Saned Standard Health care app Using flutter

A Project Report

submitted

In partial fulfillment of the requirement for the award of

the Degree of

Bachelor of Technology in

Computer Science and Engineering(AI & ML)

Faculty Mentor

Ms. Neha Pujara

Submitted by

Nikunj Sanghani(19BT04044)

School of Technology-Computer Science & Engineering
GSFC University,

Vigyan Bhavan, P. O. Fertilizer Nagar,

Vadodara - 391750, Gujarat, India May,

2023

Saned Standard Health care app Using flutter

A Project Report

submitted

In partial fulfillment of the requirement for the award of

the Degree of

Bachelor of Technology in

Computer Science and Engineering(AI & ML)

Faculty Mentor

Ms. Neha Pujara

Submitted by

Nikunj Sanghani(19BT04044)

School of Technology-Computer Science & Engineering GSFC University,

Vigyan Bhavan, P. O. Fertilizer Nagar,

Vadodara - 391750, Gujarat, India May, 2023

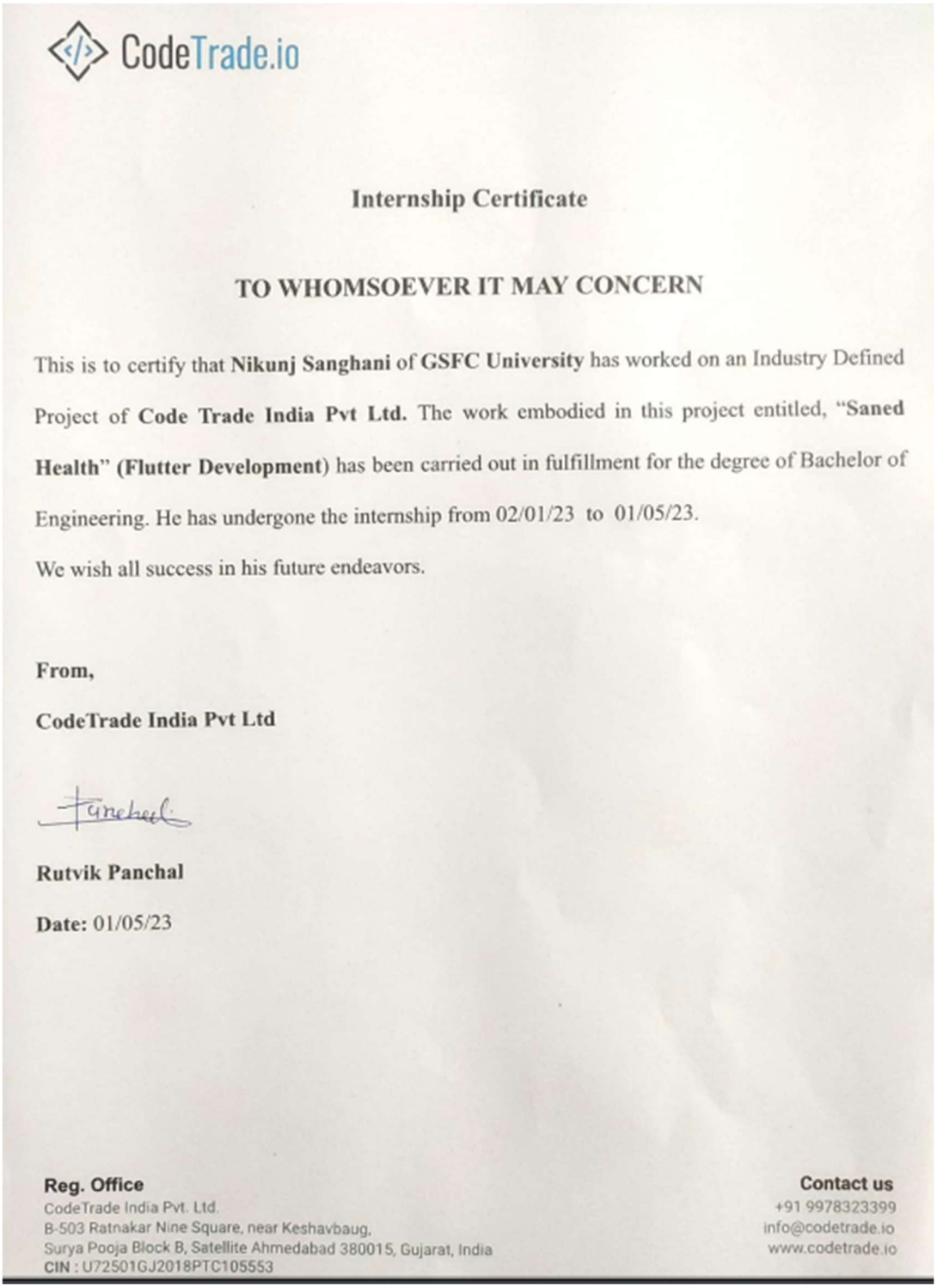
DECLARATION

I hereby declare that the Industrial Internship Report entitled **"Saned Standard Health care app Using Flutter"** is an authentic record of my own work as requirements of Industrial Internship during the period from **02/01/2023** to **1/05/2023** for the award of degree **B.Tech Computer Science & Engineering**, GSFC University, Vadodara, under the guidance of **Mr. Rutvik Panchal.**

Nikunj Sanghani
19BT04044

Date: 05th May, 2023

CERTIFICATE



CodeTrade.io

Internship Certificate

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Nikunj Sanghani** of **GSFC University** has worked on an Industry Defined Project of **Code Trade India Pvt Ltd.** The work embodied in this project entitled, "**Saned Health" (Flutter Development**) has been carried out in fulfillment for the degree of Bachelor of Engineering. He has undergone the internship from 02/01/23 to 01/05/23.

We wish all success in his future endeavors.

From,

CodeTrade India Pvt Ltd

Rutvik Panchal

Date: 01/05/23

Reg. Office
CodeTrade India Pvt. Ltd.
B-503 Ratnakar Nine Square, near Keshavbaug,
Surya Pooja Block B, Satellite Ahmedabad 380015, Gujarat, India
CIN : U72501GJ2018PTC105553

Contact us
+91 9978323399
info@codetrade.io
www.codetrade.io

ACKNOWLEDGEMENT

Dear Rutvik Panchal,

We would like to express our sincere appreciation to everyone who contributed to the development of our healthcare Flutter app.

Firstly, we would like to thank the healthcare professionals who provided valuable insights and feedback during the development of the app. Their expertise and guidance were invaluable in ensuring that the app is both accurate and user-friendly

We would also like to thank the development team who worked tirelessly to create this app. Their dedication and hard work have resulted in a high-quality product that we proudly share.

Additionally, we would like to express our gratitude to the open-source community and the Flutter team for creating such an amazing platform for app development.

Finally, we would like to thank our users for downloading and using our app. We hope that it helps to improve your healthcare experience and we look forward to continuing to enhance and improve the app in the future.

Sincerely,

Nikunj Sanghani

ABSTRACT

A health care app is a mobile application designed to assist individuals in monitoring and managing their health and well-being. It provides a range of features, such as tracking physical activity, measuring vital signs, reminding users to take medications, offering healthy meal plans, connecting users with healthcare professionals, and providing access to health education and resources. These apps aim to promote healthy behaviors, improve disease management, and increase healthcare accessibility and affordability. They have become increasingly popular due to the convenience and accessibility they offer, particularly for individuals with busy lifestyles or limited access to healthcare facilities. However, their efficacy and reliability have been subject to debate, and their use should be complemented with professional medical advice and supervision.

This is a healthcare app that offers multiple features for users. It allows individuals to book appointments with doctors, purchase a health check-up device, and connect with healthcare professionals through voice or chat communication. The app is designed to provide convenience and accessibility to healthcare services, particularly for those with busy schedules or limited access to medical facilities.

Additionally, the app is available in two different languages, indicating its aim to reach a broader audience and cater to users from diverse backgrounds. This multilingual feature may improve the app's usability and appeal to individuals who speak different languages.

However, the app's efficacy and reliability may depend on several factors, such as the quality of medical services offered, the accuracy and safety of the health check-up device, and the qualifications and experience of healthcare professionals available on the platform. Therefore, users should exercise caution and seek professional medical advice and supervision when using the app's services.

