VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA", BELAGAVI - 590 018

A MINI PROJECT REPORT

on

"SMARTMEDICS"

Submitted by

Shreeganesh	4SF20IS092
Vraj K Rabara	4SF20IS116

In partial fulfillment of the requirements for the VI semester

MOBILE APPLICATION DEVELOPMENT

of

BACHELOR OF ENGINEERING

in

INFORMATION SCIENCE & ENGINEERING

Under the Guidance of

Mr. Ganaraj K

Assistant Professor, Department of ISE

at

SAHYADRI

College of Engineering & Management

An Autonomous Institution

MANGALURU

2022 - 23

SAHYADRI

College of Engineering & Management

An Autonomous Institution

MANGALURU

Department of Information Science & Engineering

CERTIFICATE

This is to certify that the **Mini Project** entitled **"Smartmedics"** has been carried out by **Shreeganesh (4SF20IS092)** and **Vraj K Rabara (4SF20IS116)**, the bonafide students of Sahyadri College of Engineering & Management in partial fulfillment of the requirements for the VI semester **Mobile Application Development (18ISMP68)** of **Bachelor of Engineering in Information Science & Engineering** of Visvesvaraya Technological University, Belagavi during the year 2022 - 23. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of mini project work.

Mr. Ganaraj K	**Dr. Mustafa Basthikodi**
Assistant Professor	Professor & Head
Dept. of ISE, SCEM	Dept. of ISE & CSE(DS), SCEM

External Practical Examination:

Examiner's Name	Signature with Date
1.	.....
2.	.....

SAHYADRI

College of Engineering & Management

An Autonomous Institution

MANGALURU

Department of Information Science & Engineering

DECLARATION

We hereby declare that the entire work embodied in this Mini Project Report titled **"Smartmedics"** has been carried out by us at Sahyadri College of Engineering and Management, Mangaluru under the supervision of **Mr. Ganaraj K** as the part of the VI semester **Mobile Application Development (18ISMP68)** of **Bachelor of Engineering** in **Information Science & Engineering**. This report has not been submitted to this or any other University.

Shreeganesh (4SF20IS092)

Vraj K Rabara (4SF20IS116)

SCEM, Mangaluru

Abstract

In the rapidly evolving landscape of healthcare technology, the integration of smart solutions has revolutionized the way individuals manage their health and access medical services. The convergence of smart medical assistance systems and online pharmacy ordering platforms has given rise to a new paradigm in healthcare delivery. This introduction provides an overview of Smartmedics, an innovative online pharmacy ordering application that combines the power of smart medical assistance with seamless medication procurement, offering users a comprehensive and convenient healthcare experience.Smartmedics is an advanced mobile application designed to simplify the process of ordering medications online. By harnessing the potential of smart technologies and leveraging the convenience of digital platforms, Smartmedics aims to empower individuals to take control of their health by providing easy access to a wide range of medications and pharmaceutical services. The application focuses on addressing the challenges often associated with traditional pharmacy services, such as long waiting times, limited availability, and the need for physical visits.

Acknowledgement

It is with great satisfaction and euphoria that we are submitting the Mini Project Report on **"Smartmedics"**.We have completed it as a part of the VI semester **Mobile Application Development (18ISMP68)** of **Bachelor of Engineering** in **Information Science & Engineering** of Visvesvaraya Technological University, Belagavi.

We are profoundly indebted to our guide, **Mr. Ganaraj K**, Assistant Professor, Department of Information Science & Engineering for innumerable acts of timely advice, encouragement and We sincerely express our gratitude.

We express our sincere gratitude to **Dr. Mustafa Basthikodi**, Professor &Head, Department of Information Science & Engineering for his invaluable support and guidance.

We sincerely thank **Dr. Rajesha S**, Principal, Sahyadri College of Engineering & Management who have always been a great source of inspiration.

Finally, yet importantly, We express our heartfelt thanks to our family & friends for their wishes and encouragement throughout the work.

