefits of Pill Organizers:

- **Improved Adherence:** By pre-sorting medications, pill organizers can reduce the risk of missed doses or accidental overdoses.
- **Enhanced Organization**: They provide a clear and organized system for managing medications, making it easier to track intake.
- **Reduced Cognitive Burden:** Pill organizers can alleviate the cognitive load associated with remembering complex medication schedules, especially for individuals with cognitive impairment.

Limitations of Pill Organizers:

- **Reliance on User Memory:** While pill organizers can assist with medication timing, they still rely on users to remember to take the medications from the organizer.
- **Limited Flexibility:** They may not be suitable for complex dosing schedules or medications that need to be taken at irregular intervals.
- **Risk of Contamination:** If not cleaned regularly, pill organizers can become a breeding ground for bacteria and mold.

While pill organizers have been a valuable tool for improving medication adherence, it's important to recognize their limitations and consider combining them with other strategies, such as technology-based solutions, to maximize their effectiveness.

2. **Medication Calendars:A Visual Aid for Adherence**;
   Medication calendars are another traditional tool used to improve medication adherence. These calendars typically consist of a grid with days of the week and times of day, allowing patients to visually track their medication intake.

   How Medication Calendars Work:

   - **Visual Tracking:** Patients can mark off each dose taken on the calendar, creating a visual record of their adherence.
   - **Reminder System:** The calendar serves as a visual reminder, prompting patients to take their medications at the appropriate times.

- **Easy-to-Use:** Medication calendars are simple to use and require minimal technical skills.

Benefits of Medication Calendars:

- **Improved Adherence:** By providing a visual representation of the medication schedule, calendars can help patients stay on track.
- **Enhanced Organization:** They can help patients organize their medication regimen and avoid confusion.
- **Flexibility:** Medication calendars can be customized to accommodate various dosing schedules and medication types.

Limitations of Medication Calendars:

- **Reliance on User Memory:** While calendars can serve as visual reminders, they still rely on users to remember to check the calendar and take their medications.
- **Limited Flexibility:** They may not be suitable for complex dosing schedules or medications that need to be taken at irregular intervals.
- **Potential for Errors:** If not used consistently, medication calendars can lead to errors in tracking medication intake.

To maximize the effectiveness of medication calendars, it is important to use them in conjunction with other strategies, such as setting alarms, using pill organizers, and consulting with healthcare providers.

3. **Alarms and Timers: A Simple Yet Effective Tool for Medication Adherence;**

Alarms and timers have long been used as a straightforward method to improve medication adherence. By setting specific times for medication intake, these devices can serve as reliable reminders, particularly for individuals with busy schedules or memory impairments.

How Alarms and Timers Work:

- **Time-Based Reminders:** Users can set alarms on their phones, watches, or dedicated alarm clocks to coincide with their medication schedule.
- **Auditory and Visual Cues:** Alarms can be customized to include auditory cues (e.g., ringing, beeping) and visual cues (e.g., flashing lights) to maximize their effectiveness.
- **Flexibility:** Alarms and timers can be easily adjusted to accommodate changes in medication schedules or lifestyle.

Benefits of Alarms and Timers:

- Improved Adherence: Timely reminders can help individuals remember to take their medications as prescribed.

* Reduced Reliance on Memory: Alarms and timers can alleviate the cognitive burden of remembering complex medication schedules.
* Accessibility: Most people have access to alarm and timer functions on their phones or other devices.

Limitations of Alarms and Timers:

* Reliance on Device Function: If a device malfunctions or is not charged, the alarm may not go off, leading to missed doses.
* Potential for Alarm Fatigue: Excessive use of alarms can lead to alarm fatigue, reducing their effectiveness.
* Lack of Personalization: Alarms and timers may not be as personalized as other methods, such as mobile apps, which can provide tailored reminders and educational content.

While alarms and timers are a simple and accessible tool for improving medication adherence, it is essential to use them in conjunction with other strategies to maximize their effectiveness. For example, combining alarms with pill organizers or medication calendars can create a comprehensive approach to medication management.

2.2.3 Limitations of Traditional Methods

While traditional methods can be effective, they have limitations:

1. Reliance on Memory: These methods rely on patients' memory, which can be impaired by age, cognitive decline, or other factors.
2. Lack of Real-Time Monitoring: Traditional methods do not provide real-time monitoring of medication adherence, making it difficult to identify and address issues promptly.
3. Limited Flexibility: These methods may not be suitable for complex dosing schedules or medications that need to be taken at irregular intervals.
4. Potential for Human Error: Human error, such as forgetting to set an alarm or misplacing a pill organizer, can lead to missed doses or medication errors.
5. Limited Personalization: Traditional methods may not be as personalized as technology-based solutions, which can tailor reminders and interventions to individual needs.

   To overcome these limitations, it is essential to consider incorporating technology-based solutions into medication adherence strategies. By combining traditional methods with digital tools, healthcare providers can develop more effective and personalized approaches to improve medication adherence among the elderly population.

2.3 Technological interventions for medication adherence

Technology has revolutionized the way we manage our health, including medication adherence. By leveraging mobile applications, wearable devices, and smart home systems, individuals can improve their adherence to medication regimens, leading to better health outcomes.

2.3.1 Mobile Applications;

Mobile health (mHealth) applications have emerged as a powerful tool for improving medication adherence. These apps can provide personalized reminders, track medication intake, and offer educational resources.

* Personalized Reminders: mHealth apps can deliver timely reminders tailored to individual medication schedules. Users can set specific times for each medication, and the app will send notifications to their smartphone or smartwatch.

* Medication Tracking: Users can log their medication intake in the app, allowing for real-time monitoring of adherence patterns. This data can be shared with healthcare providers to track progress and identify areas for improvement.

* Educational Resources: Many mHealth apps provide educational information about medications, side effects, and interactions. This information can help users understand their treatment plans and make informed decisions about their health.

* Social Support: Some apps offer social features, allowing users to connect with others and share experiences. This can provide emotional support and motivation to stay on track with medication regimens.

2.3.2 Wearable Devices;

Wearable devices, such as smartwatches and fitness trackers, can be used to monitor vital signs and deliver medication reminders.

* Real-time Monitoring: Wearable devices can track heart rate, blood pressure, and other health metrics, allowing for early detection of potential health issues.

* Discreet Reminders: Reminders can be delivered discreetly via vibrations or silent notifications, minimizing disruptions to daily activities.

* Integration with mHealth Apps: Wearable devices can be integrated with mHealth apps to provide a comprehensive solution for medication management. Data from wearable devices can be synced with mHealth apps to provide a more complete picture of a user's health and adherence.

2.3.3 Smart Home Systems

Smart home systems can be used to automate tasks related to medication management, such as turning on lights or playing specific sounds to signal medication time.

* Automated Reminders: Smart speakers can be programmed to deliver voice-activated reminders, making it easier for users to stay on track.

* Environmental Cues: Smart home systems can create a supportive environment for medication adherence by adjusting lighting, temperature, or playing calming music.

* Integration with Other Devices: Smart home systems can integrate with other devices, such as smart pill dispensers or wearable devices, to create a seamless medication management experience.

By leveraging these technologies, individuals can improve their medication adherence, reduce the risk of adverse health outcomes, and enhance their overall quality of life.

2.4 Advantages and Disadvantages of Technological Solutions for Medication Adherence

While technology offers significant potential for improving medication adherence, it is essential to consider both the advantages and disadvantages of different solutions.

2.4.1 Mobile Applications

Advantages:

1. Personalization: Apps can be tailored to individual needs and preferences.
2. Accessibility: They are easily accessible through smartphones and tablets.
3. Real-time Monitoring: Users can track their adherence and identify areas for improvement.
4. Educational Resources: Apps can provide valuable information about medications and health conditions.

Disadvantages:

1. Reliance on Technology: Users must have access to a compatible device and reliable internet connectivity.
2. User Engagement: Maintaining long-term engagement with the app can be challenging.
3. Data Privacy Concerns: Users may have concerns about the privacy and security of their health data.

2.4.2 Wearable devices

Advantages;

1. Continuous Monitoring: Wearable devices can track vital signs and physical activity.
2. Discreet Reminders: Reminders can be delivered silently, minimizing disruptions.
3. Integration with mHealth Apps: Data from wearable devices can be integrated with mHealth apps to provide a comprehensive view of health.

Disadvantages:

1. Battery Life: Wearable devices may require frequent charging.
2. Comfort and Style: Some individuals may find wearable devices uncomfortable or unappealing.
3. Data Accuracy: The accuracy of data collected by wearable devices can vary.

2.4.3 Smart Home Systems

Advantages;

1. Automation: Smart home systems can automate tasks related to medication management.
2. Environmental Control: They can create a supportive environment for medication adherence by adjusting lighting, temperature, or playing calming music.
3. Integration with Other Devices: Smart home systems can integrate with other devices, such as smart pill dispensers or wearable devices, to create a seamless medication management experience.

Disadvantages:

1. Initial Cost: Smart home systems can be expensive to set up and maintain.
2. Technical Complexity: Some users may find the technology complex to use.
3. Reliance on Internet Connectivity: Smart home systems often rely on a stable internet connection.

To maximize the benefits of technology-based solutions, it is important to select the right tools for individual needs and preferences. Healthcare providers can play a crucial role in educating patients about the benefits of technology and helping them choose the best solutions for their specific needs.

2.5 Design principles for effective medical reminder systems:

Designing effective medical reminder systems requires careful consideration of several key principles to ensure their usability, accessibility and safety.

1. **User-Centered Design**:
    - **Intuitive Interface**: The system should have a user-friendly interface that is easy to navigate for both patients and healthcare providers.
    - **Accessibility**: Ensure the system is accessible to users with disabilities by incorporating features like screen readers and adjustable text sizes.
2. **Personalization**:
    - **Customizable Reminders**: Allow users to set personalized reminders based on their medication schedules, appointment times, and preferences.

- **Multiple Notification Channels**: Provide options for notifications via SMS, email, push notifications, and phone calls to cater to different user preferences.

3. **Security and Privacy**:
    - **Data Encryption**: Implement robust encryption methods to protect sensitive patient data.
    - **Compliance**: Ensure the system complies with relevant healthcare regulations and standards, such as HIPAA in the United States.
4. **Reliability and Accuracy**:
    - **Accurate Scheduling**: Ensure that reminders are sent at the correct times and that the system accurately tracks medication schedules and appointments.
    - **Redundancy**: Implement backup systems to ensure reminders are sent even if there are technical issues.
5. **Engagement and Motivation**:
    - **Interactive Features**: Include features that engage users, such as progress tracking, motivational messages, and rewards for adherence.
    - **Feedback Mechanism**: Provide a way for users to give feedback on the system's effectiveness and usability.
6. **Integration with Healthcare Systems**:
    - **Interoperability**: Ensure the system can integrate with existing electronic health records (EHR) and other healthcare systems to provide a seamless experience.
    - **Data Sharing**: Allow healthcare providers to access patient adherence data to monitor and adjust treatment plans as needed.
7. **Scalability and Flexibility**:
    - **Scalable Architecture**: Design the system to handle an increasing number of users and data without compromising performance.
    - **Customizable Modules**: Allow for customization to meet the specific needs of different healthcare providers and patient populations.
8. **Analytics and Reporting**:
    - **Data Analytics**: Incorporate analytics to track patient adherence, identify trends, and provide insights for healthcare providers.

- **Reporting Tools**: Provide tools for generating reports on patient adherence and system performance.

By adhering to these design principles, a medical reminder system can effectively improve patient adherence to medication schedules and appointments, ultimately enhancing healthcare outcomes.

2.6 Review of Related Research Works

The literature review provides an overview of the existing research on medication reminder systems. For instance:

Various medication reminder systems have been developed for different mobile platforms. Some systems, like MediHealth, provide user interfaces for setting medication schedules and alerts (Slagle et al., 2011). More advanced systems utilize technologies such as sensors, radio frequency identification (RFID), or motion detection to ensure medication adherence (Becker et al., 2009; Ammouri & Bilodeau, 2008; Prasad, 2013). There are numerous mobile applications for medication management, including those designed to prevent medication administration errors, such as Wedjet (Zao et al., 2010). A recent review identified over 328 mobile applications for medication management (Tabi et al., 2019).

Mobile health systems, a subset of eHealth, use mobile devices for various healthcare services. These applications serve functions such as health data collection, information delivery, patient monitoring, telemedicine, and facilitating healthcare worker collaboration (Qiang et al., 2011). A randomized clinical study was carried out by Morawski et al. (2018) to assess the effect of Medisafe on medication adherence in individuals with uncontrolled hypertension. According to the research, app users' adherence was noticeably better than that of the control group. One of the mHealth strategies for medication adherence that Armitage et al. (2020) considered in their systematic review was Medisafe. According to the research, these apps typically had a favorable impact on adherence; among them is Medisafe, which has undergone more thorough evaluation.

The usage of Medisafe and other medication reminder applications by users was investigated by Santo et al. (2019). They discovered that encouraging long-term app usage mostly involved personalized features and user-friendly interfaces. Johnson et al. (2021) investigated patient opinions about medication reminder applications using qualitative research. Medisafe users expressed gratitude for features including the ability to track various prescriptions and alerts about drug interactions.

In the digital age, the need for effective health management systems has become increasingly paramount, and the development of health reminder systems has emerged as a promising strategy to address this growing concern. The chronic illness burden continues to rise, and researchers have explored innovative technological solutions to

enhance patient adherence to medical treatment regimens. One such approach is the utilization of automated alerts and reminders targeting patients. These IT interventions have been shown to contribute to improved health outcomes when delivered at the point of care to clinical providers. However, there is growing interest in providing alerts and reminders directly to patients to improve healthcare self-management. Leveraging IT to foster patient engagement in their healthcare regimen may be particularly significant, as patients only spend a small amount of time in a clinical environment. Increasing the reach of support and guidance from providers to patients' daily lives can encourage the incorporation of health behaviors into their daily routines (Perri-Moore et al., 2016).

While clinical decision support systems have become a common intervention, studies have highlighted the potential problem of alert fatigue, where the burden to users from these systems can lead to dissatisfaction with electronic health records (EHRs) and even clinician burnout (Kwan et al., 2020). Strategies to minimize this burden are under investigation, underscoring the importance of carefully considering the implementation and design of health reminder systems.