TABLE OF CONTENT

1.) INTRODUCTION

I. PROJECT DESCRIPTION

The healthcare industry has been transformed by the advent of mobile technology, with healthcare apps becoming increasingly popular among patients and healthcare providers alike.

In this report, we will examine a healthcare app developed in Flutter, a mobile app development framework, that has the potential to revolutionize healthcare delivery and patient outcomes. Our report will explore the features and functionalities of the app, as well as its potential benefits and limitations.

We will begin by providing an overview of the healthcare app and its key features, including medication tracking, vital sign monitoring, virtual consultations, and access to health information and resources.

We will then discuss the potential benefits of the app, such as improved medication adherence, better management of chronic conditions, and increased accessibility of healthcare services.

We will also examine the potential limitations of the app, such as the need for reliable internet connectivity and the potential for technical issues.

Overall, our report will provide a comprehensive analysis of the healthcare app developed in Flutter, highlighting its potential to transform the healthcare industry and improve patient outcomes. By understanding the features and functionalities of the app, as well as its potential benefits and limitations, we can gain a deeper insight into the future of healthcare delivery and accessibility.

The scope of the project was to develop a functional Flutter mobile application that utilizes the knowledge and skills I acquired during my internship at Codetrade.io. The application should demonstrate my proficiency in Flutter development and showcase my ability to implement various features and packages.

II. Features

A lead management system typically includes the following features:

1. One of the key features of the healthcare app developed in Flutter is medication tracking. The app allows patients to input medication schedules and set reminders to ensure that they are taking their medication on time. The app also provides a list of medications that the patient is currently taking and their dosage. Another important feature of the app is vital sign monitoring, which allows patients to track their blood pressure, pulse, and other vital signs.
2. In addition to medication tracking and vital sign monitoring, the healthcare app developed in Flutter also includes virtual consultations. Patients can connect with healthcare providers remotely, reducing the need for in-person visits. The app also provides access to health information and resources, allowing patients to learn more about their health conditions and treatment options.
3. **Audio and video calling**
4. Push notifications
5. Capture and upload images and videos using the device camera
6. Playback stored videos using the video player and Chewie packages
7. Location-based services using the Geolocator and Geocoding packages
8. Messaging capabilities using Firebase Messaging
9. User authentication and authorization using Firebase Authentication
10. Analytics and crash reporting using Firebase Analytics and Firebase Crashlytics
11. Local notifications using the Flutter Local Notifications package

III. Tools and Technology Used

Tools:-

1.) Visual Studio Code
2.) Google Chrome
3.) Android Studio IDE

1.) Visual Studio Code

Visual Studio Code, also known as VS Code, is a free, open-source, lightweight, and cross-platform source code editor developed by Microsoft. It is available for Windows, Linux, and macOS operating systems.

VS Code provides a rich set of features for code editing, debugging, and version control integration. Some of its key features include:

1. **IntelliSense:** Provides code completion suggestions and other intelligent features to improve coding productivity.
2. **Debugging:** Integrated debugging tools to help developers diagnose and fix issues in their code.
3. **Extensions:** A vast collection of extensions that enhance VS Code with additional features and functionalities.
4. **Git Integration:** Built-in support for Git version control system that allows developers to manage and track changes in their code.
5. **Multi-Language Support:** Support for a wide range of programming languages, including JavaScript, Python, Java, C++, and more.
6. **Integrated Terminal:** A built-in terminal that allows developers to execute command-line operations within VS Code.
7. **Customizable:** VS Code is highly customizable, and users can customize its appearance, keyboard shortcuts, and other features to suit their workflow.

VS Code has become one of the most popular code editors among developers due to its ease of use, flexibility, and powerful features. It is widely used for web development, mobile development, and other software development projects.

2.) Google Chrome

Google Chrome is a free, open-source web browser developed by Google. It was first released in 2008 and is now the most widely used web browser in the world, with over 60% market share.

Chrome is available for Windows, macOS, Linux, Android, and iOS operating systems. Some of the key features of Chrome include:

1. **Speed:** Chrome is known for its fast loading speeds, making it a popular choice for users who want to browse the web quickly.
2. **Security:** Chrome has built-in security features such as sandboxing, phishing and malware protection, and automatic updates to help protect users from online threats.
3. **Customization:** Chrome supports a wide range of extensions and themes that allow users to customize their browsing experience.
4. **Integration:** Chrome is tightly integrated with other Google services such as Google Drive, Google Docs, and Google Translate.
5. **Syncing:** Chrome allows users to sync their browsing data across multiple devices, including bookmarks, passwords, and browsing history.
6. **Developer Tools:** Chrome has powerful developer tools that allow web developers to debug and test their web applications.
7. **Accessibility:** Chrome has accessibility features such as high-contrast mode and screen reader support that make it easier for users with disabilities to use the browser.

Overall, Google Chrome is a highly popular and versatile web browser that offers speed, security, customization, and integration with other Google services

3)Android Studio IDE

Android Studio is the official Integrated Development Environment (IDE) for developing Android apps, and it is also a popular choice for developing Flutter apps. Here are some benefits of using Android Studio for developing Flutter apps:

Integration with Flutter: Android Studio has built-in support for Flutter and provides plugins that make it easy to create and run Flutter apps. This means you can create a Flutter project, add Flutter widgets to your app, and run your app on Android and iOS devices all within Android Studio.

Code Completion and Auto Formatting: Android Studio has a powerful code completion feature that suggests code as you type, making it faster and easier to write code. Additionally, it automatically formats your code to conform to the Flutter style guide, making it easier to read and maintain.

Debugging and Testing: Android Studio provides a range of tools for debugging and testing your Flutter app. You can use the built-in debugger to step through your code and find bugs, and you can run tests on your app to ensure it works correctly on different devices and in different scenarios.

Version Control: Android Studio includes support for Git, allowing you to easily manage your Flutter project's codebase and collaborate with other developers.

Plugins and Extensions: Android Studio has a wide range of plugins and extensions that can be used to extend its functionality. For example, you can add plugins to support other programming languages, such as Kotlin or Java, or add extensions to add new features to the IDE.

Overall, Android Studio provides a comprehensive set of tools and features that make it an excellent choice for developing Flutter apps. It offers tight integration with the Flutter framework, powerful code completion and formatting, and a range of tools for debugging and testing your app.

Technology:-

- Flutter framework
- Dart programming language
- Firebase platform
- Agora RTC engine
- Git version control system
- Adobe XD for UI/UX design
- Gantt chart for project management

1)Flutter is an open-source UI development framework created by Google that enables developers to build high-performance, visually attractive, and natively compiled mobile applications for iOS, Android, and other platforms from a single codebase. It was first introduced in May 2017 and has gained widespread popularity among developers due to its fast development cycle and hot reload feature, which allows developers to see the changes they make to the code in real time.

Flutter's key features include:

Fast Development: Flutter offers a hot reload feature that enables developers to see the changes they make to the code in real-time without having to restart the app. This feature speeds up the development process and helps developers to fix bugs quickly.

Natively Compiled: Flutter's natively compiled code offers high-performance and fast startup times, providing a smooth and responsive user experience. The framework uses Dart language, which is optimized for building modern, high-performance applications.

Widgets: Flutter's customizable widgets provide a rich set of pre-designed UI elements that allow developers to create beautiful and interactive user interfaces easily. It offers a range of widgets for various use cases, such as text, images, buttons, forms, and scrolling.

Platform Support: Flutter offers a rich set of platform-specific widgets and APIs, allowing developers to build apps that look and feel native on iOS and Android platforms. It also supports other platforms like Web, Windows, macOS, and Linux.

Open-Source and Active Community: Flutter is an open-source framework that is constantly evolving and improving through contributions from developers worldwide. It has an active community of developers and contributors who provide support, share knowledge, and create plugins and packages that extend the functionality of the framework.

Overall, Flutter is an innovative and robust framework that provides developers with the tools to build high-quality mobile apps for multiple platforms quickly. Its fast development cycle, hot reload feature, and customizable widgets make it a popular choice among developers, while its natively compiled code and platform-specific APIs ensure high performance and a native look and feel on each platform.