Shreeganesh
4SF20IS092
VI Sem, B.E., ISE
SCEM, Mangaluru

Vraj K Rabara
4SF20IS116
VI Sem, B.E., ISE
SCEM, Mangaluru

Table of Contents

List of Figures

Chapter 1

Introduction

The Smartmedics application is designed to revolutionize the way individuals access and purchase medications, providing a convenient and user-friendly platform for online pharmaceutical services. This abstract presents an overview of the app's key features, highlighting its potential to streamline the medication procurement process and improve overall healthcare experiences.The primary objective of the application is to offer a seamless and accessible means for individuals to order and receive medications from the comfort of their homes. The application enables users to browse a comprehensive catalog of medications, facilitating quick and easy searches based on specific drug names, conditions, or prescriptions. Through an intuitive and user-friendly interface, users can review detailed information about each medication, including dosage instructions, side effects, and potential drug interactions, empowering them to make informed decisions regarding their healthcare.In conclusion, the application offers a transformative solution to streamline medication procurement processes, making it easier for individuals to access the medications they need. By combining convenience, comprehensive medication information,this application has the potential to revolutionize the online pharmacy industry and improve healthcare experiences for users

1.1 Overview

SmartMedics is an advanced online pharmacy ordering application that combines smart medical assistance with convenient medication procurement. It offers an extensive medication catalog, prescription management,personalized recommendations,medication reminders, real-time health monitoring, secure payment, and delivery. SmartMedics provides convenience, personalized healthcare, improved medication adherence, data security, and an enhanced healthcare experience by seamlessly integrating smart technologies with online pharmacy services.

1.2 Purpose

The purpose of SmartMedics, an online pharmacy ordering application, is to provide individuals with a convenient and user-friendly platform to access and manage their medications. It aims to simplify the medication procurement process, enhance medication adherence, and empower users to make informed decisions about their health. By integrating smart medical assistance features, SmartMedics strives to improve the overall healthcare experience and promote better health outcomes.

1.3 Scope

The scope of SmartMedics, the online pharmacy ordering application, includes providing a comprehensive platform for individuals to browse and order medications . The application also incorporates real-time health monitoring and secure payment and delivery services. It aims to cater to a wide range of users, enhance medication accessibility, and streamline the medication procurement process, ultimately improving the overall healthcare experience for individuals.

Chapter 2

Requirements Specification

2.1 Hardware Specification

- Processor : Intel(R) Core(TM) i5-1005G1 CPU @ 2.40GHz
- RAM : 8GB
- Hard Disk : 500GB
- Input Device : Standard keyboard and Mouse
- Output Device : Monitor

2.2 Software Specification

- Programming Language : Java 17.0.1
- Markup Language : XML 1.0
- IDE : Android Studio 2022.1
- Database: Google Firestore

Chapter 3

System Design

3.1 Architecture Diagram

The architecture diagram is a visual representation of the components, structure, and relationships of a system or application . The architecture diagram of the application is as shown in the below figure 3.1:

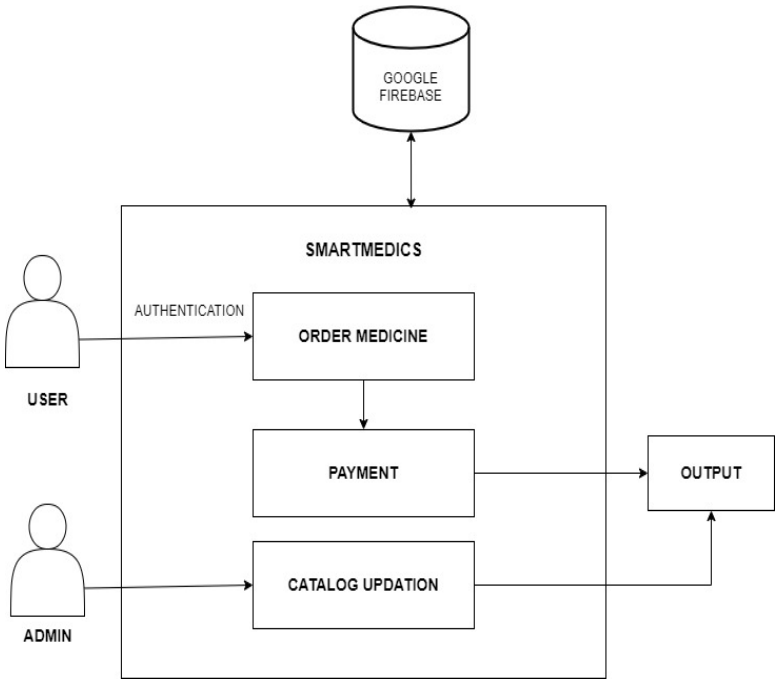

Figure 3.1: Architecture Diagram of Smartmedics

The architecture diagram consists of user and admin module .Users access the application through a user interface, where they can browse medications and place orders. The server handles user authentication, stores user data in a database, and integrates with a payment gateway for secure transactions. The admin module provides tools for managing inventory, processing orders, and generating reports.

3.2 Application Modules

3.2.1 User Module

On the user side, there would be a user interface (UI) component that allows users to browse the available medications, search for specific products, and place orders .The server would also handle user authentication and authorization, ensuring that only registered and authenticated users can access the application's functionalities. User data, such as personal information and order history, would be stored in a database for retrieval and processing.

3.2.2 Admin Module

The admin module would provide additional functionalities for administrators to manage the application. It would include features such as managing product inventory, updating prices, reviewing and processing user orders, and generating reports. The admin module would have its own UI, separate from the user interface, and would communicate with the server through dedicated APIs.

3.3 End Users

The end user of online pharmacy ordering application focuses on providing a user-friendly interface for individuals to browse medications, add them to their shopping carts, and place orders. It includes features such as user registration, medication search and browsing, and a seamless checkout process.

3.4 Limitations

- **Limited Accessibility:** Not everyone has access to the internet or may face barriers in using online platforms so it limit its reach and accessibility , particularly in remote areas or with limited digital literacy.
- **Device Compatibility:** Due to the vast number of device models and variations in hardware specifications, there can be compatibility issues.
- **Tracking:** The application does not include real time tracking of the item being delivered to the user.

Chapter 4

Implementation

4.1 Overview

Android is a Linux-based mobile operating system that primarily runs on smartphones and tablets. The Android platform includes an operating system based upon the Linux kernel, a GUI, a web browser and end-user applications that can be downloaded. Although the initial demonstrations of Android featured a generic QWERTY smartphone and large VGA screen, the operating system was written to run on relatively inexpensive handsets with conventional numeric keypads.Android was released under the Apache v2 open source license; this allows for many variations of the OS to be developed for other devices, such as gaming consoles and digital cameras. Android is based on open source software, but most Android devices come preinstalled with a suite of proprietary software, such as Google Maps, YouTube, Google Chrome and Gmail.

Android began its life as a Palo Alto-based startup company called Android Inc., in 2003. Originally, the company set out to develop an operating system for digital cameras, but it abandoned those efforts in lieu of reaching a broader market. Google acquired Android Inc. and its key employees in 2005. Google marketed the early mobile platform to handset manufacturers and mobile carriers with its major benefits as flexibility and upgradability.

4.2 Languages Used

4.2.1 Java

Java is one of the powerful general-purpose programming language, created in 1995 by Sun Microsystems (now owned by Oracle). Java is Object-Oriented. However, it is not considered as pure object-oriented as it provides support for primitive data types (like int, char, etc) and does not provide low-level programming functionalities like pointers. Also, Java code is always written in the form of classes and objects. Android heavily relies on the Java programming language all the SDKs required to build for android applications use the standard libraries of Java. If one is coming from a traditional programming background like C, C++, Java is easy to learn. Java is designed to be platform-independent, making it suitable for developing applications that can run on different operating systems, including Android. Android Studio, the official IDE for Android application development, provides built-in support for Java, making it the primary language choice for Android development.