In addition to the challenge of alert fatigue, research has also explored the effectiveness of specific reminder modalities, such as short message service (SMS) reminders. A meta-analysis on the use of SMS reminders has shown that they have value in reducing non-attendance rates in a wide variety of settings, providing a simple and efficient option for enhancing the accessibility and utilization of healthcare services by patients (Guy et al., 2011). However, the effectiveness of SMS reminders is highly dependent on patient engagement, highlighting the need for a deeper understanding of the connection between patient experience, adherence, and health outcomes (Perri-Moore et al., 2016).

Ultimately, the emerging role of technology in enhancing patient adherence to medical treatment regimens holds significant promise, but the complexities and nuances of designing effective health reminder systems require careful consideration and further research. Recent studies have explored the efficacy of various technological solutions, including mHealth, home telemonitoring systems, web-based support, patient portals, and personal health records. These methods may offer solutions to the limitations of traditional means, particularly for conditions requiring complex self-management and lifestyle adjustment.

Moreover, researchers have examined the impact of cognitive impairment and technological literacy on the adoption and continued use of online self-management interventions for chronically ill patients. As the population with chronic diseases continues to grow, it is crucial to develop integrated solutions that address both the technological and human factors to ensure the successful implementation of health reminder systems.

Alahäivälä & Oinas-Kukkonen (2016) claim that gamification in Mango Health System can result in better health outcomes and habits. Thus, sustaining patient engagement and

raising adherence rates may be greatly aided by Mango Health's use of points and prizes for compliance. For any software that deals with health, privacy and data security are crucial factors. Mango Health was one of the health applications whose privacy policies and data management procedures were examined by Huckvale et al. (2019). Although the study could not provide Mango Health-specific results, it did highlight the significance of open data policies and compliance with privacy laws in health reminder systems.

MyTherapy is a popular mobile application that assists individuals in managing their medication schedules and improving medication adherence. Schmidt et al. (2018) compared the usage of MyTherapy to conventional therapy in individuals with type 2 diabetes in a randomized controlled experiment. Throughout 12 weeks, the study discovered that patients utilizing MyTherapy considerably improved their adherence to oral antidiabetic drugs, accompanied by increased glycemic control. According to the researchers, the combination of the app's motivating features and timely reminders is what made it successful. According to Schwebel & Larimer (2018), text message reminders are an efficient and beneficial tool for healthcare services when it comes to medication adherence in MyTherapy.

Johnson & Johnson created the Care4Today medication reminder system in response to the rising issue of prescription non-adherence. The goal of this digital health intervention is to enhance patient outcomes by monitoring drug adherence and sending out timely reminders. According to Dayer et al. (2013), the system consists of an online platform and a mobile app that provide features including refill reminders, adherence data, and personalized prescription regimens. Furthermore, it's important to consider how mHealth apps affect healthcare practitioners. The information gathered by applications such as Care4Today can help medical practitioners by providing better patient monitoring and treatment planning guidance. According to research by Stawarz et al. (2015), medical professionals valued the extra information that mHealth solutions offered since it enabled them to make better clinical judgments. This demonstrates how these systems are advantageous to patients as well as providers.

The effectiveness of tailored, user-friendly mHealth treatments in enhancing health outcomes is supported by current research. The system is a useful tool in contemporary healthcare because of its all-encompassing approach and robust data protection features. An app called RxmindMe was created to assist individuals in better managing their prescription regimens and adherence. For any health reminder system to be successful, user participation is essential. Sarzynski et al. (2017) investigated the opinions of senior citizens on RxmindMe and other medication management applications in qualitative research. Participants voiced worries about privacy and the need for continuing technical assistance, but they also liked the app's straightforward UI and reminder functions. The significance of taking user requirements and preferences into account when designing medication reminder applications was highlighted by this study.

It's crucial to remember, though, that not every study on medication reminder applications has shown consistently favorable outcomes. While not directly examining RxmindMe, a randomized controlled trial by Morawski et al. (2018) discovered that using a smartphone app to track medication adherence did not substantially enhance blood pressure control in hypertensive individuals. This emphasizes the need for more study on how well these applications work over the long run to enhance clinical outcomes.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

This chapter presents the system analysis and design of the proposed Medication Reminder System for the Elderly. It includes an evaluation of existing systems, an overview of the proposed solution, system requirements, architecture, and detailed design components.

3.1 System Analysis

System Analysis is the process of studying a system or its components to identify its objectives, understand its functionality, and determine how it can be improved or optimized. It involves gathering and interpreting data about a system's current state, identifying problems or inefficiencies, and proposing solutions to meet specific goals.

In simpler terms, system analysis is about understanding what a system does, how it works, and how it can be made better. This process is commonly used in fields like software development, business operations, engineering, and IT to design, improve, or replace systems.

3.2 Analysis of Existing Systems

Existing medication reminder systems include traditional methods such as pill organizers, paper calendars, and alarms, as well as modern technological solutions like mobile applications and wearable devices. These systems aim to improve medication adherence by providing reminders and tracking mechanisms.

3 Traditional Methods:

- **Pill Organizers:** These are physical containers with compartments for each day of the week and time of day. They help users organize medications but rely heavily on the user's memory to take the medications.
- **Paper Calendars:** Users manually mark off doses taken, which can be prone to errors and lack real-time monitoring.

- **Alarms and Timers:** Simple devices or phone alarms remind users to take their medications, but they lack personalization and integration with healthcare providers.

4 Technological Solutions:

- **Mobile Applications:** Apps like Medisafe and MyTherapy provide features such as reminders, medication tracking, and educational resources. However, these apps often require internet connectivity and may not cater specifically to the needs of elderly users.
- **Wearable Devices:** Devices like smartwatches offer discreet reminders and health monitoring but may be uncomfortable or unappealing to some elderly users.
- **Smart Home Systems:** These systems automate tasks like medication reminders but are expensive and require technical expertise to set up.

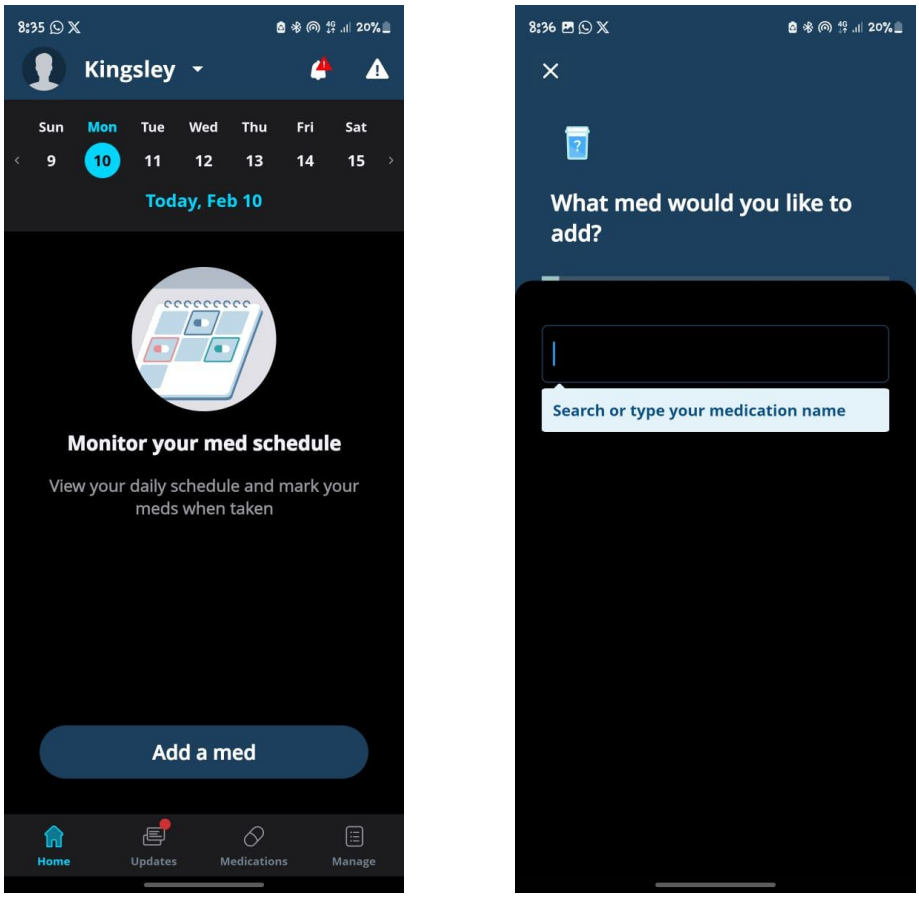

Fig1: Images of Medisafe pills reminder app

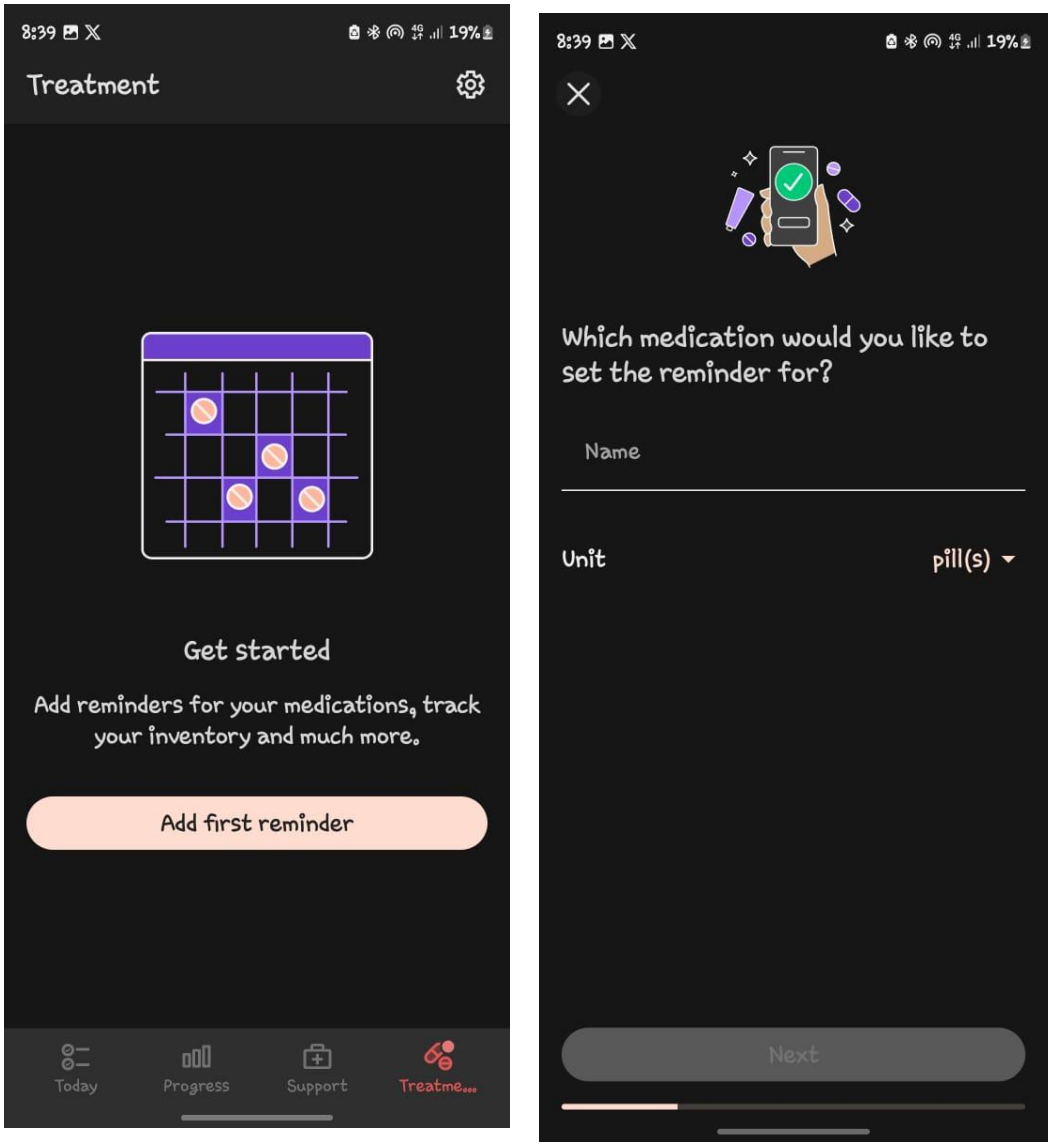

Fig 2: Images of MyTherapy pills reminder app

3.3 Problems with Existing Systems

Despite their benefits, existing systems have several limitations:

1. **Reliance on Memory:** Traditional methods depend on the user's ability to remember to check the organizer, calendar, or alarm.

2. **Lack of Real-Time Monitoring:** Many systems do not provide real-time data on adherence, making it difficult for healthcare providers to intervene promptly.

3. **Limited Personalization:** Existing solutions often fail to cater to the specific needs of elderly users, such as larger fonts, simple interfaces, and auditory or vibratory alerts.

4. **Technical Complexity:** Some technological solutions are too complex for elderly users who may lack digital literacy.

5. **Cost and Accessibility:** Advanced systems like smart home setups are expensive and inaccessible to many elderly individuals, especially in low-income settings.

3.4.1 Overview of the Proposed System

The proposed system is a mobile application designed to address the limitations of existing solutions. Key features include:

- **Personalized Reminders**: Customizable schedules with support for multiple medications and variable doses.
- **Offline Functionality**: Local storage (SQLite) ensures reliability without internet dependency.
- **Adherence Tracking**: Historical logs and visual reports to monitor compliance.
- **Elderly-Friendly Design**: Simplified UI with large fonts, voice commands, and high-contrast visuals.

3.4.2 Advantages Over Existing Systems:

- Real-time notifications via Flutter Local Notifications.
- Secure local data storage using SQLite.
- Cross-platform compatibility (iOS/Android) via Flutter.