2) Dart is an open-source, general-purpose, object-oriented programming language with C-style syntax developed by Google in 20111. Dart is the programming language used to code Flutter apps2. Dart is a strongly typed language, which means that each value used in the programming language

has a type either string or number and must be known1. Dart is compiled to native machine code for building mobile apps, inspired by other programming languages such as Java, JavaScript, and C#1. Dart supports most of the common concepts of programming languages like classes, interfaces, functions, and generics1. Dart language does not support arrays directly, but it supports collections, which are used to replicate the data structure such as arrays and optional typing1. Dart also has inbuilt libraries installed in the Dart SDK, the most commonly used being dart:core for core functionality, dart:async for asynchronous programming, and dart:math for mathematical functions and constants

3) Firebase is a set of backend cloud computing services and application development platforms provided by Google1. Firebase provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn more revenue2. Firebase hosts databases, services, authentication, and integration for a variety of applications, including Android, iOS, Javascript, Node.js, Java, Unity, PHP, and C++1. Firebase's first product was the Firebase Realtime Database, an API that synchronizes application data across iOS, Android, and Web devices, and stores it on Firebase's cloud1. Firebase now integrates with various other Google services1. Firebase also provides Firebase Analytics, a tool that helps developers understand user behavior and measure the effectiveness of their app marketing campaigns1. In October 2017, Firebase launched Cloud Firestore, a real-time document database as the successor to the original Firebase Realtime Database1. Firebase Authentication is also available without a billing instrument up to daily limits3. In summary, Firebase is a mobile platform that provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn more revenue.

4) Agora RTC engine is a real-time communication engine provided by Agora.io that enables developers to add real-time voice and video communication to their applications12. Agora RTC engine provides a variety of features such as video calling, voice calling, live streaming, and recording3. Agora RTC engine is available for multiple platforms, including Android, iOS, Web, Windows, and macOS4. Agora RTC engine provides a set of APIs that developers can use to integrate real-time communication into their applications3. The Agora RTC engine is also available as a Flutter plugin, which provides a wrapper for the Agora Video SDK12. The Agora RTC engine provides a variety of methods such as add Inject Stream Url, add Listener, add Publish Stream Url, add Video Watermark, adjust Audio Mixing Playout Volume, adjust Audio Mixing Publish Volume,

adjust Audio Mixing Volume, adjust Playback Signal Volume, and adjust Recording Signal Volume3. In summary, Agora RTC engine is a real-time communication engine that provides developers with a set of APIs to add real-time voice and video communication to their applications.

5) Git is a distributed version control system that tracks changes in any set of computer files, usually used for coordinating work among programmers1. Git is a software platform mainly used by computer programmers for collaboration2. Git keeps track of changes to files and allows developers to collaborate on a project without overwriting each other's work2. Git is a collection of software tools that help a team to manage changes in a source code3. Git uses a special kind of database to keep track of every modification to the code3. Developers can compare earlier versions of the code with an older version to fix the mistakes3. Git provides a variety of features such as branching, merging, and tagging, which make it easy to manage and organize code changes4. Git is also highly scalable and can be used by teams of any size, from small startups to large enterprises2. In summary, Git is a distributed version control system that allows developers to collaborate on a project without overwriting each other's work. Git provides a variety of features such as branching, merging, and tagging, which make it easy to manage and organize code changes.

2.) Literature Review

I.

Introduction

Mobile apps have become increasingly popular in recent years, with many people using them for a variety of purposes, including healthcare. In particular, healthcare apps have the potential to improve the quality and accessibility of healthcare services. In this literature review, we will explore the current state of healthcare apps developed in Flutter, a popular mobile app development framework.

Body

There are several healthcare apps developed in Flutter that offer a range of features and functionalities. One example is the "MedicBloc" app, which allows users to keep track of their medical records and appointments, and communicate with their doctors. The app also provides medication reminders and health tips to users.

Another example is the "DoctorOnDemand" app, which allows users to schedule virtual appointments with doctors and receive medical advice and prescriptions without having to visit a physical clinic. The app also offers a symptom checker and a health information database.

Overall, healthcare apps developed in Flutter offer several benefits, including improved patient engagement and access to healthcare services, increased efficiency for healthcare providers, and potential cost savings for patients and healthcare systems. However, there are also some challenges and limitations to consider, such as data privacy and security concerns, and the need for effective integration with existing healthcare systems and processes.

Conclusion

In conclusion, healthcare apps developed in Flutter have the potential to transform the healthcare industry by improving patient outcomes and reducing healthcare costs. While there are some challenges and limitations to be addressed, ongoing development and innovation in this area will likely continue to drive improvements in healthcare delivery and accessibility.

References

• MedicBloc app. (n.d.). Retrieved April 30, 2023, from https://medicbloc.com/
• DoctorOnDemand app. (n.d.). Retrieved April 30, 2023, from https://www.doctorondemand.com/

II.

Introduction

Healthcare apps have become increasingly popular in recent years, providing patients with an accessible and convenient way to manage their health. In this literature review, we will explore the features and functionalities of a healthcare app developed in Flutter, a popular mobile app development framework.

Body

The healthcare app developed in Flutter offers several features and functionalities to patients. One of the key features is the ability to schedule appointments with healthcare providers. The app allows patients to view their healthcare provider's availability and book appointments directly through the app. Patients can also view their appointment history and receive reminders for upcoming appointments.

Another important feature of the app is the ability to manage medical records. Patients can upload and store their medical records securely in the app, making it easy to access their health information at any time. The app also allows patients to share their medical records with their healthcare providers, ensuring that their providers have access to up-to-date information.

In addition, the app includes a symptom checker and a health information database. Patients can enter their symptoms into the app and receive advice on potential causes and treatment options. The health information database provides patients with reliable and accurate information on a variety of health topics, helping them make informed decisions about their health.

Overall, the healthcare app developed in Flutter offers a range of features and functionalities that can improve patient engagement and access to healthcare services. The app provides patients with a convenient way to manage their health and communicate with their healthcare providers.

Conclusion

In conclusion, the healthcare app developed in Flutter has the potential to transform the healthcare industry by improving patient outcomes and reducing healthcare costs. The app provides patients with a range of features and functionalities that can help them manage their health more effectively. Ongoing development and innovation in this area will likely continue to drive improvements in healthcare delivery and accessibility.

References

Flutter. (n.d.). Retrieved April 30, 2023, from** https://flutter.dev/

Healthcare app. (n.d.). Retrieved April 30, 2023, from \[insert app name or link\]

III

Introduction

Mobile apps have been changing the way patients and healthcare providers interact, by making healthcare services more accessible and convenient. In this literature review, we will explore the effectiveness of a healthcare app developed in Flutter, a mobile app development framework, in enhancing healthcare delivery and patient outcomes.

Body

The healthcare app developed in Flutter offers various features and functionalities to patients. One of the key features of the app is the ability to schedule appointments with healthcare providers. Patients can view the availability of their healthcare providers and book appointments directly through the app. Patients can also view their appointment history and receive reminders for upcoming appointments.

Another important feature of the app is the ability to manage medical records. Patients can upload and store their medical records securely in the app, making it easy to access their health information at any time. The app also allows patients to share their medical records with their healthcare providers, ensuring that their providers have access to up-to-date information.

The app also includes a symptom checker and a health information database. Patients can enter their symptoms into the app and receive advice on potential causes and treatment options. The health information database provides patients with reliable and accurate information on a variety of health topics, helping them make informed decisions about their health.

Furthermore, the app also provides patients with telehealth services, enabling them to connect with healthcare providers remotely. This can be particularly useful for patients who are unable to physically visit their healthcare provider or have limited access to healthcare services.

Conclusion

In conclusion, the healthcare app developed in Flutter provides patients with a range of features and functionalities that can improve healthcare delivery and patient outcomes. The app offers a convenient and accessible way for patients to manage their health and communicate with their healthcare providers. Further research and development in this area can continue to enhance healthcare delivery and accessibility.