4.2.2 XML

XML stands for Extensible Markup Language. XML is a markup language much like HTML used to describe data. It is derived from Standard Generalized Markup Language(SGML). Basically, the XML tags are not predefined in XML. We need to implement and define the tags in XML. XML tags define the data and used to store and organize data. It's easily scalable and simple to develop. In Android, the XML is used to implement UI-related data, and it's a lightweight markup language that doesn't make layout heavy. XML only contains tags, while implementing they need to be just invoked. Basically in Android XML is used to implement the UI-related data. So understanding the core part of the UI interface with respect to XML is important. The User Interface for an Android application is built as the hierarchy of main layouts, widgets. The layouts are ViewGroup objects or containers that control how the child view should be positioned on the screen. Widgets here are view objects, such as Buttons and text boxes.

4.3 Android Studio

Android Studio is the official Integrated Development Environment (IDE) for android application development. Android Studio provides more features that enhance our productivity while building Android applications. Android Studio was announced on 16th May 2013 at the Google I/O conference as an official IDE for Android application development. It started its early access preview from version 0.1 in May 2013. The first stable built version was released in December 2014, starts from version 1.0. Since 7th May 2019, Kotlin is Google's preferred language for Android application development. Besides this, other programming languages are supported by Android Studio. It has a flexible Gradle-based build system. It has a fast and feature-rich emulator for application testing. Android Studio has a consolidated environment where we can develop for all Android devices. Apply changes to the resource code of our running application without restarting the application. Android Studio provides extensive testing tools and frameworks. It supports C++ and NDK. It provides build-in supports for Google Cloud Platform. It makes it easy to integrate Google Cloud Messaging and application Engine.

4.4 Google Firebase

Google Firebase is a mobile application development platform from Google with powerful features for developing, handling, and enhancing applications. Firebase is a backend platform for building web and mobile applications. Firebase is fundamentally a collection of tools developers can rely on, creating applications and expanding them based on demand. Developers relying on this platform get access to services that they would have to develop themselves, and it enables them to lay focus on delivering robust application experiences.Some of the Google Firebase platform's standout features include databases, authentication, push messages, analytics, file storage, and much more. Since the services are cloud-hosted, developers can smoothly perform on-demand scaling without any hassle. Firebase is currently among the top application development platforms relied upon by developers across the globe. The first Firebase product launched was the Realtime Database. It is an API for application data synchronization across Android, web, and iOS devices.

4.5 Pseudocodes

Pseudocode for User Registering:

The RegisterActivity handles user registration using Firebase authentication. It retrieves email and password input from EditText fields and validates them. The registerUser() method creates a new user using FirebaseAuth. Upon successful registration, a toast message is displayed, and the user is redirected to the LoginActivity. If the registration fails, an error message is displayed.

```
register.setOnClickListener((v) -> {
    String txt_email, txt_password, txt_phone, txt_name;
    txt_email = email.getText().toString();
    txt_password = password.getText().toString();
    txt_phone = phone.getText().toString();
    txt_name = username.getText().toString();
    if (txt_email.isEmpty() || txt_password.isEmpty() || txt_name.isEmpty() || txt_phone.isEmpty()) {
        Toast.makeText( context: RegisterActivity.this, text: "Empty credentials", Toast.LENGTH_SHORT).show();
    } else if (txt_password.length() < 8) {
        Toast.makeText( context: RegisterActivity.this, text: "Password should be at least 8 characters long!", Toast.LENGTH_SHORT).show();
    } else if (!Pattern.compile( regex: "^(?=.*[a-z])(?=.*[A-Z])(?=.*\\d)(?=.*[@$!%*?&])[A-Za-z\\d@$!%*?&]+$").matcher(txt_password).matches()) {
        Toast.makeText( context: RegisterActivity.this, text: "Password should contain at least one uppercase letter, one lowercase letter, one digit,
    } else if (!Patterns.EMAIL_ADDRESS.matcher(txt_email).matches()) {
        Toast.makeText( context: RegisterActivity.this, text: "Invalid email address", Toast.LENGTH_SHORT).show();
    } else if (!txt_phone.matches( regex: "\\d{10}")) {
        Toast.makeText( context: RegisterActivity.this, text: "Invalid phone number", Toast.LENGTH_SHORT).show();
    } else if (txt_name.length() < 3) {
        Toast.makeText( context: RegisterActivity.this, text: "Username should be at least 3 characters long!", Toast.LENGTH_SHORT).show();
    } else {
        registerUser(txt_email, txt_password, txt_phone, txt_name);
    }
});
```