3.5.1 System Development Methodology: Waterfall Model

The Waterfall Model is chosen for this project due to its linear, structured approach, which suits the well-defined requirements and scope of the medication reminder

system. The model ensures thorough documentation and sequential progression through phases, critical for academic projects. Below is its application to the system's development:

3.5.2 Phases of the Waterfall Model

1. **Requirements Gathering & Analysis**
    - **Activities**:
        - Conducted interviews with elderly users and caregivers to identify pain points (e.g., forgetfulness, complex schedules).
        - Defined functional requirements (reminders, tracking) and non-functional requirements (usability, reliability).
    - **Output**:
        - Documented **System Requirements**
2. **System Design**
    - **Activities**:
        - Designed the system architecture (three-tier), database schema (SQLite tables), and UI prototypes (Figma).
        - Selected Flutter and Local Notifications for cross-platform compatibility.
    - **Output**:
        - **Architecture Diagrams**, **ER Diagrams**, and **UI Wireframes**
3. **Implementation**
    - **Activities**:
        - Developed the app using Flutter:
            - Integrated flutter_local_notifications for reminders.
            - Implemented SQLite for local data storage (e.g., Medications, IntakeHistory tables).
    - **Output**:
        - Functional app with screens for scheduling and reminders
4. **Testing**
    - **Activities**:
        - Unit Testing: Validated CRUD operations for SQLite.

- Integration Testing: Verified notification triggers and caregiver alerts.
- Usability Testing: Conducted with elderly users to assess UI accessibility.

- **Output**:
  - Test reports documenting bug fixes (e.g., notification delays).

5. **Deployment & Maintenance**
   - **Activities**:
     - Deployed the app to Google Play Store/Apple App Store.
     - Provided user manuals for elderly users and caregivers.
     - Planned future updates (e.g., voice assistant integration).
   - **Output**:
     - Live application and maintenance plan.

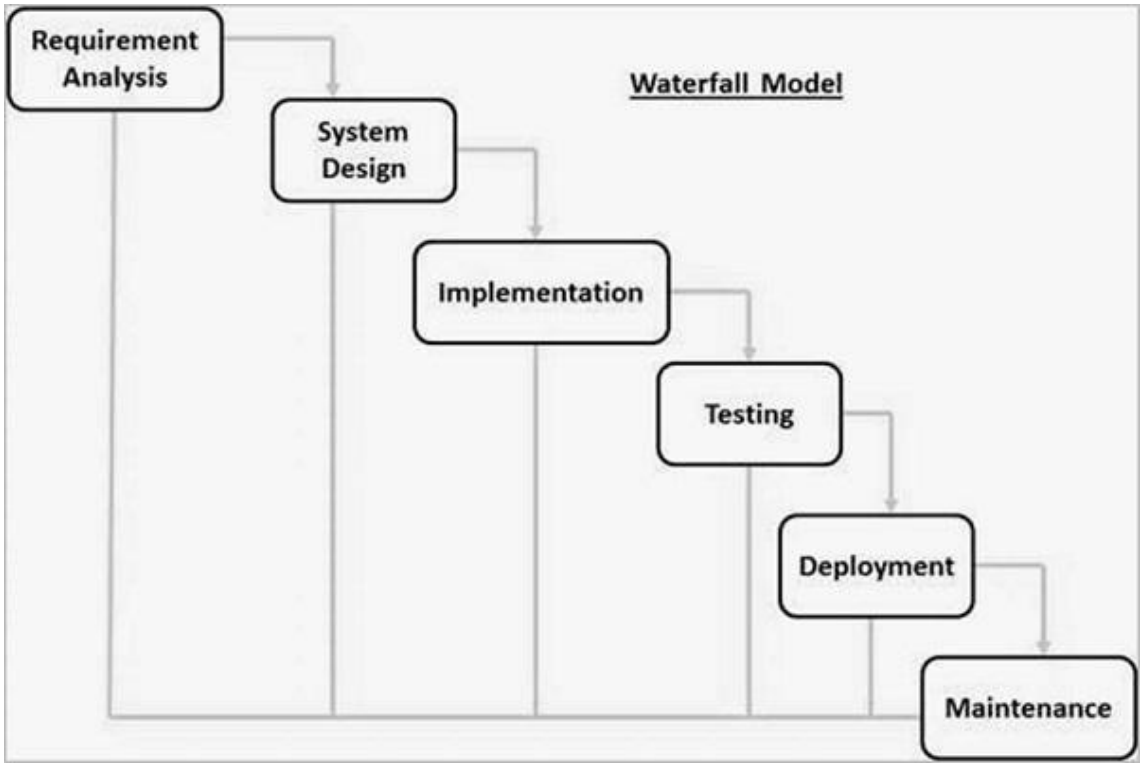

Fig3: Waterfall model

3.5.3 Rationale for Choosing Waterfall Model

1. **Clear Requirements**: The project scope (elderly-focused app with reminders) is well-defined, minimizing the need for iterative changes.

2. **Documentation Emphasis**: Required for academic rigor; each phase produces deliverables (e.g., SRS, design docs).
3. **Predictability**: Sequential phases ensure structured progress, aligning with university project timelines

3.6 System Requirements

3.6.1 Functional Requirements

1. **User Registration/Profile Management**:
    - Create user profiles (name, age, medical conditions).
2. **Medication Scheduling**:
    - Define medication names, dosages, frequencies, and durations.
3. **Reminder System**:
    - Push notifications with snooze/dismiss options.
4. **Data Export**:
    - Share adherence reports with healthcare providers.

3.6.2 Non-Functional Requirements

1. **Usability**: Intuitive interface with minimal navigation steps.
2. **Reliability**: Notifications must trigger consistently, even when the app is closed.
3. **Performance**: Fast response time (<1 second) for CRUD operations.
4. **Security**: Local data encryption for SQLite databases.
5. **Scalability**: Modular architecture to support future enhancements.

3.7 Proposed System Architecture

The system follows a **three-tier architecture**:

1. **Presentation Layer (UI)**:
    - Built with Flutter widgets for cross-platform compatibility.

- Screens: Login, Medication List, Reminder Setup, Adherence Dashboard.

2. **Application Layer (Logic)**:

    - Manages reminders using the **Flutter Local Notifications** package.
    - Handles business logic for scheduling, tracking, and alerts.

3. **Data Layer (Storage)**:

    - SQLite database for storing user profiles, medications, and logs.
    - Local data persistence for offline access.

3.8 System Design

3.8.1 System Interface and Architecture

The system adopts a three-tier architecture to separate concerns and ensure modularity:

- **Presentation Layer (Frontend)**
    - **Technology:** Flutter framework (Dart language).
    - **Key Screens:**
        - **Login/Registration:** Simplified sign-up with minimal fields (name, age, emergency contact).
        - **Add Medication:** Form to input medication details (name, dosage, frequency, start/end dates).
        - **Reminder Settings:** Customize notification tones, snooze duration (5/10/15 mins), and voice alerts.
    - **Accessibility Features:**
        - **Large Fonts & High Contrast:** Predefined themes for visually impaired users.
- **Application Layer (Backend Logic)**
    - **Core Modules:**
        - **Reminder Scheduler:**

- Uses flutter_local_notifications to schedule alerts based on user-defined frequencies (e.g., daily, weekly, custom intervals).
- Handles recurring reminders (e.g., "Take every 8 hours") via AndroidAlarmManager (Android) and Timer (iOS).

- **Data Validator:**
  - Ensures no overlapping medication schedules (e.g., two reminders at the same time).
- **State Management:**
  - Uses Provider package to manage global state (e.g., user profile, active medications).
- **Data Layer (Persistence)**
- **Database:** SQLite for offline storage.
- **Tables:**
  - Users (userID, name, age).
  - Medications (medID, userID, medName, frequency, startDate, en dDate).
  - Reminders (reminderID, medID, time, isRecurring).
  - IntakeHistory (historyID, medID, date, status).

3.9 Database Design

Entity-Relationship Diagram (ERD):

Entity Relationship Diagram, also known as **ERD**, ER Diagram or ER model, is a type of structural diagram for use in database design. An ERD contains different symbols and connectors that visualize two important information: The major entities within the system scope, and the inter-relationships among these entities.

And that's why it's called "**Entity**" "**Relationship**" diagram (ERD)!

When we talk about entities in ERD, very often we are referring to business objects such as people/roles (e.g. Student), tangible business objects (e.g. Product), intangible business objects (e.g. Log), etc. "Relationship" is about how these entities relate to each other within the system.

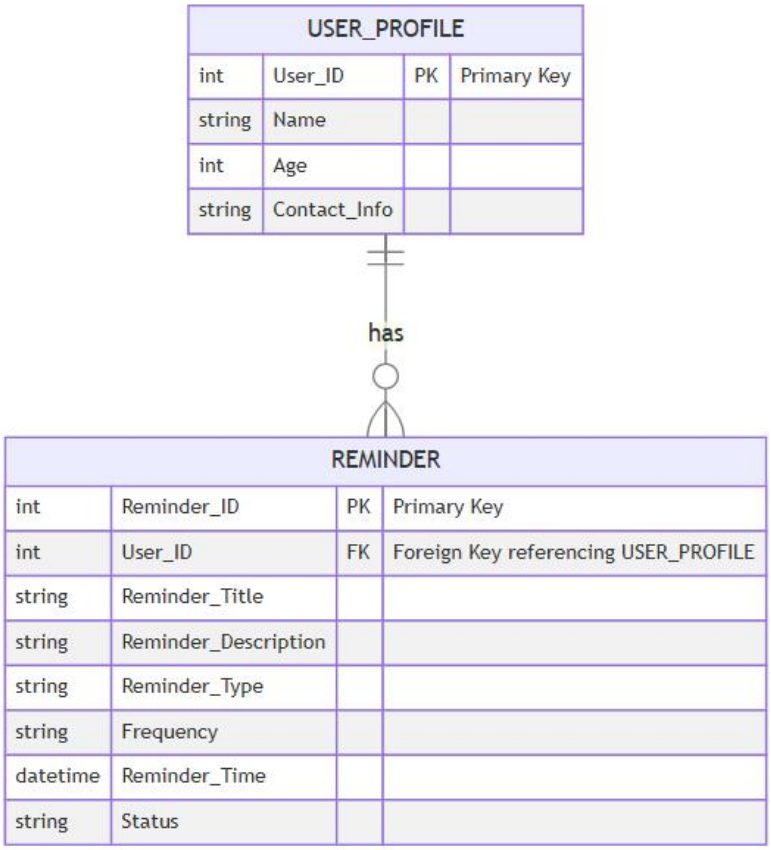

Fig4: Entity relationship diagram of Medical Reminder database

Table Specifications:

1. **Users Table:**

Column	Type	Description
userID	**INTEGER**	**Primary key, auto-increment.**
name	**TEXT**	**User's full name.**

Column	Type	Description
age	INTEGER	User's age.

2. **Reminders Table:**

Column	Type	Description
reminderID	INTEGER	Primary key, auto-increment.
Reminder category	TEXT	Foreign key linking to Medications.
Time	DATETIME	Scheduled reminder time.
Frequency	DATETIME	time for recurring reminders.

3.9.1 User Interface (UI) Design

The UI design focuses on ensuring ease of use for elderly users:

- **Home/Dashboard Screen:**
  - A clean layout that presents a daily overview of active reminders.
  - Large buttons and icons for ease of navigation.
- **Reminder Entry Screen:**
  - Simple forms with guided steps for entering reminder details.
  - Use of dropdown menus, date and time pickers, and text fields with ample spacing.
- **Reminder Details Screen:**
  - Displays all details of a specific reminder, along with options to edit or delete.
  - Provides clear indicators (e.g., check marks or color changes) when a reminder is completed.

- **Settings Screen:**

- Allows users to personalize the look and feel of the app, adjust notification preferences, and access help sections.

3.9.2 Process Flow

The process flow outlines the interactions and sequences of operations within the system:

1. **User Registration/Authentication:**
   - On first launch, users create a profile or log in.
   - The system authenticates the user and loads their personalized settings.
2. **Creating a Reminder:**
   - The user navigates to the reminder entry screen.
   - The user inputs details (title, type, time, frequency, and notes).
   - The app validates and stores the reminder in the SQLite database.
3. **Scheduling Notifications:**
   - Based on the user's inputs, the system calculates the schedule for reminders.
   - The Flutter local notification package is used to schedule these notifications.
4. **Receiving Notifications:**
   - At the scheduled times, the notification service triggers an alert.
   - The user can view the details, mark the reminder as completed, or opt to reschedule.
5. **Managing Reminders:**
   - The dashboard and reminder list allow users to view active and past reminders.
   - Users can update or delete reminders, with changes reflected in real time.

6. **Error Handling:**

   - The system provides error messages or prompts for re-entry if any data input errors occur.
   - It also supports recovery options (such as rescheduling) in case of missed notifications.

3.10 System Development Tools

Tool	Purpose
Flutter 3.7	**Cross-platform UI development.**
SQLite 3.40	**Local database management.**
Figma	**UI prototyping and wireframing.**
Android Studio	**Emulator testing (Android).**
Xcode	**Emulator testing (iOS).**

3.11 UML Diagrams

Unified Modeling Language (UML) is a standardized visual modeling language that is a versatile, flexible, and user-friendly method for visualizing a system's design. Software system artifacts can be specified, visualized, built, and documented with the use of UML.

- We use UML diagrams to show the behavior and structure of a system.
- UML helps software engineers, businessmen, and system architects with modeling, design, and analysis.

3.12 UML – Use Case Diagram

Use case diagrams belong to the behavioral category of UML diagrams. They capture the functional requirements of a system by showing which users or external systems

(called “actors”) interact with the system and what functions (called “use cases”) the system performs for them.

Key Elements:

- **Actors:** Represent external users, systems, or roles that interact with your system. They are typically depicted as stick figures.
- **Use Cases:** Represent the functions or services the system provides. They’re usually drawn as ovals or ellipses.
- **Associations:** Lines connecting actors to use cases, which indicate that an actor participates in that functionality.
- **Relationships Among Use Cases:** Sometimes use cases include relationships like «include» (a use case always calls another use case) or «extend» (a use case may add additional behavior under certain conditions).

  For the Medical Reminder System:

  - **Actor:** Elderly User
  - **Use Cases:** Login, enter reminder details, schedule reminder, receive notifications.