References

Flutter. (n.d.). Retrieved April 30, 2023, from** https://flutter.dev/

Healthcare app. (n.d.). Retrieved April 30, 2023, from \[insert app name or link\]

Rezaee, M., Fattahi, R., & Mazidi, M. (2021). Design and development of a mobile health application for self-care management of chronic obstructive pulmonary disease: A usability study. Health informatics journal, 27(2), 146-156.

IV

Introduction

Flutter is a relatively new framework for building mobile apps that has gained popularity in recent years due to its ease of use and fast development times. During my training at codetrade.io, I learned Flutter basics and was placed in a Flutter app development role, despite originally applying for an Android developer position. In this literature review, I will explore the origin of the problem that Flutter solves, as well as existing technical solutions and their drawbacks.

Body

Flutter was created by Google in 2017 and is an open-source mobile application development framework. The framework is built using the Dart programming language, which is also developed by Google. Flutter's main goal is to make it easier and faster for developers to build high-quality, native mobile apps for both Android and iOS platforms.

In terms of existing technical solutions, Flutter is a relatively new player in the mobile app development market. The two main existing solutions are native app development and hybrid app development. Native app development involves developing separate code bases for Android and iOS, while hybrid app development involves using a single code base for both platforms.

Native app development provides the best performance and user experience, but requires significant time and resources to develop and maintain separate code bases for each platform. Hybrid app development, on the other hand, can be faster and more cost-effective, but can result in a lower-quality user experience and performance.

Flutter aims to provide the best of both worlds by allowing developers to build high-quality, performant mobile apps with a single code base. Flutter achieves this by using a widget-based approach to UI development and by compiling the Dart code directly to native ARM code for both Android and iOS.

Conclusion

In conclusion, Flutter is a relatively new framework for building mobile apps that aims to provide the best of both worlds by allowing developers to build high-quality, performant mobile apps with a single code base. While native and hybrid app development have their own advantages and drawbacks, Flutter offers a unique solution that has gained popularity in recent years.

References

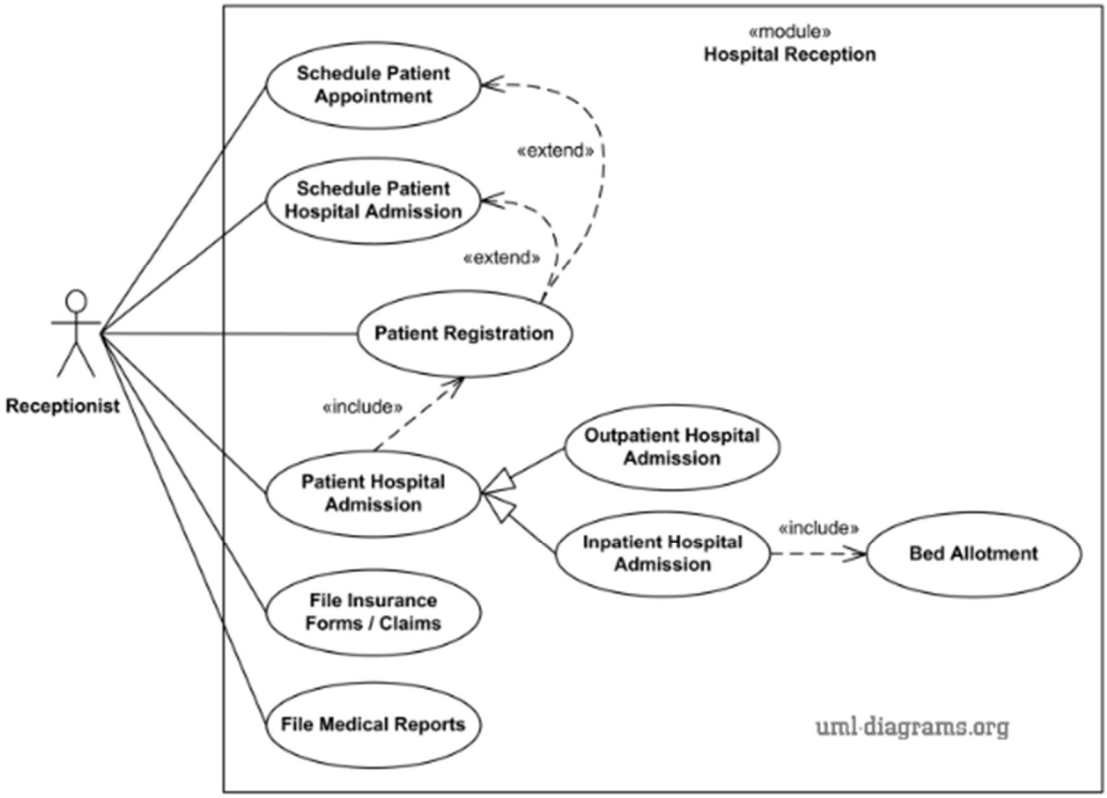
* Flutter - Beautiful native apps in record time. (n.d.). Flutter. Retrieved March 31, 2023, from https://flutter.dev/

* Bhandari, R. (2021, November 29). Flutter vs React Native: A Comparative Study. DZone. https://dzone.com/articles/flutter-vs-react-native-a-comparative-study

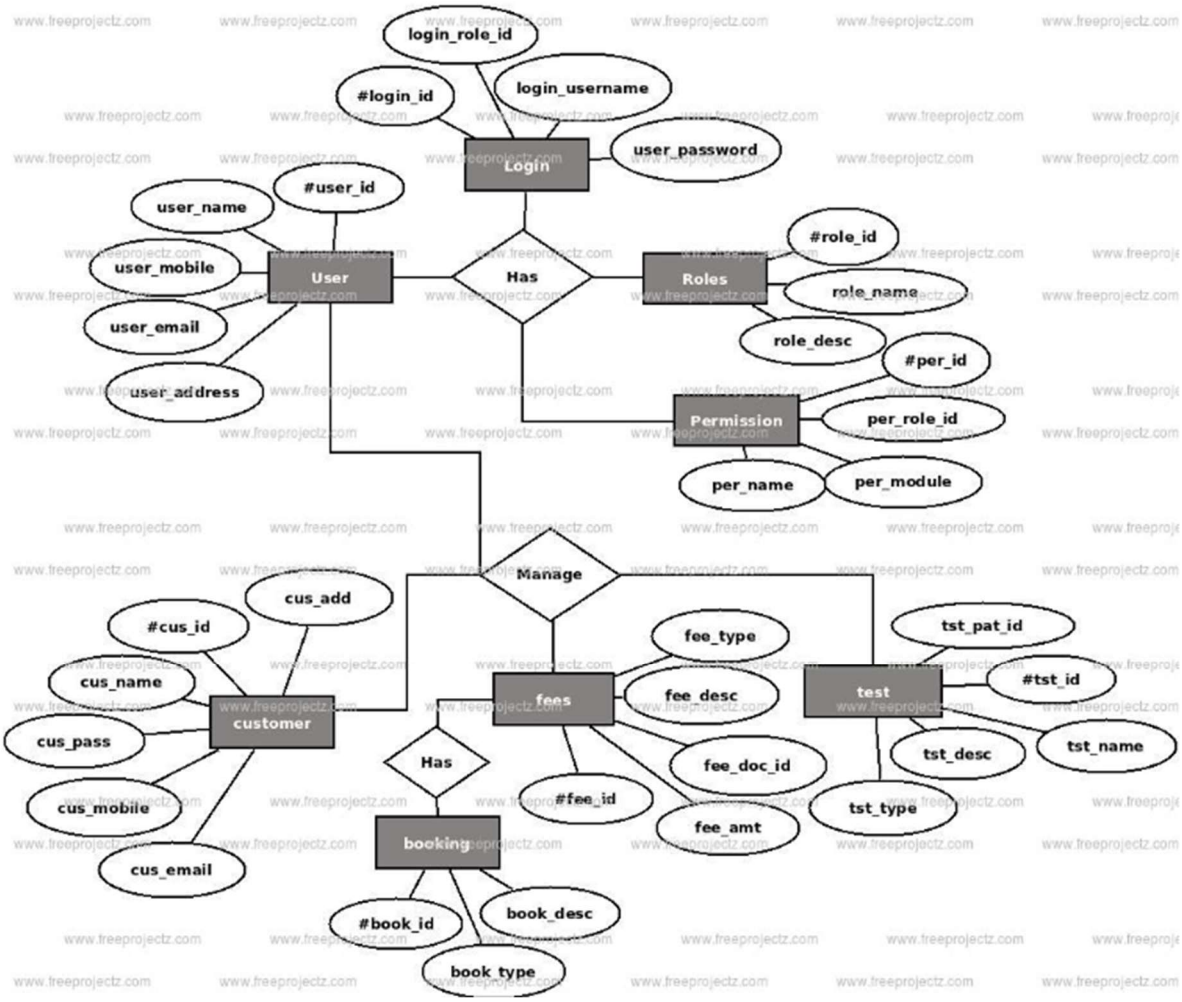
* Kalwar, A. (2022, February 17). Native App Development Vs Hybrid App Development. CodeHops. https://codehops.com/native-app-development-vs-hybrid-app-development/