Figure 4.1: Pseudocode for User Registering

Pseudocode for Adding Address:

The AddAddressActivity allows users to input their address details such as name, address, city, postal code, and phone number. When the user clicks the add address button, the input is validated, and the address information is stored in Firestore. Upon successful addition, a toast message is displayed, and the user is directed to the DetailedActivity.

```
if (!userNumber.isEmpty() && !userAddress.isEmpty() && !userCity.isEmpty() && !username.isEmpty() && !userCode.isEmpty()) {

    Map<String,String> map = new HashMap<>();
    map.put( k: "userAddress",final_address);

    firestore.collection( collectionPath: "CurrentUser").document(auth.getCurrentUser().getUid())
            .collection( collectionPath: "Address").add(map).addOnCompleteListener(new OnCompleteListener<DocumentReference>() {
                @Override
                public void onComplete(@NonNull Task<DocumentReference> task) {
                    if(task.isSuccessful()) {
                        Toast.makeText( context: AddAddressActivity.this,  text: "Address Added", Toast.LENGTH_SHORT).show();
                        startActivity(new Intent( packageContext: AddAddressActivity.this,AddressActivity.class));
                        finish();
                    }
                }
            });

}else {
    Toast.makeText( context: AddAddressActivity.this,  text: "Kindly Fill All Fields", Toast.LENGTH_SHORT).show();
}
```

Figure 4.2: Pseudocode for Adding Address

Pseudocode for Calculating TotalPrice in Cart:

The onClick() method invokes calculateTotalPriceAndProceedToPayment(). Inside calculateTotalPriceAndProceedToPayment(), the total price is calculated by iterating through the cart items and creating an OrderModel object for each item. An intent is then created to proceed to the payment process, passing the total price and ordered items as extras.

```
checkoutButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { calculateTotalPriceAndProceedToPayment(); }

    1 usage
    private void calculateTotalPriceAndProceedToPayment() {


        orderedProducts = new ArrayList<>();
        double totalPrice = 0.0;

        // Iterate through the cart items and calculate the total price
        for (CartItemModel cartItem : cartItemList) {
            double price = Double.parseDouble(cartItem.getPrice());
            totalPrice += price;

            // Create an OrderModel object for each cart item and add it to the list
            OrderModel orderItem = new OrderModel(cartItem.getName(), cartItem.getImg_url(), cartItem.getPrice(),
            orderedProducts.add(orderItem);
        }
        // Proceed to the payment activity and pass the total price
        Intent intent = new Intent( packageContext: CartActivity.this, AddressActivity.class);
        intent.putExtra( name: "sourceActivity",  value: "CartActivity");
        intent.putExtra( name: "totalPrice", totalPrice);
        intent.putExtra( name: "orderedProducts", orderedProducts);
        startActivity(intent);

    }
});
```

Figure 4.3: Pseudocode for Calculating TotalPrice in Cart:

Pseudocode for Adding Details to Cart:

The code snippet checks if all required fields in the address form are filled. If they are not empty, it adds the address information to a HashMap and stores it in Firestore under the "Address" collection. Upon successful completion, a toast message is displayed, and the user is redirected to the AddressActivity. If any field is empty, a toast message prompts

the user to fill all fields.

```
final HashMap<String, Object> cartMap = new HashMap<>();

cartMap.put("productImage", imageUrl); // Store the image URL
cartMap.put("productName", detailedName.getText().toString());
cartMap.put("productPrice", String.valueOf(totalPrice));
cartMap.put("productQuantity", quantity.getText().toString());


firestore.collection( collectionPath: "AddToCart") CollectionReference
        .document(auth.getCurrentUser().getUid()) DocumentReference
        .collection( collectionPath: "User") CollectionReference
        .add(cartMap) Task<DocumentReference>
        .addOnCompleteListener(new OnCompleteListener<DocumentReference>() {
            @Override
            public void onComplete(@NonNull Task<DocumentReference> task) {
                if (task.isSuccessful()) {
                    Toast.makeText( context: DetailedActivity.this,  text: "Added to Cart", Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText( context: DetailedActivity.this,  text: "Failed to add to Cart", Toast.LENGTH_SHORT).show();
                }
                finish();
            }
        });
```

Figure 4.4: Pseudocode for Adding Details to Cart

Pseudocode for Retrive Popular and New Product Details:

The code snippet checks if product models (newProductsModel and popularProductModel) are not null. If not null, it retrieves the necessary information such as price, image URL, name, description, benefits, and direction of use. It then updates the UI elements with the retrieved data.

```
    // New Products
    if (newProductsModel != null) {
        // Ensure the price is of numeric data type, e.g., double or int
        double productPrice = Double.parseDouble(newProductsModel.getPrice());
        totalPrice = (int) (productPrice * totalQuantity);

        Glide.with(getApplicationContext()).load(newProductsModel.getImg_url()).into(detailedImg);
        detailedName.setText(newProductsModel.getName());
        description.setText(newProductsModel.getDescription());
        benefits.setText(newProductsModel.getBenefits());
        direction.setText(newProductsModel.getDirection_of_use());
        price.setText(String.valueOf(totalPrice)); // Set the price TextView with the new totalPrice
    }

    // popular Products
    if (popularProductModel != null) {
        // Ensure the price is of numeric data type, e.g., double or int
        double productPrice = Double.parseDouble(popularProductModel.getPrice());
        totalPrice = (int) (productPrice * totalQuantity);

        Glide.with(getApplicationContext()).load(popularProductModel.getImg_url()).into(detailedImg);
        detailedName.setText(popularProductModel.getName());
        description.setText(popularProductModel.getDescription());
        benefits.setText(popularProductModel.getBenefits());
        direction.setText(popularProductModel.getDirection_of_use());
        price.setText(String.valueOf(totalPrice)); // Set the price TextView with the new totalPrice
    }
```

Figure 4.5: Pseudocode for Retrive Popular and New Product Details

Pseudocode for Store Ordered Products Details:

The code snippet stores order data in Firebase Firestore, including product details and the order's date and time. It uses a HashMap to organize the data and saves it to the

"Orders" collection with the document ID derived from the current user. A completion listener handles the success or failure of the storage operation and displays a corresponding toast message.

```
// Store the product name, price, image, and ordered datetime in Firebase
String currentDate = new SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss", Locale.getDefault()).format(Calendar.getInstance().getTime());
HashMap<String, Object> orderData = new HashMap<>();
orderData.put("productName", productName);
orderData.put("price", amount);
orderData.put("image", productImage);
orderData.put("orderedDateTime", currentDate);

FirebaseFirestore.getInstance().collection( collectionPath: "Orders") CollectionReference
        .document(auth.getCurrentUser().getUid()) DocumentReference
        .collection( collectionPath: "Products") CollectionReference
        .document() // Use auto-generated document ID
        .set(orderData) Task<Void>
        .addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                // Handle the completion of storing the order data
                if (task.isSuccessful()) {
                    // Order data stored successfully
                    Toast.makeText( context: PaymentActivity.this, text: "Order placed successfully!", Toast.LENGTH_SHORT).show();
                } else {
                    // Failed to store order data
                    Toast.makeText( context: PaymentActivity.this, text: "Failed to place the order!", Toast.LENGTH_SHORT).show();
                }
            }
        });
```