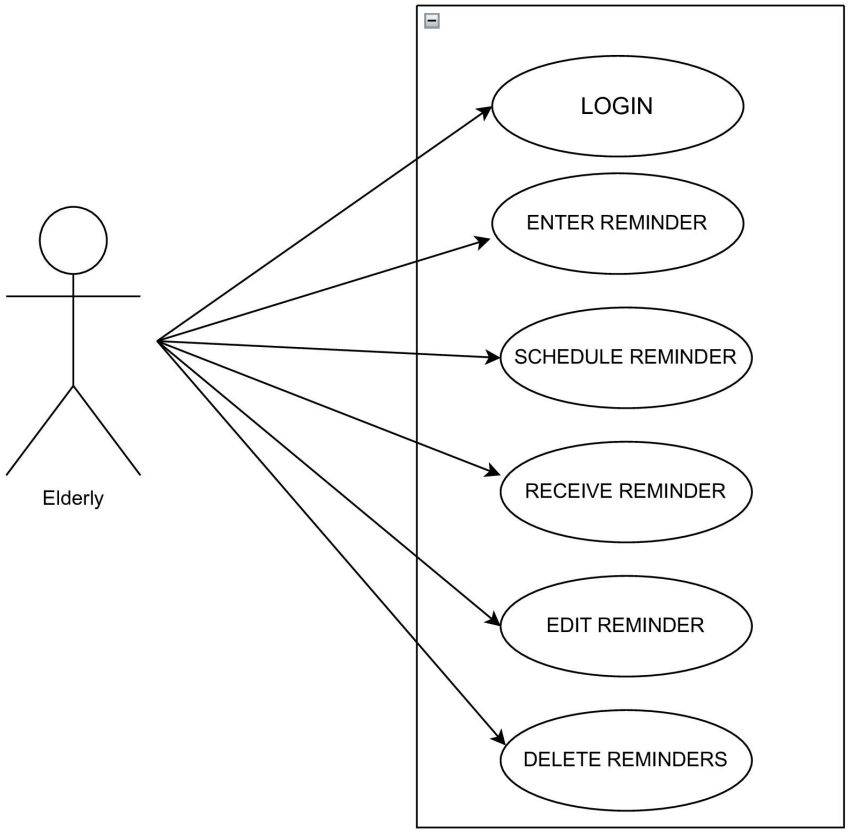

Fig 5: Medical Reminder system use case diagram

3.13 UML – Class Diagram

A class diagram is a type of static structure diagram in the Unified Modeling Language (UML) that serves as a blueprint for the system's structure. It shows the system's classes, the attributes and operations (or methods) those classes contain, and the relationships among the classes. In essence, class diagrams allow developers, designers, and stakeholders to visually model the static aspects of an object-oriented system.

Key Elements of a Class Diagram

- **Classes:**

  Represented by boxes (often divided into compartments), a class encapsulates data and behavior. The typical layout is three sections:

  - **Top section:** Displays the class name.

- **Middle section:** Lists the attributes (or properties) of the class.
- **Bottom section:** Contains the methods (or operations) the class can perform.

  This structure helps in identifying what each object "knows" (its attributes) and what it "can do" (its methods).

- **Relationships:**

  The diagram uses different types of lines and arrows to illustrate various relationships between classes:

  - **Association:** A general connection between classes.
  - **Inheritance (Generalization):** Depicts an "is-a" relationship, where one class (the subclass) inherits from another (the superclass).
  - **Aggregation and Composition:** Special types of association that denote "has-a" relationships. In composition, for example, the lifetime of the part is strictly dependent on the whole.
  - **Dependency:** Indicates that a change in one class might affect another even if they are not directly connected as attributes.

- **Visibility Modifiers:**

  Symbols such as + (public), - (private), and # (protected) precede attributes or methods to denote their access levels, helping specify which parts of a class are accessible to other parts of the system.

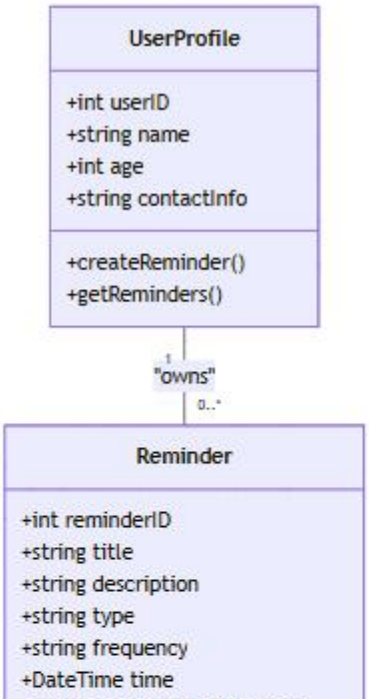

Fig6: Medical Reminder system class diagram

CHAPTER FOUR

SYSTEM IMPLEMENTATION

This chapter details the practical realization of the personalized health reminder system. It explains the software tools and technologies used, describes the testing and documentation efforts, evaluates the system's usability, and discusses future maintenance and enhancements.

4.1 Software Implementation Tools

The software implementation of the system is underpinned by a suite of modern tools and technologies. This section explains the implementation languages, frameworks, platforms, deployment strategies, and additional tools that have contributed to the development of the application.

4.1.1 Implementation Languages

- **Dart:**

  The primary programming language used is Dart, which is the foundation of Flutter. Dart provides both ahead-of-time (AOT) and just-in-time (JIT) compilation, enabling rapid development cycles, expressive syntax, and robust performance. Dart's strong type system and asynchronous programming model are particularly well-suited for building reactive mobile applications.

- **SQL:**

  Structured Query Language (SQL) is used for interacting with the SQLite database, which manages local data storage. SQL queries are designed to efficiently execute CRUD (Create, Read, Update, Delete) operations on the reminder records.

4.1.2 Implementation Framework and Packages

- **Flutter Framework:**

  Flutter is chosen for its cross-platform capabilities, which allow the system to run on both Android and iOS with a single codebase. Flutter's widget-centric architecture facilitates the creation of a responsive, intuitive, and user-friendly interface, which is critical for the elderly user demographic. The reactive framework supports real-time updates and dynamic content adjustments based on user interactions.

- **SQLite:**

  A lightweight and efficient database engine is integrated for local storage, ensuring data is available offline. The choice of SQLite also minimizes dependencies on network connectivity, a critical factor for the target user group.

- **Flutter Local Notification Package:**

  This package provides a robust mechanism for scheduling and delivering notifications, ensuring that personalized health reminders are delivered reliably and on time.

4.1.3 Integrated Development Environments (IDEs):

Visual Studio Code and **Android Studio** are employed for coding, debugging, and performance profiling. Their robust plugin ecosystems (including Flutter and Dart plugins) streamline the development process.

4.1.4 Implementation Platforms

- **Mobile Platforms:**

  The application is primarily developed for mobile platforms, including:

  - **Android:** Utilizing Android SDK integration with Flutter.
  - **iOS:** Leveraging iOS-specific libraries and Flutter's cross-compilation features.

This dual-platform support ensures that the application is accessible to a wide range of elderly users, regardless of their preferred mobile device.

\

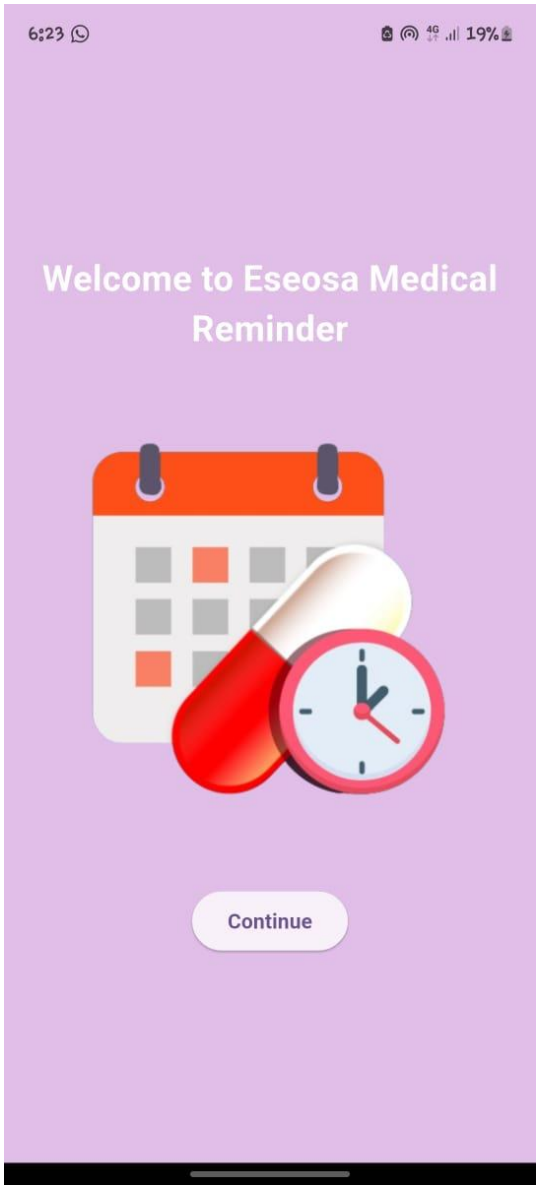

Fig 7: Medical Reminder Welcome screen

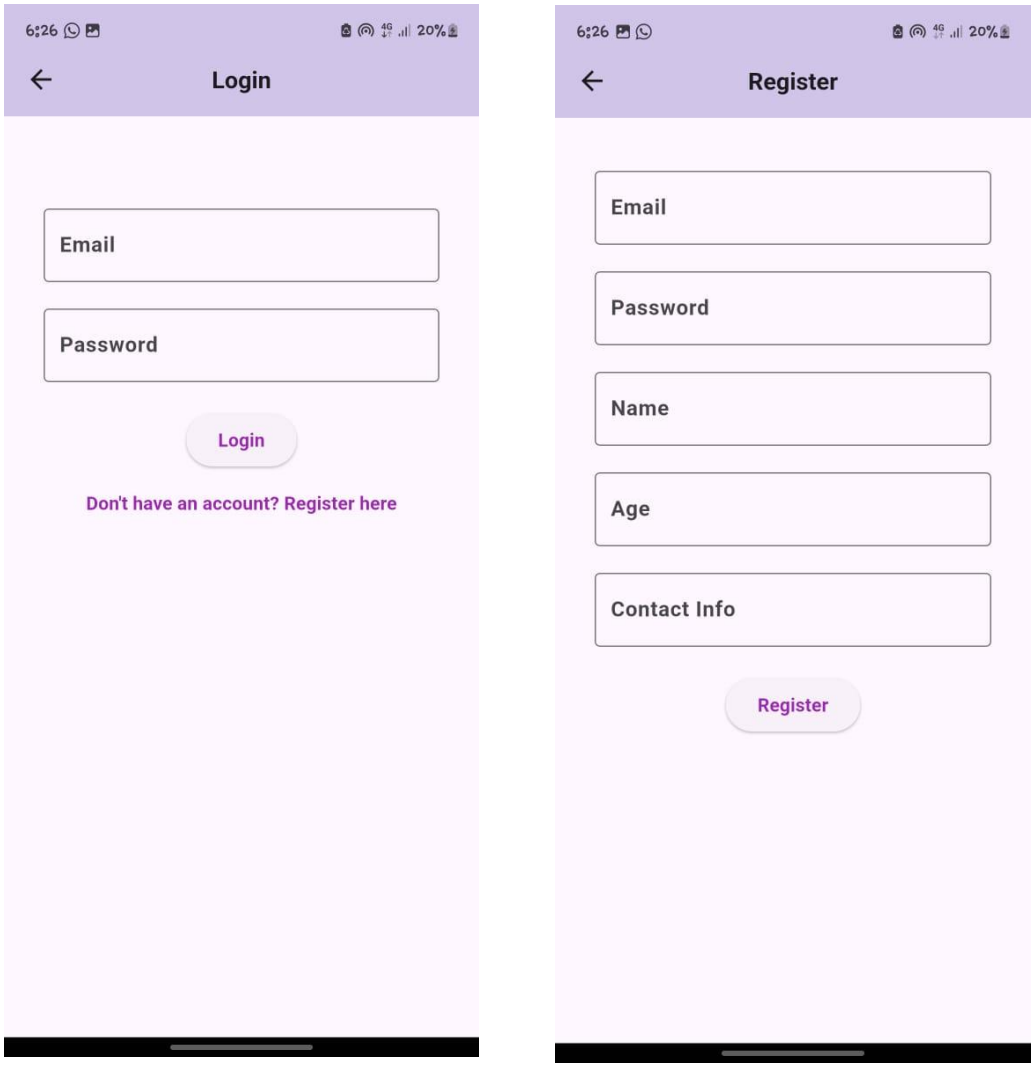

Fig 8: User Authentication Screens

This screen provides a user-friendly login interface where elderly users can enter their registered email and password to access the medication reminder system and also enables new users to register by entering their personal details, including name, age, contact information, email, and password. Once the necessary information is provided,

they can tap the "Register" button to create an account, allowing them to receive personalized medication reminders.

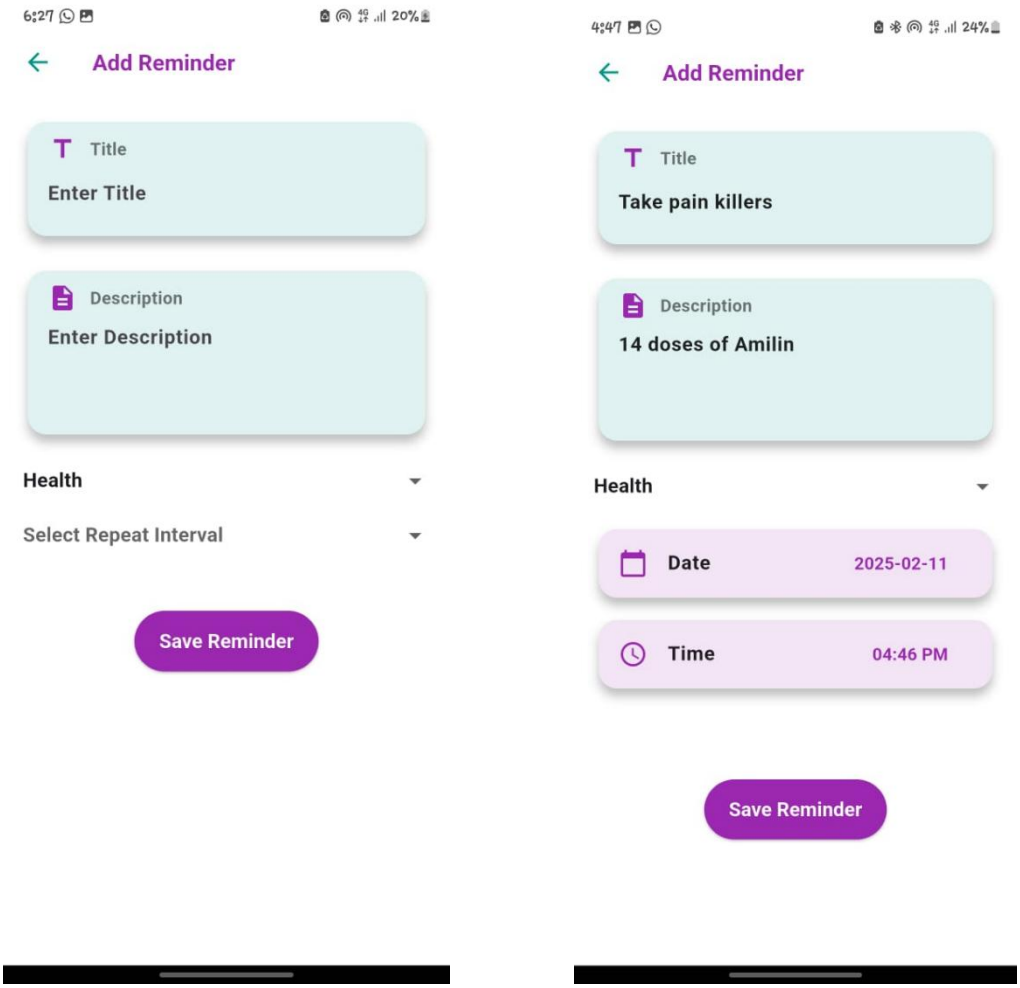

Fig 9: Add reminder screens

The "Add Reminder" screen enables users to set reminders with a title, description, and category (e.g., Health). Users can also specify a repeat interval and schedule reminders with a date and time selection feature.