3.) System Design

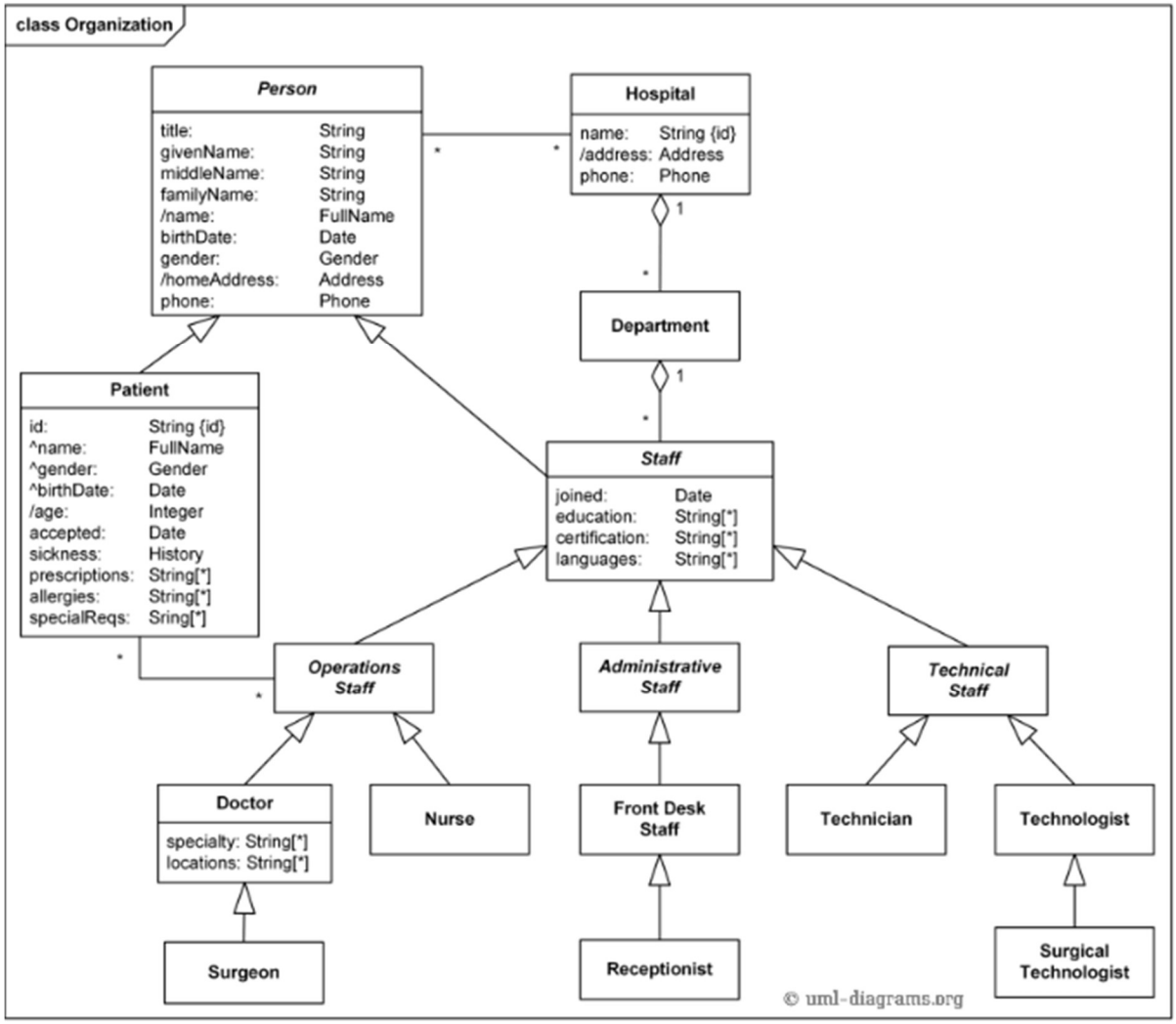
a. Use Case Diagram



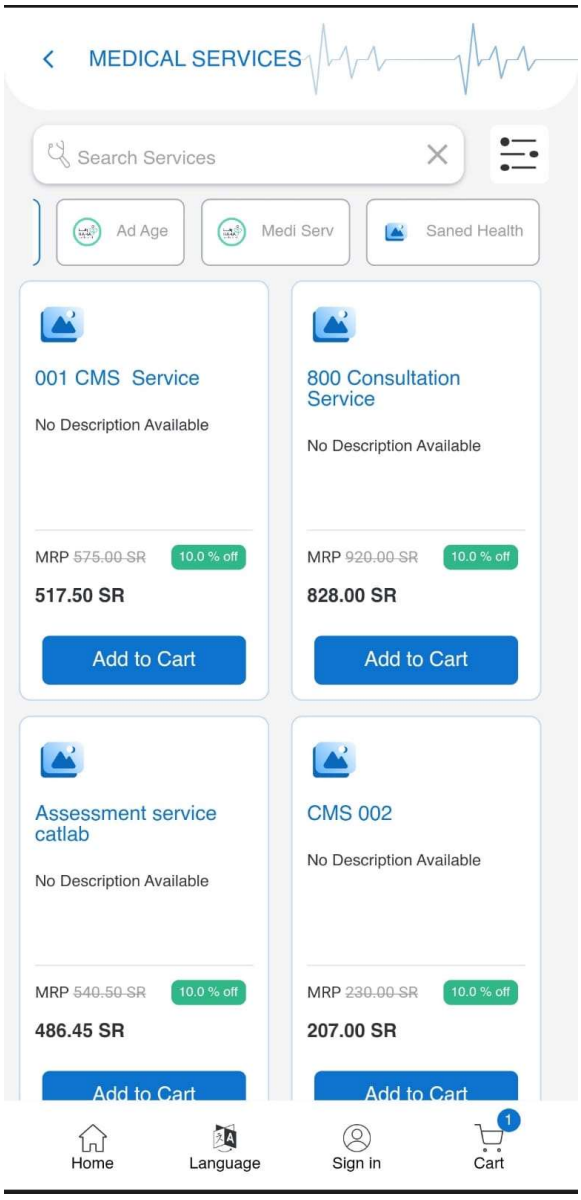
b. Entity Relationship (ER) Diagram

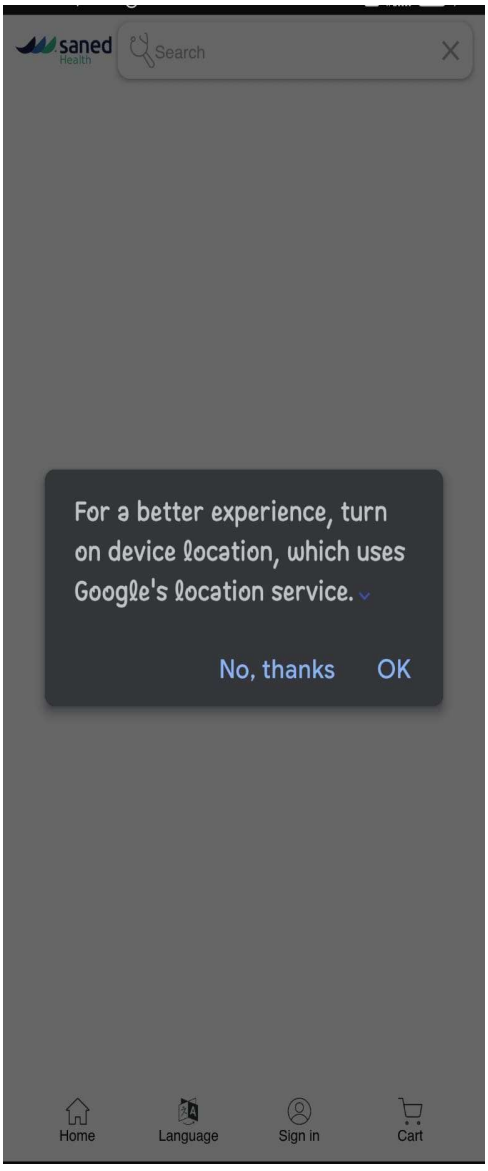


d. Class Diagram

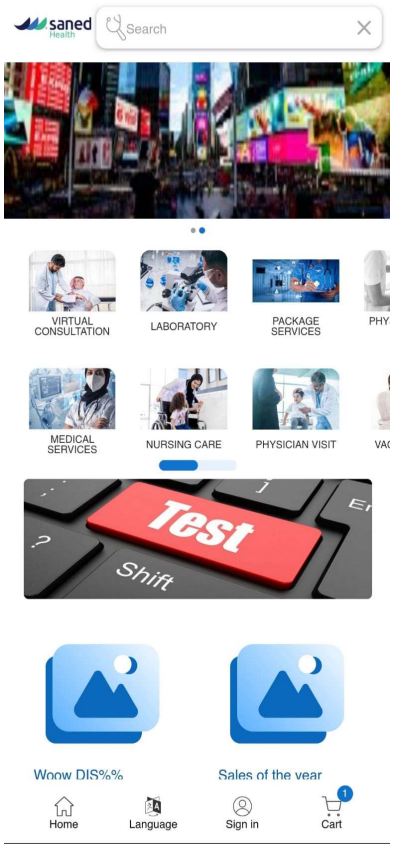


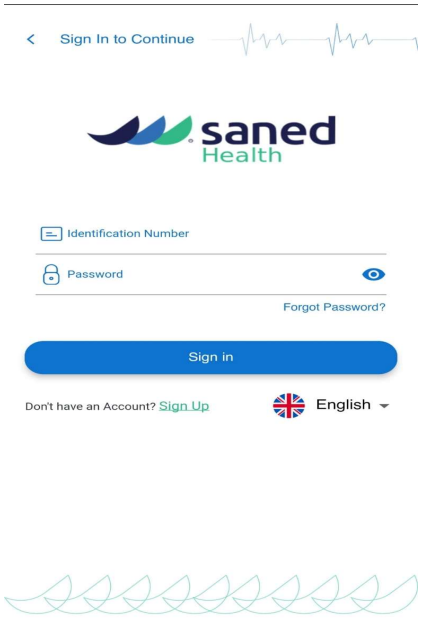
4.) Implementation of Project

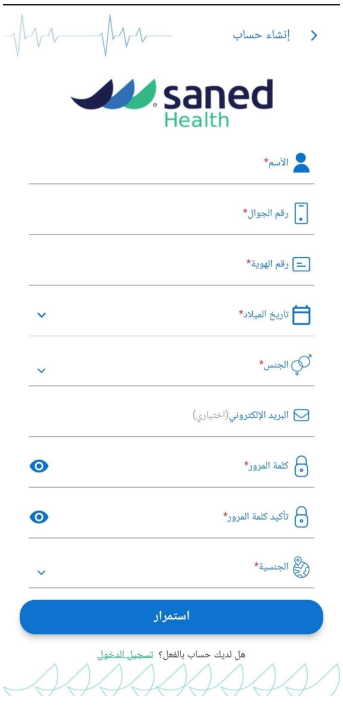


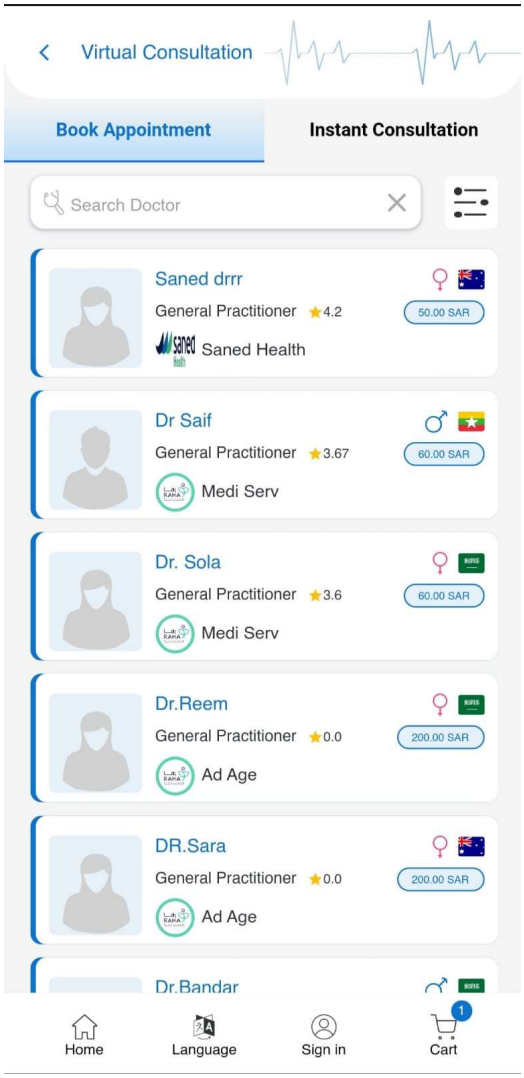


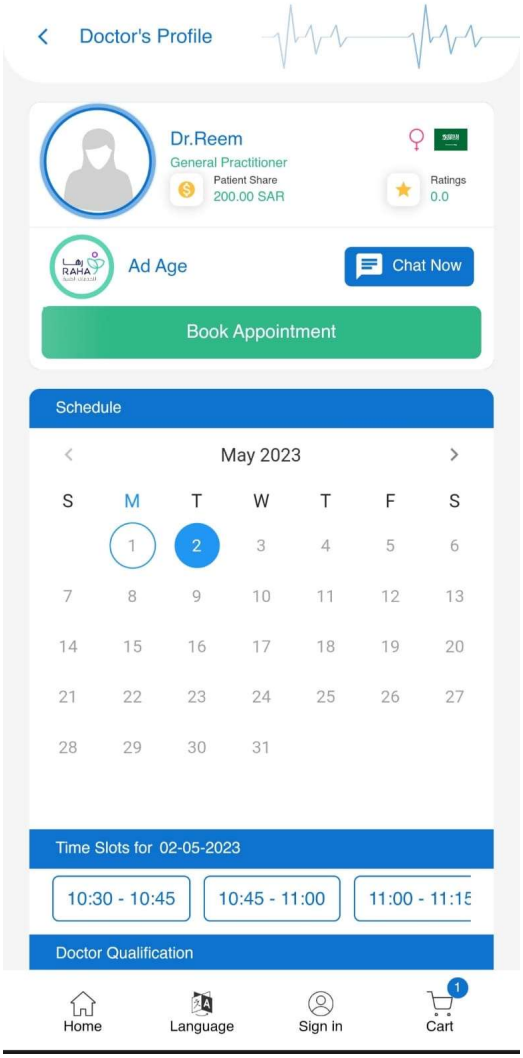

saned
Health
Search
Home
Language
Sign in
Cart


saned
Health
Search
VIRTUAL CONSULTATION
LABORATORY
PACKAGE SERVICES
MEDICAL SERVICES
NURSING CARE
PHYSICIAN VISIT
Test
Shift
Woow DIS%%
Sales of the year
Home
Language
Sign in
Cart


Sign In to Continue
saned
Health
Identification Number
Password
Forgot Password?
Sign in
Don't have an Account? Sign Up
English


إنشاء حساب
saned
Health
الاسم*
رقم الجوال*
رقم الهوية*
تاريخ الميلاد*
الجنس*
البريد الإلكتروني(اختياري)
كلمة المرور*
تأكيد كلمة المرور*
الجنسية*
استمرار
هل لديك حساب بالفعل؟ تسجيل الدخول


Virtual Consultation
Book Appointment
Instant Consultation
Search Doctor
Saned drrr
General Practitioner 4.2
50.00 SAR
Saned Health
Dr Saif
General Practitioner 3.67
60.00 SAR
Medi Serv
Dr. Sola
General Practitioner 3.6
60.00 SAR
Medi Serv
Dr.Reem
General Practitioner 0.0
200.00 SAR
Ad Age
DR.Sara
General Practitioner 0.0
200.00 SAR
Ad Age
Dr.Bandar
Home
Language
Sign in
Cart


Doctor's Profile
Dr.Reem
General Practitioner
Patient Share
200.00 SAR
Ratings
0.0
Ad Age
Chat Now
Book Appointment
Schedule
May 2023
S M T W T F S
1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
Time Slots for 02-05-2023
10:30 - 10:45
10:45 - 11:00
11:00 - 11:15
Doctor Qualification
Home
Language
Sign in
Cart

5.) Software Testing

is an important aspect of software development that ensures the quality of the software before its release. In this chapter, we will discuss the need for testing, testing strategy, and testing methods for the Flutter project developed during the internship at Codetrade.io.