Figure 4.6: Pseudocode for Store Ordered Products Details

Pseudocode for Retrive Ordered Products Details from firestore:

The code retrieves order data from Firebase Firestore, specifically from the "Orders" collection and the "Products" subcollection. It sorts the documents by the "orderedDateTime" field in descending order. It then extracts the product details, handles the price field based on its type, and creates an OrderModel object for each order, adding it to the orderList.

```
// Retrieve the ordered product data from Firebase
firestore.collection( collectionPath: "Orders") CollectionReference
        .document(auth.getCurrentUser().getUid()) DocumentReference
        .collection( collectionPath: "Products") CollectionReference
        .orderBy( field: "orderedDateTime", Query.Direction.DESCENDING) Query
        .get() Task<QuerySnapshot>
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                orderList.clear();
                for (QueryDocumentSnapshot document : task.getResult()) {
                    // Retrieve the product details
                    String productName = document.getString( field: "productName");
                    String productImage = document.getString( field: "image");
                    String productPrice = document.getString( field: "price");

                    // Handle the price field correctly based on its actual type
                    double price = Double.parseDouble(productPrice);
                    Object priceObj = document.get("price");
                    if (priceObj instanceof Double) {
                        price = (Double) priceObj;
                    } else if (priceObj instanceof Long) {
                        price = ((Long) priceObj).doubleValue();
                    }

                    String orderedDateTime = document.getString( field: "orderedDateTime");
                    // Create a new OrderModel object
                    OrderModel orderModel = new OrderModel(productName, productImage, String.valueOf(price), orderedDateTime);
                    // Add the OrderModel to the list
                    orderList.add(orderModel);
                }
```

Figure 4.7: Pseudocode for Retrive Ordered Products Details from Firestore

Chapter 5

Results and Disscussion

Login Activity:

The login page is an simple and intuitive design that allows users to log in using their email and password.The login page of the SmartMedics application ensures a secure and convenient login experience for users, allowing them to access their accounts using their registered email and password.

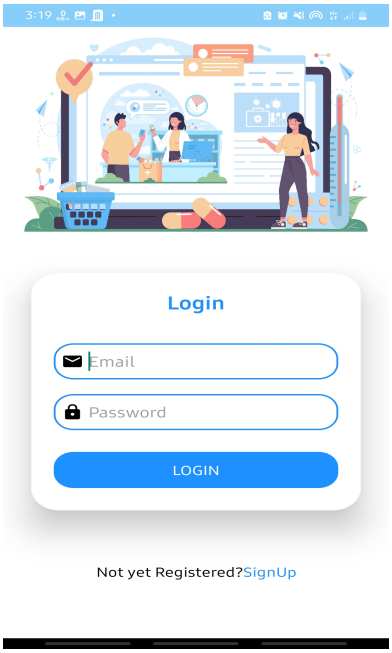

Figure 5.1: Login Activity

Sign-Up Activity:

Users without an existing account are granted the ability to create a new account. The sign-up facilitates a smooth and secure registration process, allowing new users to create an account by providing their desired username, email, phone number, and password.

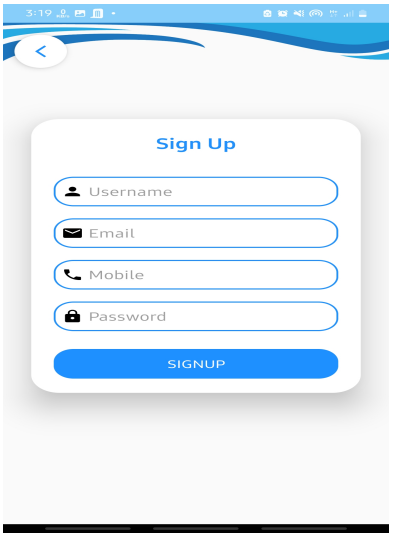

Figure 5.2: Sign-Up Activity

Home Page Activity:

The home page of the smartmedics application is designed to provide users with easy access to various features and information. It offers a visually appealing layout with easily accessible categories, a search bar for quick item lookup, and highlights new products to keep users informed about the latest offerings.