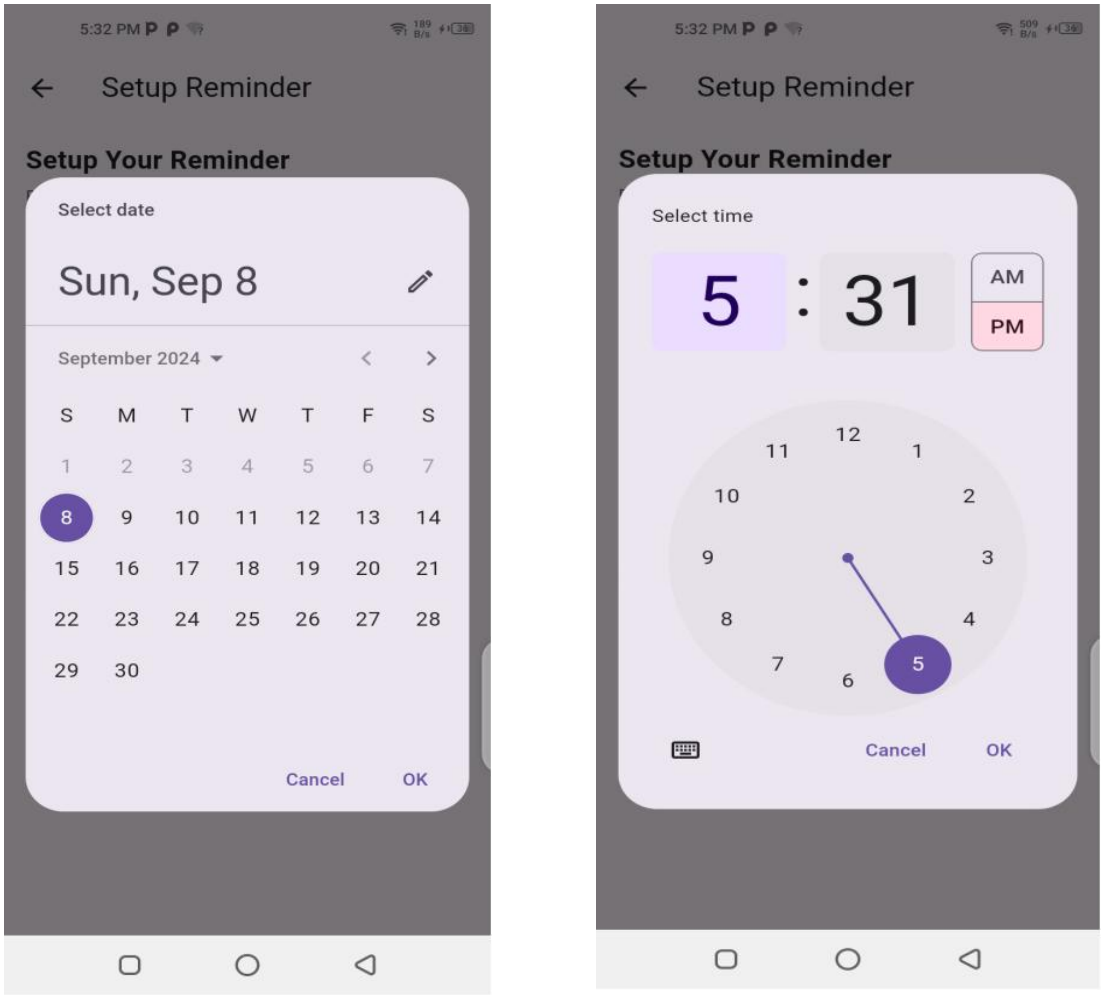

Fig 10: Select time and date for reminder setup

The **Setup Reminder** screen allows users to define when their reminders should trigger.

- **Time Picker:** Users can select the exact time for their reminder using an intuitive clock interface. They can toggle between **AM and PM** to ensure accuracy.
- **Date Picker:** Users can choose a specific date using a calendar interface, making it easy to schedule one-time or future reminders.

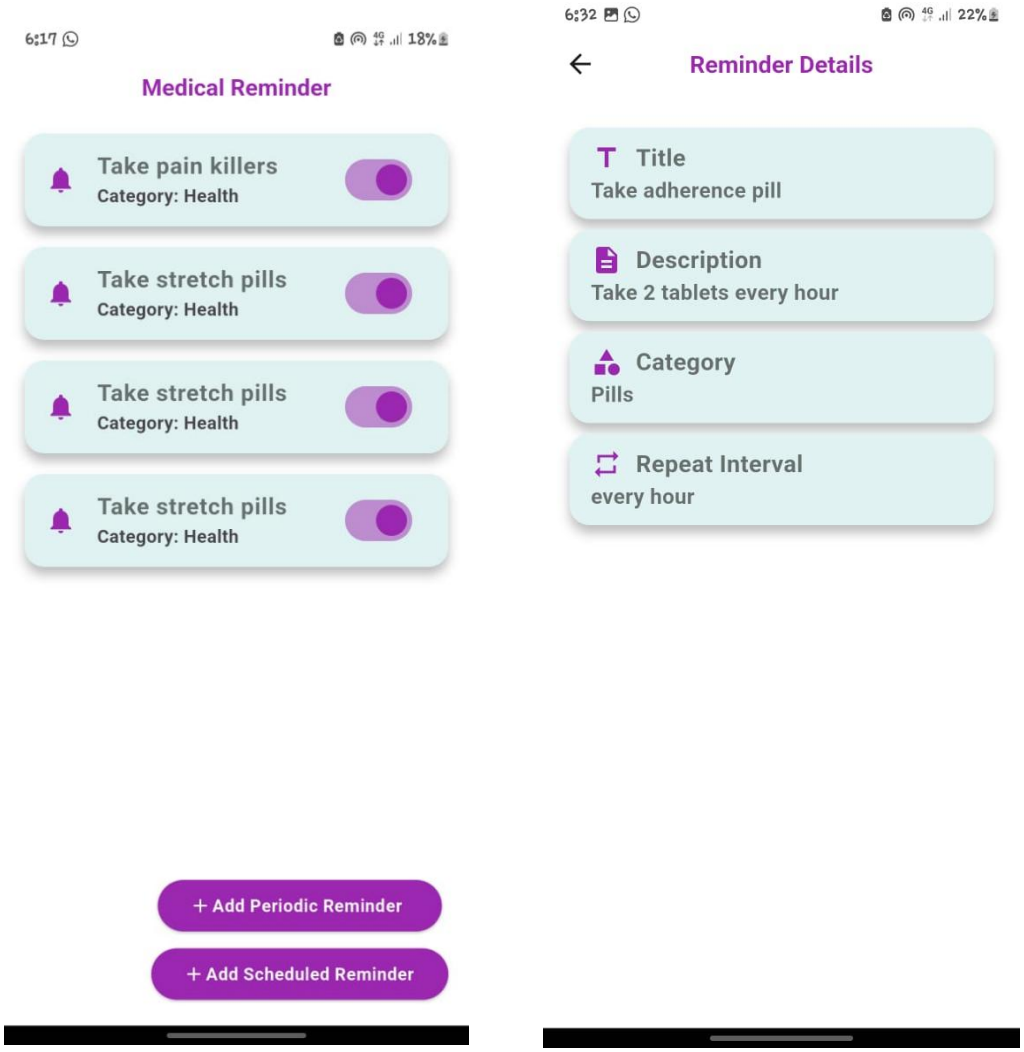

Fig 11: Home screen and reminder details screen

A home screen displays all active reminders, allowing users to enable or disable them using a toggle switch.Users can view specific reminder details, including medication dosage, repeat intervals, and categories.

4.1.5 Deployment Platforms

- **Application Stores:**

- **Google Play Store:** For distribution to Android users.
- **Apple App Store:** For distribution to iOS users.

Both stores require adherence to stringent security and performance guidelines, ensuring the final product is robust and reliable.

- **Version Control and CI/CD:**
  The system uses Git for version control, with repositories hosted on platforms like GitHub. Continuous Integration/Continuous Deployment (CI/CD) pipelines are set up using tools such as GitHub Actions or CircleCI to automate testing, build processes, and deployments.

4.2 User Documentation and System Testing

An effective system implementation not only involves building the software but also ensuring that it is thoroughly tested and well-documented. This section covers the approaches to user documentation and the testing methodologies employed.

4.2.1 User Documentation

- **User Manuals and Quick Start Guides:**
  Comprehensive user manuals will be developed, including step-by-step instructions on how to install, navigate, and use the application. Quick start guides and FAQs are included to help users troubleshoot common issues.
- **Video Tutorials:**
  A series of short, engaging video tutorials will be produced to guide users through key functionalities, such as setting up reminders, customizing notifications, and managing user profiles.
- **In-App Help and Tooltips:**
  Contextual help is provided throughout the application using tooltips and interactive prompts. This feature is particularly useful for elderly users who might need additional guidance during their first interactions with the system.

4.2.2 System Testing

- **Unit Testing:**

  Individual components and functions are rigorously tested using Dart's built-in testing frameworks. Unit tests ensure that each method and class performs as expected.

- **Integration Testing:**

  Integration tests are conducted to verify that different modules (e.g., the user interface, database management, and notification system) work together seamlessly. This ensures that data flows correctly from the UI to the database and that notifications are triggered appropriately.

- **System Testing:**

  Comprehensive system testing is performed in simulated environments that mimic real-world usage scenarios. These tests cover functionality, performance, and reliability under various conditions, including intermittent connectivity and high load.

- **User Acceptance Testing (UAT):**

  A group of target users (elderly individuals and their caregivers) participate in UAT sessions. Their feedback is used to refine the interface, improve usability, and ensure that the system meets the practical needs of its audience.

- **Automated Testing Tools:**

  Tools like Flutter's integration test suite and third-party frameworks such as Mockito for Dart are employed to automate test execution, regression testing, and performance monitoring.

4.3 System Usability Evaluation

Evaluating the usability of the system is critical, especially given the target user base. This section outlines the methodologies and metrics used to assess system usability.

4.3.1 Evaluation Methodologies

- **Heuristic Evaluation**

  Expert evaluators use established usability heuristics (e.g., Nielsen's heuristics)

to identify potential usability issues in the system's design. This method helps in early detection of design flaws.

- **User Surveys and Interviews:**
  Structured surveys and semi-structured interviews will be conducted with elderly users and caregivers. The surveys include standardized usability scales such as the System Usability Scale (SUS) to quantify user satisfaction.

- **Observation and Field Studies:**
  Observational studies are carried out in real-world settings, allowing researchers to see firsthand how users interact with the application. Field studies provide insights into common challenges and user behavior in natural environments.

4.3.2 Usability Metrics

- **Task Completion Rate:**
  The percentage of users who successfully complete key tasks (e.g., setting a reminder) without assistance.

- **Error Rate:**
  The frequency and type of errors encountered by users during interaction, which helps identify areas for improvement.

- **Time on Task:**
  The amount of time users spend completing specific actions, providing an indication of the system's efficiency.

- **User Satisfaction:**
  Quantitative metrics (such as SUS scores) and qualitative feedback are used to assess overall user satisfaction with the system.

4.3.3 Results and Iterations

- **Analysis of Feedback:**
  User feedback is meticulously analyzed, with common issues and suggestions for improvement documented.

- **Iterative Improvements:**
  Based on the evaluation results, iterative design modifications are implemented

to enhance usability. This iterative approach ensures the system evolves to better meet user needs and preferences.

4.4 System Maintenance and Future Enhancements

- **Maintenance Plan:**

  A detailed maintenance plan outlines regular updates, bug fixes, and performance optimizations. Maintenance procedures are defined to ensure long-term system stability and security.

- **Future Enhancements:**

  Potential future enhancements include:

  - Integration with wearable devices for real-time health monitoring.
  - Cloud synchronization to enable cross-device data access.
  - Advanced analytics to provide personalized health insights.
  - Additional language support and accessibility improvements for broader user adoption.

4.5 Performance Evaluation

- **Benchmarking:**

  The system is benchmarked against key performance indicators such as response time, database query efficiency, and notification delivery latency. Performance tests are run under various load conditions to ensure scalability.

- **Stress Testing:**

  Stress tests simulate extreme usage scenarios to evaluate system robustness. The outcomes guide performance tuning and help ensure that the system remains responsive even under high demand.

4.6 Challenges and Lessons Learned

- **Technical Challenges:**

  The implementation encountered challenges such as ensuring seamless offline

functionality, managing asynchronous tasks in Flutter, and optimizing database operations with SQLite.

- **User-Centric Design Considerations:**
  Iterative testing with the target user group revealed critical insights regarding accessibility and ease of use, underscoring the importance of continuous user feedback.