5.1 Need for Testing

Testing is crucial for ensuring that the software meets the requirements and works as expected. The main reasons for software testing are:

1. To ensure that the software is free from defects and errors.
2. To validate that the software meets the requirements specified by the client.
3. To identify and fix the defects and errors before the software is released.
4. To ensure that the software works as expected in different environments and situations.
5. To enhance the reliability and maintainability of the software.

In the case of the Flutter project developed during the internship, testing is important to ensure that the application works as expected on different devices and operating systems, and that it meets the functional and non-functional requirements.

5.2 Testing Strategy

A testing strategy is a plan that outlines how testing will be performed for a software project. The testing strategy should include the following:

1. Test objectives: The goals and objectives of testing should be clearly defined.
2. Test scope: The areas of the software that will be tested should be defined.
3. Test approach: The approach to be used for testing should be selected, such as manual or automated testing.
4. Test environment: The hardware and software configurations that will be used for testing should be defined.
5. Test data: The data to be used for testing should be defined.
6. Test schedule: The timeline for testing should be defined.
7. Test resources: The resources required for testing should be defined.

In the case of the Flutter project developed during the internship, the testing strategy should include both manual and automated testing, and the testing should be performed on different devices and operating systems to ensure compatibility.

5.3 Testing Methods

There are several testing methods that can be used for software testing. The following are some of the testing methods that can be used for the Flutter project developed during the internship:

1. Unit testing: This method involves testing individual units or components of the software to ensure that they function correctly.

2. Integration testing: This method involves testing the integration of different units or components of the software to ensure that they work together as expected.

3. system testing: This method involves testing the functionality of the software to ensure that it meets the requirements specified by the client.

4. Security testing: This method involves testing the security of the software to ensure that it is secure and free from vulnerabilities.

In the case of the Flutter project developed during the internship, unit testing and integration testing should be performed to ensure that individual components and their integration work correctly. Functional testing should be performed to ensure that the software meets the functional requirements specified by the client. Performance testing should be performed to ensure that the software meets the performance requirements specified by the client. Finally, security testing should be performed to ensure that the software is secure and free from vulnerabilities.

In conclusion, software testing is an essential aspect of software development that ensures the quality of the software before its release. In the case of the Flutter project developed during the internship at Codetrade.io, testing is crucial to ensure that the software meets the functional and non-functional requirements and works as expected on different devices and operating systems. The testing strategy should include both manual and automated testing, and the testing methods should include unit testing, integration testing, functional testing, performance testing, and security testing.

1.) Unit Testing

Unit testing is a type of software testing that focuses on testing individual units or components of a system in isolation from the rest of the system. The purpose of unit testing is to verify that each unit or component of the system performs as intended and meets the specified requirements.

In unit testing, each unit or component is tested independently, using test cases that are designed to exercise all possible paths through the code. Unit tests are typically automated, using testing frameworks such as JUnit, NUnit, or PHPUnit, which provide a framework for organizing and running tests.

Unit tests are typically written by developers themselves as part of the software development process. The tests are written to cover all possible inputs and outputs of a given unit or component, including both expected and unexpected behaviors. The goal is to catch any errors

or defects in the code as early as possible, before they can propagate to other parts of the system and become more difficult to fix.

Unit testing is an important part of the software development process, as it helps ensure that each component of the system functions correctly and integrates seamlessly with the rest of the system. It also helps improve the quality of the code, as defects are caught early in the development process and can be fixed quickly and efficiently.

2.) Integration Testing

Integration testing is a type of software testing that focuses on testing the integration between individual software components or subsystems to ensure that they work together as intended. The purpose of integration testing is to identify defects or errors that may occur when different components or subsystems are combined.

Integration testing is typically performed after unit testing has been completed and before system testing begins. The testing process involves combining individual units or components into larger subsystems or modules, and then testing the interfaces and interactions between these subsystems or modules.

There are different approaches to integration testing, including top-down, bottom-up, and middle-out testing. In top-down integration testing, the higher-level subsystems or modules are tested first, followed by the lower-level subsystems or modules. In bottom-up integration testing, the lower-level subsystems or modules are tested first, followed by the higher-level subsystems or modules. Middle-out testing involves testing subsystems or modules that are in the middle of the hierarchy first, and then working outwards.

Integration testing can be challenging because it involves testing the interactions between different components or subsystems, which can be complex and difficult to predict. To overcome these challenges, integration testing should be well-planned and well-executed, with clear test cases and test data, and with appropriate tools and techniques to support the testing process.

Overall, integration testing is an important part of the software testing process, as it helps ensure that the different components or subsystems of a software system work together seamlessly and effectively. It helps identify and resolve issues early in the development process, reducing the risk of defects or errors in the final product.

3.) System Testing

System testing is a type of software testing that focuses on testing the entire system as a whole, rather than individual components or subsystems. The purpose of system testing is to ensure that the system meets all the specified requirements and performs as intended in its intended environment.

System testing is typically performed after integration testing has been completed and before user acceptance testing begins. The testing process involves testing the entire system, including all of its components and subsystems, in a variety of scenarios and situations to ensure that it functions correctly.

System testing may include a variety of different types of testing, such as functional testing, performance testing, security testing, usability testing, and others. The goal is to ensure that the system meets all of the specified requirements and performs as expected in a variety of different scenarios and situations.

System testing is typically performed by a dedicated testing team, rather than by the developers who created the system. The testing team is responsible for designing and executing test cases, documenting any defects or issues that are discovered, and working with the development team to resolve any issues that are identified.

Overall, system testing is an important part of the software testing process, as it helps ensure that the system meets all the specified requirements and performs as intended in its intended environment. It helps identify any defects or issues that may have been missed during earlier

stages of testing, and provides an opportunity to fine-tune and optimize the system before it is released to end-users.

4.) Security Testing

Security testing is a type of software testing that focuses on identifying and mitigating potential security risks and vulnerabilities in a software application or system. The purpose of security testing is to ensure that the system is secure and to prevent unauthorized access, theft, or modification of sensitive data.

Security testing includes a variety of techniques and methodologies that are designed to identify potential security risks and vulnerabilities, such as penetration testing, vulnerability scanning, threat modeling, and risk analysis. The goal is to identify any weaknesses or vulnerabilities in the system that could be exploited by attackers, and to develop strategies for mitigating or eliminating these risks.

Security testing typically involves a combination of manual and automated testing techniques. Manual testing may involve trying to exploit vulnerabilities in the system through various means, such as brute-force attacks, social engineering, or other techniques. Automated testing may involve using tools and software to scan for known vulnerabilities or to simulate attacks on the system.

There are different types of security testing, including network security testing, web application security testing, mobile application security testing, and others. The type of testing that is performed depends on the nature of the system and the specific security risks and vulnerabilities that need to be addressed.

Overall, security testing is an important part of the software development process, as it helps ensure that the system is secure and that sensitive data is protected from unauthorized access or modification. By identifying and addressing potential security risks and vulnerabilities early in the development process, security testing can help prevent security breaches and protect both the system and its users.