- **Lessons for Future Projects:**
  Future projects can benefit from early user involvement in the design process, robust automated testing pipelines, and flexible architectures that accommodate rapid iteration.

CHAPTER FIVE

SUMMARY AND CONCLUSION

This chapter provides an overview of the entire research study, synthesizing the key aspects from system analysis, design, and implementation. It reflects on the major findings, the contributions to the field of elderly care through technology, and the limitations encountered. The chapter concludes with recommendations for future work that can further enhance the system and broaden its impact.

5.1 Summary of the Research

The primary objective of this research was to design and implement a personalized health reminder system aimed at improving the medication adherence and overall health management of elderly users. The study commenced with a thorough analysis of existing systems, identifying critical gaps such as non-intuitive user interfaces, generic notification methods, and over-reliance on continuous connectivity. Based on these insights, a novel solution was proposed and developed using modern tools and methodologies.

Key phases of the research included:

- **System Analysis and Design:**
  An in-depth examination of the shortcomings of existing medication reminder solutions led to the development of a system architecture that prioritizes simplicity, accessibility, and reliability. The system was conceptualized to include a user-friendly interface, robust local notification mechanisms using Flutter's local notification package, and efficient data management with SQLite for offline capabilities.

- **System Implementation:**
  The application was built using Dart and Flutter, leveraging cross-platform compatibility to support both Android and iOS devices. The integration of SQLite ensured local data storage while the use of Visual Studio Code and Android Studio streamlined development through advanced debugging, performance profiling, and version control.

- **Testing and Usability Evaluation:**
  A rigorous testing regime was applied, including unit testing, integration testing, system testing, and user acceptance testing (UAT). Usability evaluations, comprising heuristic assessments, user surveys, and field studies, confirmed that the system met the accessibility and functionality requirements of elderly users and their caregivers.

5.2 Key Findings

The research yielded several important findings that contribute to the field of mobile health applications for the elderly:

- **Enhanced User Experience:**
  The system's design, emphasizing large fonts, clear icons, and simple navigation—proved effective in addressing the usability challenges faced by elderly users. Feedback from user testing indicated a significant improvement in ease of use and overall satisfaction.

- **Robust Notification System:**
  The integration of the Flutter local notification package ensured that reminders were delivered reliably and on time. This reliability is crucial in preventing missed doses and promoting better medication adherence.

- **Offline Functionality:**
  By employing SQLite for local data storage, the system maintained functionality in offline environments, thus overcoming limitations related to inconsistent internet connectivity—a common issue for the target demographic.

- **Efficient Development Process:**
  The combined use of Visual Studio Code and Android Studio not only expedited development cycles through features like hot reload and real-time debugging but also ensured a robust quality assurance process via integrated testing and profiling tools.

5.3 Contributions and Impact

research makes several noteworthy contributions:

- **User-Centric Design Approach:**
  The system exemplifies a design approach that is sensitive to the needs of elderly users. By focusing on accessibility and simplicity, it sets a benchmark for future mobile health applications aimed at this demographic.

- **Technical Innovation:**
  The effective use of Flutter for cross-platform development, combined with local data storage and dynamic notification scheduling, demonstrates a scalable and efficient solution that minimizes dependency on external networks.

- **Health Management Improvement:**
  The personalized health reminder system directly addresses issues related to medication non-adherence—a significant concern in elderly care—thereby potentially reducing health risks and improving quality of life for its users.

5.4 Limitations and Future Work

- While the research successfully achieved its objectives, several limitations were identified:

- **Scope of User Testing:**
  The user testing phase, although thorough, was conducted with a limited sample size. Expanding the demographic diversity in future studies would provide more comprehensive insights into usability and performance.

- **Feature Set:**
  The current iteration focuses primarily on medication and health reminders. Future versions could incorporate additional functionalities such as real-time biometric monitoring, integration with wearable devices, and advanced analytics for personalized health insights.

- **Connectivity and Data Synchronization:**
  The emphasis on offline functionality via SQLite is advantageous for immediate

accessibility; however, integrating a cloud-based synchronization feature could enhance data backup, multi-device accessibility, and overall user experience.

- **Future Work Recommendations:**

- **Advanced Personalization:**
  Explore the use of machine learning algorithms to analyze user behavior and health data, enabling predictive and adaptive reminder systems tailored to individual needs.

- **Enhanced Integration:**
  Incorporate cloud services to enable real-time data synchronization, comprehensive backup, and cross-device compatibility, thereby improving the robustness of the health management system.

- **Broader Testing:**
  Engage in extensive user testing with a larger and more diverse user base, including caregivers, to refine the system's usability and functionality further.

5.5 Conclusion

In conclusion, the personalized health reminder system developed in this research represents a significant advancement in addressing the unique challenges associated with elderly health management. Through a user-centric design and robust technical implementation, the system not only improves medication adherence but also lays the groundwork for future innovations in mobile health technology.

The outcomes of this research underscore the potential for digital solutions to contribute meaningfully to elderly care, enhancing both independence and quality of life. As the system evolves through further research and development, it is expected to serve as a comprehensive tool that supports healthier living and more proactive health management for the elderly population.

The journey from conceptualization to implementation has provided valuable insights and established a solid foundation for subsequent work in this area. Continued advancements in technology and user-centered design are essential for further

improving the accessibility and effectiveness of health management systems for vulnerable populations.

REFERENCES

Adler, A. J., Martin, N., Mariani, J., Tajer, C. D., Owolabi, O. O., Free, C., Serrano, N. C., Casas, J. P., & Perel, P. (2017). Mobile phone text messaging to improve medication adherence in secondary prevention of cardiovascular disease. *The Cochrane database of systematic reviews*, *4*(4), CD011851. https://doi.org/10.1002/14651858.CD011851.pub2

Alahäivälä, T., & Oinas-Kukkonen, H. (2016). Understanding persuasion contexts in health gamification: A systematic analysis of gamified health behavior change support systems literature. *International Journal of Medical Informatics*, *96*, 62-70.https://doi.org/10.1016/j.ijmedinf.2016.02.006

Ammouri, S., & Bilodeau, G. A. (2008). Face and hands detection and tracking applied to the monitoring of medication intake.*Proceedings of Canadian Conference on Computer and Robot Vision*, 147-154.

Appari, A., & Johnson, M. E. (2010). Information security and privacy in healthcare: Current state of research. *International Journal of Internet and Enterprise Management*, *6*(4), 279-314.

Armitage, L. C., Kassavou, A., & Sutton, S. (2020). Do mobile device apps designed to support medication adherence demonstrate efficacy? A systematic review of randomized controlled trials, with meta-analysis. *BMJ Open*, *10*(1), e032045.

Azeez Adeyinka Idowu, Rafiu Adesina Ganiyu, Iyiola Adebayo Timothy, & Bamidele Paul Ayomide. (2024). Design and implementation of medicine time reminder system using sound and vibrator Indicators. *International Journal of Advanced Research in Computer and Communication Engineering*. DOI: 10.17148/IJARCCE.2024.13819

Bhati, S., Soni, H., Zala, V., Vyas, P., & Sharma, Y. (2017). Smart Medicine Reminder Box. *IJSTE - International Journal of Science Technology & Engineering*, *3*(10), April 2017.

Choudhry, N. K., Krumme, A. A., Ercole, P. M., Girdish, C., Tong, A. Y., Khan, N. F., Brennan, T. A., Matlin, O. S., Shrank, W. H., & Franklin, J. M. (2017). Effect of Reminder Devices on Medication Adherence: The REMIND Randomized Clinical Trial. *JAMA internal medicine*, *177*(5), 624–631. https://doi.org/10.1001/jamainternmed.2016.9627

Dayer, L., Heldenbrand, S., Anderson, P., Gubbins, P. O., & Martin, B. C. (2013). Smartphone medication adherence apps: Potential benefits to patients and providers. *Journal of the American Pharmacists Association*, *53*(2), 172-181.

Huckvale, K., Torous, J., & Larsen, M. E. (2019). Assessment of the data sharing and privacy practices of smartphone apps for depression and smoking cessation. *JAMA Network Open*, *2*(4), e192542.

Kamal, A. K., Shaikh, Q., Pasha, O., Azam, I., Islam, M., Memon, A. A., Rehman, H., Akram, M. A., Affan, M., Nazir, S., Aziz, S., Jan, M., Andani, A., Muqeet, A., Ahmed, B., & Khoja, S. (2015). A randomized controlled behavioral intervention trial to improve medication adherence in adult stroke patients with

prescription tailored Short Messaging Service (SMS)-SMS4Stroke study.*BMC neurology*, *15*, 212. https://doi.org/10.1186/s12883-015-0471-5s

Kwan, J. L., Lo, L., Ferguson, J., Goldberg, H., Diaz-Martinez, J. P., Tomlinson, G., ... & Rapoport, M. J. (2020). Computerised clinical decision support systems and absolute improvements in care: meta-analysis of controlled clinical trials. *BMJ*, *370*, m3216.

Mohammed, S., Glennerster, R., & Khan, A. J. (2016). Impact of a Daily SMS Medication Reminder System on Tuberculosis Treatment Outcomes: A Randomized Controlled Trial. *PloS one*, *11*(11), e0162944. https://doi.org/10.1371/journal.pone.0162944

Morawski, K., Ghazinouri, R., Krumme, A., Lauffenburger, J. C., Lu, Z., Durfee, E., ... & Choudhry, N. K. (2018). Association of a smartphone application with medication adherence and blood pressure control: The MedISAFE-BP randomized clinical trial.*JAMA Internal Medicine*, *178*(6), 802-809.

Qiang, C. Z., Yamamichi, M., Hausman, V., & Altman, D. (2011). Mobile Applications for the Health Sector. *ICT Sector Unit World Bank*.

Sarzynski, E., Decker, B., Thul, A., Weismantel, D., Melaragni, R., Cholakis, E., ... & Given, C. (2017). Beta testing a novel smartphone application to improve medication adherence. *Telemedicine and e-Health*, *23*(4), 339-348.

Schmidt, S., Schuchert, A., Krieg, T., & Oeff, M. (2018). Home telemonitoring in patients with chronic heart failure: A chance to improve patient care? *DeutschesÄrzteblatt International*, *115*(40), 659.

Schwebel, F. J., & Larimer, M. E. (2018, September 1). Using text message reminders in health care services: A narrative literature review. *Elsevier BV*, *13*, 82-104. https://doi.org/10.1016/j.invent.2018.06.002

Slagle, J. M., Gordon, J. S., Harris, C. E., Davison, C. L., Culpepper, D. K., Scott, P., & Johnson, K. B. (2011). MyMediHealth – Designing a next generation system for child-centered medication management.*Journal of Biomedical Informatics*, *43*(5), 27-31.

Stawarz, K., Cox, A. L., & Blandford, A. (2015). Beyond self-tracking and reminders: Designing smartphone apps that support habit formation. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 2653-2662. https://doi.org/10.1145/2702123.2702230

Tabi, K., Randhawa, A. S., Choi, F., Mithani, Z., Albers, F., Schnieder, M., Nikoo, M., Vigo, D., Jang, K., Demlova, R., & Krausz, M. (2019). Mobile Apps for Medication Management: Review and Analysis. *JMIR mHealth and uHealth*.

Zao, J. K., Wang, M. Y., Peihsuan, T., & Liu, J. W. S. (2010). Smart Phone Based Medicine Intake Scheduler, Reminder and Monitor. *IEEE e-Health Networking Applications and Services (Healthcom)*, 162-168.

APPENDIX

SOURCE CODE

auth_db_helper.dart

```
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';

class DatabaseHelper {
  // Singleton instance
  static final DatabaseHelper _instance = DatabaseHelper._internal();
  factory DatabaseHelper() => _instance;
  DatabaseHelper._internal();

  static Database? _db;
  Future<Database> get db async {
    if (_db != null) return _db!;
    _db = await initDb();
    return _db!;
  }

  // Initialize the database
  Future<Database> initDb() async {
    String databasePath = await getDatabasesPath();
    String path = join(databasePath, 'health_reminder.db');

    return await openDatabase(path, version: 1, onCreate: _onCreate);
  }

  // Create the User table
  void _onCreate(Database db, int version) async {
```

```
    await db.execute('''
      CREATE TABLE User(
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        name TEXT,
        age INTEGER,
        contact TEXT,
        email TEXT UNIQUE,
        password TEXT
      )
    ''');
  }

  // Insert a new user into the User table
  Future<int> insertUser(Map<String, dynamic> user) async {
    var dbClient = await db;
    return await dbClient.insert('User', user);
  }

  // Retrieve a user by email and password
  Future<Map<String, dynamic>?> getUser(String email, String password) async {
    var dbClient = await db;
    List<Map<String, dynamic>> result = await dbClient.query(
      'User',
      where: 'email = ? AND password = ?',
      whereArgs: [email, password],
    );
    if (result.isNotEmpty) {
      return result.first;
    }
    return null;
  }
}
```

reminder_dp_heler.dart

```
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';

class DbHelper {
  static late Database _db;
  static bool _isInitialized = false;

  static Future<void> initDb() async {
    if (!_isInitialized) {
      final dbPath = await getDatabasesPath();
      _db = await openDatabase(join(dbPath, 'my_reminders.db'),
          onCreate: (db, version) async {
        await db.execute('''
          CREATE TABLE my_reminders(
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            title TEXT,
            description TEXT,
            isActive INTEGER,
            remindersTime TEXT,
            category TEXT,
            repeatInterval TEXT,
            dailyTime TEXT
          )
        ''');
      }, version: 1);
      _isInitialized = true;
    }
  }
```

```
static Future<List<Map<String, dynamic>>> getReminders() async {
  await initDb();
  return await _db.query('my_reminders');
}

static Future<Map<String, dynamic>?> getRemindersById(int? id) async {
  await initDb();
  final List<Map<String, dynamic>> results = await _db.query(
    'my_reminders',
    where: 'id = ?',
    whereArgs: [id],
  );
  if (results.isNotEmpty) {
    return results.first;
  }
  return null;
}

static Future<int> addReminder(Map<String, dynamic> reminder) async {
  await initDb();
  return await _db.insert('my_reminders', reminder);
}

static Future<void> updateReminder(
    int id, Map<String, dynamic> reminder) async {
  await initDb();
  await _db.update(
    'my_reminders',
    reminder,
    where: 'id = ?',
    whereArgs: [id],
  );
```

```
  }

  static Future<void> deleteReminder(int id) async {
    await initDb();
    await _db.delete(
      'my_reminders',
      where: 'id = ?',
      whereArgs: [id],
    );
  }

  static Future<void> toggleReminder(int id, bool isActive) async {
    await initDb();
    await _db.update(
      'my_reminders',
      {'isActive': isActive ? 1 : 0},
      where: 'id = ?',
      whereArgs: [id],
    );
  }
}
```

Login.dart

```
import 'package:flutter/material.dart';
import 'package:med_reminder/database/auth_db_helper.dart';
import 'package:med_reminder/screens/home_screen.dart';
import 'registration_screen.dart';

class LoginScreen extends StatefulWidget {
  @override
  _LoginScreenState createState() => _LoginScreenState();
}
```

```
class _LoginScreenState extends State<LoginScreen> {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();

  DatabaseHelper dbHelper = DatabaseHelper();

  bool isLoading = false;

  // Handle user login
  void _login() async {
    if (_formKey.currentState!.validate()) {
      String email = _emailController.text;
      String password = _passwordController.text;

      var user = await dbHelper.getUser(email, password);
      if (user != null) {
        setState(() {
          isLoading = true;
        });
        await Future.delayed(const Duration(seconds: 5));
        Navigator.pushReplacement(context,
            MaterialPageRoute(builder: (context) => const HomeScreen()));
        setState(() {
          isLoading = true;
        });
      } else {
        setState(() {
          isLoading = true;
        });
        // Show error if credentials are invalid
```

```
      ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(content: Text('Invalid email or password')));
    }
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Login'),
      centerTitle: true,
      backgroundColor: Colors.deepPurple[100],
    ),
    body: Form(
      key: _formKey,
      child: Padding(
        padding: const EdgeInsets.symmetric(horizontal: 30),
        child: SingleChildScrollView(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              SizedBox(
                height: 70,
              ),
              TextFormField(
                controller: _emailController,
                decoration: const InputDecoration(
                  labelText: 'Email',
                  border: OutlineInputBorder(borderSide: BorderSide()),
                ),
                validator: (value) {
```

```
    if (value == null || value.isEmpty) {
      return 'Please enter your email';
    }
    return null;
  },
),
SizedBox(
  height: 20,
),
TextFormField(
  controller: _passwordController,
  decoration: const InputDecoration(
    labelText: 'Password',
    border: OutlineInputBorder(borderSide: BorderSide()),
  ),
  obscureText: true,
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your password';
    }
    return null;
  },
),
const SizedBox(height: 20),
ElevatedButton(
  onPressed: _login,
  child: isLoading
      ? const CircularProgressIndicator()
      : const Text(
          'Login',
          style: TextStyle(color: Colors.purple),
        ),
```

```
            ),
            TextButton(
              onPressed: () {
                Navigator.push(
                    context,
                    MaterialPageRoute(
                        builder: (context) => RegistrationScreen()));
              },
              child: const Text(
                "Don't have an account? Register here",
                style: TextStyle(color: Colors.purple),
              ),
            ),
          ],
        ),
      ),
    ),
  );
 }
}
```

register.dart

```
import 'package:flutter/material.dart';
import 'package:med_reminder/database/auth_db_helper.dart';
import 'login_screen.dart';

class RegistrationScreen extends StatefulWidget {
 @override
 _RegistrationScreenState createState() => _RegistrationScreenState();
}
```

```
class _RegistrationScreenState extends State<RegistrationScreen> {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController _nameController = TextEditingController();
  final TextEditingController _ageController = TextEditingController();
  final TextEditingController _contactController = TextEditingController();
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();

  DatabaseHelper dbHelper = DatabaseHelper();

  bool isLoading = false;

  // Handle user registration
  void _register() async {
    if (_formKey.currentState!.validate()) {
      String name = _nameController.text;
      int age = int.parse(_ageController.text);
      String contact = _contactController.text;
      String email = _emailController.text;
      String password = _passwordController.text;

      // Create user data map
      Map<String, dynamic> user = {
        'name': name,
        'age': age,
        'contact': contact,
        'email': email,
        'password': password,
      };

      int result = await dbHelper.insertUser(user);
```

```
    if (result > 0) {
      setState(() {
        isLoading = true;
      });
      await Future.delayed(const Duration(seconds: 5));
      ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(content: Text('Registration Successful')));
      Navigator.pushReplacement(
          context, MaterialPageRoute(builder: (context) => LoginScreen()));
      setState(() {
        isLoading = true;
      });
    } else {
      setState(() {
        isLoading = false;
      });
      ScaffoldMessenger.of(context)
          .showSnackBar(const SnackBar(content: Text('Registration Failed')));
    }
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Register'),
      centerTitle: true,
      backgroundColor: Colors.deepPurple[100],
    ),
    body: Form(
      key: _formKey,
```

```
child: Padding(
  padding: const EdgeInsets.symmetric(horizontal: 30),
  child: SingleChildScrollView(
    child: Column(
      children: [
        const SizedBox(height: 40),
        TextFormField(
          controller: _emailController,
          decoration: const InputDecoration(
            labelText: 'Email',
            border: OutlineInputBorder(borderSide: BorderSide()),
          ),
          validator: (value) {
            if (value == null || value.isEmpty) {
              return 'Please enter your email';
            }
            return null;
          },
        ),
        const SizedBox(height: 20),
        TextFormField(
          controller: _passwordController,
          decoration: const InputDecoration(
            labelText: 'Password',
            border: OutlineInputBorder(borderSide: BorderSide()),
          ),
          obscureText: true,
          validator: (value) {
            if (value == null || value.isEmpty) {
              return 'Please enter a password';
            }
            return null;
```

```
  },
),
const SizedBox(height: 20),
TextFormField(
  controller: _nameController,
  decoration: const InputDecoration(
    labelText: 'Name',
    border: OutlineInputBorder(borderSide: BorderSide()),
  ),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your name';
    }
    return null;
  },
),
const SizedBox(height: 20),
TextFormField(
  controller: _ageController,
  decoration: const InputDecoration(
    labelText: 'Age',
    border: OutlineInputBorder(borderSide: BorderSide()),
  ),
  keyboardType: TextInputType.number,
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your age';
    }
    return null;
  },
),
const SizedBox(height: 20),
```

```
            TextFormField(
              controller: _contactController,
              decoration: const InputDecoration(
                labelText: 'Contact Info',
                border: OutlineInputBorder(borderSide: BorderSide()),
              ),
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Please enter your contact information';
                }
                return null;
              },
            ),
            const SizedBox(height: 20),
            ElevatedButton(
              onPressed: _register,
              child: isLoading
                  ? const CircularProgressIndicator()
                  : const Text(
                      'Register',
                      style: TextStyle(color: Colors.purple),
                    ),
            ),
          ],
        ),
      ),
    ),
  );
 }
}
```

reminder_screen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_local_notifications/flutter_local_notifications.dart';
import 'package:med_reminder/database/db_helper.dart';
import
'package:med_reminder/screens/add_edit/add_edit_periodic_reminder_screen.dart';
import
'package:med_reminder/screens/add_edit/add_edit_schedule_reminder_screen.dart';
import
'package:med_reminder/screens/reminder_details/schedule_reminder_details_screen.dar
t';
import 'package:med_reminder/services/notification_helper.dart';
import 'package:med_reminder/services/permission_handler.dart';

class HomeScreen extends StatefulWidget {
  const HomeScreen({super.key});

  @override
  State<HomeScreen> createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  List<Map<String, dynamic>> reminderList = [];

  @override
  void initState() {
    super.initState();
    requestNotificationPermissions();
    NotificationHelper.initializeNotifications();
    _loadReminder();
  }
```

```
Future<void> _loadReminder() async {
  final reminder = await DbHelper.getReminders();
  setState(() {
    reminderList = reminder;
  });
}

Future<void> _toggleReminder(int id, bool isActive) async {
  await DbHelper.toggleReminder(id, isActive);
  if (isActive) {
    final reminder = reminderList.firstWhere((rem) => rem['id'] == id);
    if (reminder['repeatInterval'] != null) {
      NotificationHelper.scheduleRepeatedNotification(
        id: id,
        title: reminder['title'],
        body: reminder['category'],
        repeatInterval: _getRepeatInterval(reminder['repeatInterval']),
      );
    } else if (reminder['dailyTime'] != null) {
      NotificationHelper.scheduleDailyNotification(
        id: id,
        title: reminder['title'],
        body: reminder['category'],
        time: DateTime.parse(reminder['dailyTime']),
      );
    } else {
      NotificationHelper.scheduleNotification(
        id: id,
        title: reminder['title'],
        body: reminder['category'],
        scheduledDate: DateTime.parse(reminder['remindersTime']),
      );
```

```
    }
  } else {
    NotificationHelper.cancelNotification(id);
  }
  _loadReminder();
}

Future<void> _deleteReminder(int id) async {
  await DbHelper.deleteReminder(id);
  NotificationHelper.cancelNotification(id);
  _loadReminder();
}

RepeatInterval _getRepeatInterval(String interval) {
  switch (interval) {
    case 'every minute':
      return RepeatInterval.everyMinute;
    case 'every hour':
      return RepeatInterval.hourly;
    case 'daily':
      return RepeatInterval.daily;
    case 'weekly':
      return RepeatInterval.weekly;
    default:
      return RepeatInterval.daily;
  }
}

@override
Widget build(BuildContext context) {
  return PopScope(
    canPop: false,
```

```
child: Scaffold(
  backgroundColor: Colors.white,
  appBar: AppBar(
    centerTitle: true,
    automaticallyImplyLeading: false,
    backgroundColor: Colors.white,
    title: const Text(
      "Medical Reminder",
      style: TextStyle(color: Colors.purple),
    ),
    iconTheme: const IconThemeData(color: Colors.purple),
  ),
  floatingActionButton: Column(
    mainAxisAlignment: MainAxisAlignment.end,
    children: [
      SizedBox(
        width: 220,
        child: ElevatedButton(
          style: ElevatedButton.styleFrom(
            backgroundColor: Colors.purple,
            foregroundColor: Colors.white,
          ),
          onPressed: () {
            Navigator.push(context, MaterialPageRoute(builder: (context) {
              return const AddEditPeriodicReminderScreen();
            }));
          },
          child: const Row(
            children: [
              Icon(
                Icons.add,
                color: Colors.white,
```

```
        ),
        Text("Add Periodic Reminder"),
      ],
    ),
  ),
),
const SizedBox(
  height: 5,
),
SizedBox(
  width: 230,
  child: ElevatedButton(
    style: ElevatedButton.styleFrom(
      backgroundColor: Colors.purple,
      foregroundColor: Colors.white,
    ),
    onPressed: () {
      Navigator.push(context, MaterialPageRoute(builder: (context) {
        return const AddEditScheduleReminderScreen();
      }));
    },
    child: const Row(
      children: [
        Icon(
          Icons.add,
          color: Colors.white,
        ),
        Text("Add Scheduled Reminder"),
      ],
    ),
  ),
),
```

```
  ],
),
body: reminderList.isEmpty
    ? const Center(
        child: Text(
          "No Reminders Found",
          style: TextStyle(color: Colors.purple),
        ),
      )
    : ListView.builder(
        itemCount: reminderList.length,
        itemBuilder: (context, index) {
          final reminder = reminderList[index];
          return Dismissible(
              key: Key(reminder["id"].toString()),
              direction: DismissDirection.endToStart,
              background: Container(
                color: Colors.redAccent,
                padding: const EdgeInsets.only(right: 20),
                alignment: Alignment.bottomRight,
                child: const Align(
                  alignment: Alignment.centerRight,
                  child: Icon(
                    Icons.delete,
                    color: Colors.white,
                  ),
                ),
              ),
              confirmDismiss: (direction) async {
                return await _showDeleteConfirmationDialog(context);
              },
              onDismissed: (direction) {
```

```
  _deleteReminder(reminder["id"]);
  ScaffoldMessenger.of(context)
      .showSnackBar(const SnackBar(
    content: Text("Reminder Deleted"),
  ));
},
child: Card(
    color: Colors.teal.shade50,
    elevation: 6,
    margin: const EdgeInsets.symmetric(
        vertical: 8, horizontal: 16),
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(15),
    ), // RoundedRectangleBorder
    child: ListTile(
      onTap: () {
        Navigator.push(context,
            MaterialPageRoute(builder: (context) {
          return ReminderDetailsScreen(
            reminderId: reminder['id'],
          );
        }));
      },
      leading: const Icon(
        Icons.notifications,
        color: Colors.purple,
      ),
      title: Text(
        reminder['title'],
        style: const TextStyle(
          fontSize: 18,
          fontWeight: FontWeight.bold,
```

```
                color: Colors.black54,
              ),
            ),
            subtitle: Text(
              "Category: ${reminder['category']}",
            ),
            trailing: Switch(
              activeColor: Colors.purple,
              inactiveTrackColor: Colors.white,
              value: reminder['isActive'] == 1,
              onChanged: (value) {
                _toggleReminder(reminder['id'], value);
              },
            ),
          )));
        }),
    ),
  );
}

Future<bool?> _showDeleteConfirmationDialog(BuildContext context) {
  return showDialog<bool>(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        backgroundColor: Colors.white,
        title: const Text("Delete Reminder"),
        content: const Text("Are you sure you want to delete this reminder?"),
        actions: [
          TextButton(
            onPressed: () {
              Navigator.of(context).pop(false);
```