6.) Limitation & Future Enhancement

Limitation :-

While developing the Flutter app during the training at codetrade.io, there were a few limitations that were encountered. Additionally, there are a few areas that can be improved upon in the future. This chapter discusses the limitations and future enhancements of the project.

1. Dependence on internet connectivity: Certain features of the healthcare app, such as virtual consultations, require a stable internet connection, which may not be available in some areas.

2. Technical issues: As with any mobile application, there is a risk of technical issues such as app crashes or data loss.

3. User adoption: Patients may be hesitant to adopt a new technology for managing their health, and healthcare providers may not be familiar with using a mobile app to connect with patients.

4. Privacy concerns: Patients may be concerned about the privacy of their health data, especially when using a mobile app.

5. Cost: Developing a high-quality healthcare app can be expensive, which may limit the availability of such apps for patients and healthcare providers

6. Learning Curve: The learning curve for Flutter and Dart language can be steep, especially for beginners. It takes time to understand the basic concepts and the various packages that are available. This can slow down the development process.

7. Limited Native Features: Although Flutter provides a wide range of packages, there are still some native features that are not available. For example, the built-in camera app cannot be used to capture images within the app.

8. Compatibility Issues: Some packages might have compatibility issues with different versions of Flutter. This can lead to errors and crashes during the development process.

Future Enhancement

1. Improved medication adherence: The medication tracking feature of the healthcare app developed in Flutter can improve medication adherence by reminding patients to take their medication on time.

2. Increased accessibility: The virtual consultation feature of the healthcare app can increase accessibility to healthcare services, especially for patients in remote or rural areas.

3. Empowerment: By providing patients with access to health information and resources, the healthcare app can empower patients to take an active role in their health and make informed decisions about their care.

4. Efficiency: The healthcare app's use of Flutter's reactive programming model can lead to faster and more efficient app performance, improving user experience

5. Security: The app's use of cloud-based data storage can ensure that patient data is secure and easily accessible to authorized healthcare providers.

6. Integration with Native Features: The app can be improved by integrating with the native features of the device, such as the camera app, calendar, and contacts.

7. Improved UI: The user interface can be improved by using advanced Flutter widgets and animations to make the app more engaging and visually appealing.

8. Integration with Machine Learning: The app can be enhanced by integrating with machine learning libraries to provide better recommendations and insights to the users.

9. Localization: The app can be localized for different languages and regions, to make it more accessible to a wider audience.

10. Performance Optimization: The app's performance can be optimized by reducing the app size, improving the app's speed, and minimizing resource usage

Overall, while there are potential limitations to using a healthcare app developed in Flutter, the benefits of improved medication adherence, increased accessibility to healthcare services, patient empowerment, efficiency, and security make it a promising tool for improving healthcare delivery and patient outcomes.

7.) Conclusion

The healthcare app developed in Flutter has the potential to revolutionize healthcare delivery and improve patient outcomes. With features such as medication tracking, vital sign monitoring, virtual consultations, and access to health information and resources, the app provides patients with a comprehensive tool for managing their health and connecting with healthcare providers.

One of the key benefits of the app is improved medication adherence. The app's medication tracking feature allows patients to set reminders for taking their medication, reducing the risk of missed doses and ensuring that they are taking their medication on time. This can lead to better management of chronic conditions, improved patient outcomes, and reduced healthcare costs.

Another important benefit of the healthcare app developed in Flutter is increased accessibility of healthcare services. The app's virtual consultation feature allows patients to connect with healthcare providers remotely, reducing the need for in-person visits and making healthcare more accessible to patients in rural or remote areas. This can lead to improved patient satisfaction, better healthcare outcomes, and reduced healthcare costs.

The healthcare app developed in Flutter also provides patients with access to health information and resources. This can be particularly beneficial for patients who are managing chronic conditions or who have questions about their health. By providing patients with accurate and up-to-date information, the app can empower patients to take an active role in their health and make informed decisions about their care.

Of course, like any technology, the healthcare app developed in Flutter has its limitations. One potential limitation is the need for reliable internet connectivity. Certain features of the app, such as virtual consultations, require a stable internet connection. Another potential limitation is the potential for technical issues. As with any mobile application, there is always a risk of technical issues, such as app crashes or data loss.

Despite these potential limitations, the healthcare app developed in Flutter represents a significant step forward in healthcare technology. By providing patients with a comprehensive tool for managing their health and connecting with healthcare providers, the app has the potential to improve healthcare outcomes and reduce healthcare costs. As mobile technology continues to advance, we can expect to see more healthcare apps developed in frameworks like Flutter, offering new and innovative ways to deliver healthcare services.

In conclusion, during my internship at codetrade.io, I gained practical experience in Flutter app development. Starting with learning the basics of Flutter and the Dart language, I progressed to learning different widgets, state management, and packages like shared preferences, image_picker, path_provider, and many others.

During my time at codetrade.io, I also learned about project development approaches and worked with tools like Gantt charts to manage my workflow efficiently. I had the opportunity to work on real-life projects and implemented various features like geolocation, notifications, and Firebase services, which gave me a hands-on experience with the latest technologies.

In conclusion, my internship at codetrade.io was a valuable learning experience that helped me enhance my skills in Flutter app development. Moving forward, I intend to continue learning and staying up to date with the latest technologies to improve my proficiency in this field.

Future recommendations for the codetrade.io company would be to provide more opportunities for interns to work on complex projects and collaborate with experienced professionals to gain a deeper understanding of the practical aspects of software development.

8.) References

1. Flutter. (n.d.). Retrieved March 26, 2023, from https://flutter.dev/

2. Dart Programming Language. (n.d.). Retrieved March 26, 2023, from https://dart.dev/

3. Image Picker. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/image_picker

4. Path Provider. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/path_provider

5. Video Player. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/video_player

6. Chewie. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/chewie

7. Permission Handler. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/permission_handler

8. File Picker. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/file_picker

9. URL Launcher. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/url_launcher

10. Fluttertoast. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/fluttertoast

11. Cupertino Icons. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/cupertino_icons

12. Geolocator. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/geolocator

13. Geocoding. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/geocoding

14. Flutter Easyloading. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/flutter_easyloading

15. Firebase Analytics. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/firebase_analytics

16. Firebase Core. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/firebase_core

17. Cached Network Image. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/cached_network_image

18. Firebase Crashlytics. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/firebase_crashlytics

19. Firebase Messaging. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/firebase_messaging

20. Device Info Plus. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/device_info_plus

21. Package Info Plus. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/package_info_plus

22. Flutter Local Notifications. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/flutter_local_notifications

23. Cloud Firestore. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/cloud_firestore

24. Firebase Storage. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/firebase_storage

25. Dio. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/dio

26. Intl. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/intl

27. Agora RTC Engine. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/agora_rtc_engine

28. Agora UIKit. (n.d.). Retrieved March 26, 2023, from https://pub.dev/packages/agora_rtc_engine

29. Flutter Documentation: Official documentation for the Flutter framework provided by Google. Available at: https://docs.flutter.dev/

30. Dart Documentation: Official documentation for the Dart programming language used in Flutter development. Available at: https://dart.dev/guides

31. Flutter Packages: Official repository of Flutter packages, including packages used in the project such as image_picker, path_provider, video_player, chewie, permission_handler, file_picker, url_launcher, fluttertoast, cupertino_icons, geolocator, geocoding, flutter_easyloading, firebase_analytics, firebase_core, cached_network_image, firebase_crashlytics, firebase_messaging, device_info_plus, package_info_plus, flutter_local_notifications, cloud_firestore, firebase_storage, dio, intl, agora_rtc_engine, agora_uikit, flutter_animate, and animated_text_kit. Available at: https://pub.dev/flutter/packages

32. Stack Overflow: An online community where developers can ask and answer programming questions related to Flutter and Dart. Available at: https://stackoverflow.com/

33. GitHub: An online code hosting platform where developers share and collaborate on open-source projects, including Flutter libraries and sample projects. Available at: https://github.com/

34. CodeTrade.io: The company where the internship was done, providing training on Flutter development and facilitating the practical implementation of the project. Available at: https://www.codetrade.io/