```
            },
            child: const Text("Cancel"),
          ),
          TextButton(
            onPressed: () {
              Navigator.of(context).pop(true);
            },
            child: const Text("Delete"),
          ), // TextButton
        ],
      ); // AlertDialog
    },
  );
 }
}
```

add_edit_reminder_screen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter/widgets.dart';
import 'package:flutter_local_notifications/flutter_local_notifications.dart';
// import 'package:intl/intl.dart';
import 'package:med_reminder/database/db_helper.dart';
import 'package:med_reminder/screens/home_screen.dart';
import 'package:med_reminder/services/notification_helper.dart';

class AddEditPeriodicReminderScreen extends StatefulWidget {
  final int? reminderId;
  const AddEditPeriodicReminderScreen({super.key, this.reminderId});

  @override
  State<AddEditPeriodicReminderScreen> createState() =>
```

```
    _AddEditPeriodicReminderScreenState();
}

class _AddEditPeriodicReminderScreenState
    extends State<AddEditPeriodicReminderScreen> {
  final formKey = GlobalKey<FormState>();

  final TextEditingController _titleController = TextEditingController();
  final TextEditingController _descriptionController = TextEditingController();

  @override
  void initState() {
    super.initState();
    if (widget.reminderId != null) {
      fetchReminder();
    }
  }

  String category = "Health";
  String? repeatInterval;
  DateTime? dailyTime;
  DateTime? scheduleTime;
  DateTime? scheduleDate;
  DateTime _reminderTime = DateTime.now();

  Future<void> fetchReminder() async {
    try {
      final data = await DbHelper.getRemindersById(widget.reminderId);
      if (data != null) {
        _titleController.text = data['title'];
        _descriptionController.text = data['description'];
        category = data['category'];
```

```
      _reminderTime = DateTime.parse(data['remindersTime']);
      repeatInterval = data['repeatInterval'];
      if (data['dailyTime'] != null) {
        dailyTime = DateTime.parse(data['dailyTime']);
      }
    }
  } catch (e) {}
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.white,
    appBar: AppBar(
      iconTheme: const IconThemeData(color: Colors.teal),
      backgroundColor: Colors.white,
      title: Text(
        widget.reminderId == null ? "Add Reminder" : "Edit Reminder",
        style: const TextStyle(
          color: Colors.purple,
        ), // TextStyle
      ), // Text
    ), // AppBar
    body: SingleChildScrollView(
      child: Padding(
          padding: const EdgeInsets.all(16),
          child: Form(
            key: formKey,
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                _buildInputCard(
```

```
  label: "Title",
  icon: Icons.title,
  child: TextFormField(
    controller: _titleController,
    maxLines: 1,
    decoration: const InputDecoration(
      hintText: "Enter Title",
      border: InputBorder.none,
    ), // InputDecoration
    validator: (value) {
      return value!.isEmpty ? "Please enter a title" : null;
    },
  ), // TextFormField
),
const SizedBox(height: 20),
_buildInputCard(
  label: "Description",
  icon: Icons.description,
  child: TextFormField(
    controller: _descriptionController,
    maxLines: 3,
    decoration: const InputDecoration(
      hintText: "Enter Description",
      border: InputBorder.none,
    ), // InputDecoration
    validator: (value) {
      return value!.isEmpty
          ? "Please enter a description"
          : null;
    },
  ), // TextFormField
),
```

```
const SizedBox(height: 20),
DropdownButtonFormField<String>(
  value: category,
  dropdownColor: Colors.purple.shade50,
  decoration: const InputDecoration.collapsed(hintText: ""),
  items: ["Health", "Maintenance", "Pills", "Others"]
      .map((String category) {
    return DropdownMenuItem<String>(
      value: category,
      child: Text(category),
    );
  }).toList(),
  onChanged: (String? newValue) {
    setState(() {
      category = newValue!;
    });
  },
),
const SizedBox(height: 20),
// _buildDateTimerPicker(
//   label: "Date",
//   icon: Icons.calendar_today,
//   displayValue: scheduleDate != null
//       ? DateFormat("yyyy-MM-dd").format(scheduleDate!)
//       : "Select Date",
//   onPressed: _selectDate,
// ),
// const SizedBox(height: 10),
// _buildDateTimerPicker(
//   label: "Time",
//   icon: Icons.access_time,
//   displayValue: scheduleTime != null
```

```
//       ? DateFormat("hh:mm a").format(scheduleTime!)
//       : "Select Time",
//   onPressed: _selectTime,
// ),
// const SizedBox(height: 20),
DropdownButtonFormField<String>(
  value: repeatInterval,
  hint: const Text("Select Repeat Interval"),
  dropdownColor: Colors.purple.shade50,
  decoration: const InputDecoration.collapsed(hintText: ""),
  items: [
    "every minute",
    "every hour",
    "daily",
    "weekly",
    "monthly"
  ].map((String interval) {
    return DropdownMenuItem<String>(
      value: interval,
      child: Text(interval),
    );
  }).toList(),
  onChanged: (String? newValue) {
    setState(() {
      repeatInterval = newValue!;
    });
  },
),
const SizedBox(height: 20),
// _buildDateTimerPicker(
//   label: "Daily Time",
//   icon: Icons.access_time,
```

```
            //   displayValue: dailyTime != null
            //       ? DateFormat("hh:mm a").format(dailyTime!)
            //       : "Select Time",
            //   onPressed: _selectDailyTime,
            // ),
            const SizedBox(height: 30),
            Center(
              child: ElevatedButton(
                onPressed: _saveReminder,
                style: ElevatedButton.styleFrom(
                    padding: const EdgeInsets.symmetric(
                        vertical: 15, horizontal: 20),
                    textStyle: const TextStyle(
                        fontSize: 16, fontWeight: FontWeight.bold),
                    foregroundColor: Colors.white,
                    backgroundColor: Colors.purple),
                child: const Text("Save Reminder"),
              ),
            )
          ],
        ), // Column
      )), // Form // Padding
    ), // SingleChildScrollView
  ); // Scaffold
}

Widget _buildInputCard(
    {required String label, required IconData icon, required Widget child}) {
  return Card(
    elevation: 6,
    color: Colors.teal.shade50,
    shape: RoundedRectangleBorder(
```

```
      borderRadius: BorderRadius.circular(15),
    ),
    child: Padding(
      padding: const EdgeInsets.symmetric(vertical: 10, horizontal: 16),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Row(
            children: [
              Icon(icon, color: Colors.purple),
              const SizedBox(width: 10),
              Text(
                label,
                style: const TextStyle(fontWeight: FontWeight.bold),
              ),
            ],
          ),
          child
        ],
      ),
    ),
  );
}

// Widget _buildDateTimerPicker({
//   required String label,
//   required IconData icon,
//   required String displayValue,
//   required Function() onPressed,
// }) {
//   return Card(
//     elevation: 6,
```

```
//       color: Colors.purple.shade50,
//       shape: RoundedRectangleBorder(
//         borderRadius: BorderRadius.circular(15),
//       ),
//       child: ListTile(
//           leading: Icon(icon, color: Colors.purple),
//           title: Text(
//             label,
//             style: const TextStyle(
//               fontWeight: FontWeight.bold,
//             ),
//           ),
//           trailing: TextButton(
//             onPressed: onPressed,
//             child: Text(
//               displayValue,
//               style: const TextStyle(color: Colors.purple),
//             ),
//           )));
// }

// Future<void> _selectDate() async {
//   DateTime? picked = await showDatePicker(
//     context: context,
//     initialDate: _reminderTime,
//     firstDate: DateTime(2000),
//     lastDate: DateTime(2100),
//   );
//   if (picked != null) {
//     setState(() {
//       scheduleDate = DateTime(
//         picked.year,
```

```
//         picked.month,
//         picked.day,
//         _reminderTime.hour,
//         _reminderTime.minute,
//       );
//     });
//   }
// }

// Future<void> _selectTime() async {
//   TimeOfDay? picked = await showTimePicker(
//     context: context,
//     initialTime:
//         TimeOfDay(hour: _reminderTime.hour, minute: _reminderTime.minute),
//   );
//   if (picked != null) {
//     setState(() {
//       scheduleTime = DateTime(
//         _reminderTime.year,
//         _reminderTime.month,
//         _reminderTime.day,
//         picked.hour,
//         picked.minute,
//       );
//     });
//   }
// }

// Future<void> _selectDailyTime() async {
//   TimeOfDay? picked = await showTimePicker(
//     context: context,
//     initialTime: dailyTime != null
```

```
//         ? TimeOfDay(hour: dailyTime!.hour, minute: dailyTime!.minute)
//         : TimeOfDay.now(),
//   );
//   if (picked != null) {
//     setState(() {
//       dailyTime = DateTime(
//         _reminderTime.year,
//         _reminderTime.month,
//         _reminderTime.day,
//         picked.hour,
//         picked.minute,
//       );
//     });
//   }
// }

Future<void> _saveReminder() async {
  if (formKey.currentState!.validate()) {
    final newReminder = {
      'title': _titleController.text,
      'description': _descriptionController.text,
      'isActive': 1,
      'remindersTime': scheduleTime != null
          ? DateTime(
              _reminderTime.year,
              _reminderTime.month,
              _reminderTime.day,
              scheduleTime!.hour,
              scheduleTime!.minute,
            ).toIso8601String()
          : null,
      'category': category,
```

```
  'repeatInterval': repeatInterval,
  'dailyTime': dailyTime != null
      ? DateTime(
          _reminderTime.year,
          _reminderTime.month,
          _reminderTime.day,
          dailyTime!.hour,
          dailyTime!.minute,
        ).toIso8601String()
      : null,
};
if (widget.reminderId == null) {
  final reminderId = await DbHelper.addReminder(newReminder);
  NotificationHelper.scheduleRepeatedNotification(
    id: reminderId,
    title: _titleController.text,
    body: category,
    repeatInterval: _getRepeatInterval(repeatInterval!),
  );
  // if (repeatInterval != null) {
  //   NotificationHelper.scheduleRepeatedNotification(
  //     id: reminderId,
  //     title: _titleController.text,
  //     body: category,
  //     repeatInterval: _getRepeatInterval(repeatInterval!),
  //   );
  // } else if (dailyTime != null) {
  //   NotificationHelper.scheduleDailyNotification(
  //     id: reminderId,
  //     title: _titleController.text,
  //     body: category,
  //     time: dailyTime!,
```

```
//   );
// } else {
//   NotificationHelper.scheduleNotification(
//     id: reminderId,
//     title: _titleController.text,
//     body: category,
//     scheduledDate: scheduleTime!,
//   );
// }
} else {
  await DbHelper.updateReminder(widget.reminderId!, newReminder);
  if (repeatInterval != null) {
    NotificationHelper.scheduleRepeatedNotification(
      id: widget.reminderId!,
      title: _titleController.text,
      body: category,
      repeatInterval: _getRepeatInterval(repeatInterval!),
    );
  } else if (dailyTime != null) {
    NotificationHelper.scheduleDailyNotification(
      id: widget.reminderId!,
      title: _titleController.text,
      body: category,
      time: dailyTime!,
    );
  } else {
    NotificationHelper.scheduleNotification(
      id: widget.reminderId!,
      title: _titleController.text,
      body: category,
      scheduledDate: _reminderTime,
    );
```

```
      }
    }
    Navigator.push(context, MaterialPageRoute(builder: (context) {
      return const HomeScreen();
    }));
  }
 }

 RepeatInterval _getRepeatInterval(String interval) {
  switch (interval) {
    case 'every minute':
      return RepeatInterval.everyMinute;
    case 'every hour':
      return RepeatInterval.hourly;
    case 'daily':
      return RepeatInterval.daily;
    case 'weekly':
      return RepeatInterval.weekly;
    default:
      return RepeatInterval.daily;
  }
 }
}
```

reminder_details.dart

```
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'package:med_reminder/database/db_helper.dart';
import
'package:med_reminder/screens/add_edit/add_edit_periodic_reminder_screen.dart';
```

```
class ReminderDetailsScreen extends StatefulWidget {
  final int? reminderId;
  const ReminderDetailsScreen({super.key, this.reminderId});

  @override
  State<ReminderDetailsScreen> createState() => _ReminderDetailsScreenState();
}

class _ReminderDetailsScreenState extends State<ReminderDetailsScreen> {
  @override
  Widget build(BuildContext context) {
    return FutureBuilder<Map<String, dynamic>?>(
      future: DbHelper.getRemindersById(widget.reminderId),
      builder: (context, snapshot) {
        if (!snapshot.hasData) {
          return const Scaffold(
            body: Center(
              child: CircularProgressIndicator(
                color: Colors.purple,
              ),
            ),
          );
        }
        final reminder = snapshot.data!;
        return Scaffold(
          backgroundColor: Colors.white,
          appBar: AppBar(
            backgroundColor: Colors.white,
            title: const Text(
              'Reminder Details',
              style: TextStyle(color: Colors.purple),
```

```
  ),
  centerTitle: true,
),
body: Padding(
  padding: const EdgeInsets.all(16),
  child: Column(
    children: [
      _buildDetailsCard(
        label: 'Title',
        icon: Icons.title,
        content: reminder['title'],
      ),
      _buildDetailsCard(
        label: 'Description',
        icon: Icons.description,
        content: reminder['description'],
      ),
      _buildDetailsCard(
        label: 'Category',
        icon: Icons.category,
        content: reminder['category'],
      ),
      if (reminder['remindersTime'] != null)
        _buildDetailsCard(
          label: 'Reminder Time',
          icon: Icons.access_time,
          content: DateFormat("yyyy-MM-dd hh:mm a")
              .format(DateTime.parse(reminder['remindersTime'])),
        ),
      if (reminder['repeatInterval'] != null)
        _buildDetailsCard(
          label: 'Repeat Interval',
```

```
                icon: Icons.repeat,
                content: reminder['repeatInterval'],
              ),
            if (reminder['dailyTime'] != null)
              _buildDetailsCard(
                label: 'Daily Time',
                icon: Icons.access_time,
                content: DateFormat("hh:mm a")
                    .format(DateTime.parse(reminder['dailyTime'])),
              ),
          ],
        ),
      ),
      // floatingActionButton: FloatingActionButton(
      //   foregroundColor: Colors.white,
      //   backgroundColor: Colors.purple,
      //   onPressed: () {
      //     Navigator.push(context, MaterialPageRoute(builder: (context) {
      //       return AddEditReminderScreen(
      //         reminderId: widget.reminderId,
      //       );
      //     }));
      //   },
      //   child: const Icon(Icons.edit),
      // ),
    );
  },
 );
}

Widget _buildDetailsCard(
  {required String label,
```

```
    required IconData icon,
    required String content}) {
  return Card(
    elevation: 6,
    color: Colors.teal.shade50,
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(15),
    ),
    child: Padding(
      padding: const EdgeInsets.symmetric(vertical: 10, horizontal: 16),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Row(
            children: [
              Icon(icon, color: Colors.purple),
              const SizedBox(width: 10),
              Text(
                label,
                style: const TextStyle(
                    fontWeight: FontWeight.bold, fontSize: 18),
              ),
            ],
          ),
          Text(content,
              style: const TextStyle(fontSize: 16, color: Colors.black54))
        ],
      ),
    ),
  );
 }
}
```