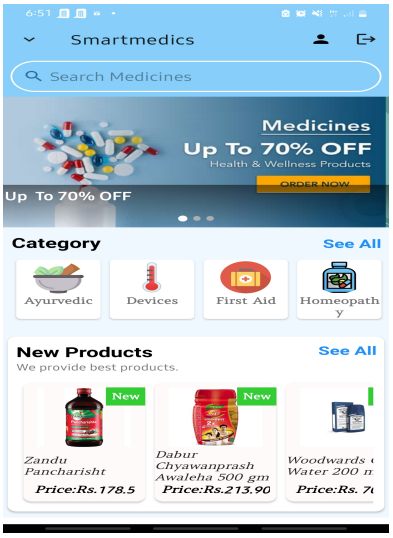

Figure 5.3: Home Page Activity

Profile Activity:

Welcome to your personal profile page on our smartmedics application! This page is all about you, where your personal information like name ,email id ,address,orders and cart will be displayed and you can manage your account settings.

Figure 5.4: Profile Activity

Product Detail Activity:

The product details page provides all the essential information you need to make informed decisions about your medicines. It includes medicines name, description, benefits , direction of use, pricing, and quantity. Stay informed and confident in your healthcare choices with our comprehensive product details page.

Figure 5.5: Product Detail Activity

MyCart Activity:

The my cart page provides a convenient and user-friendly experience. It allows you to manage your selected medications, check quantity, calculate the total cost of your order.

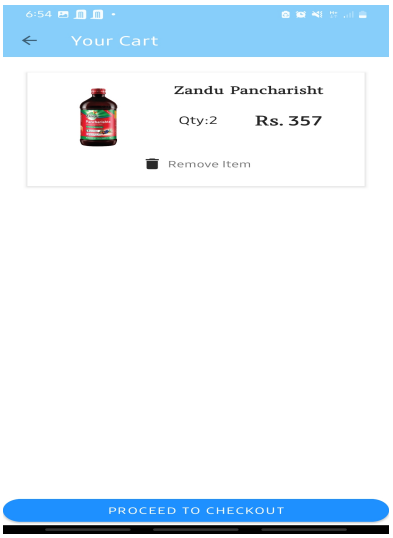

Figure 5.6: MyCart Activity

Add Address Activity:

You can add new address in the add address page. For adding a new address u have to fill the details like name, address, city, pincode and phone number.

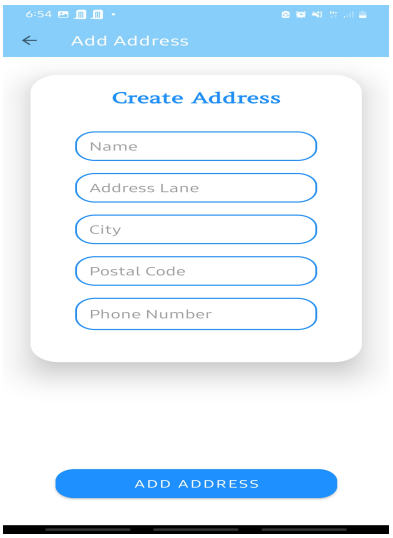

Figure 5.7: Add Address Activity

Order History Activity:

The order history page provides a summary of your orders.You can view order's details, including name of medicine, date and time , and total costs.

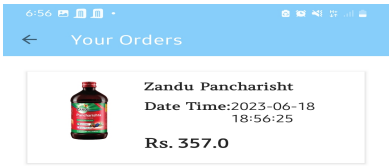

Figure 5.8: Order History Activity

Payment Activity:

Our payment page on the smartmedics application ensures a secure and seamless checkout process. You can review your order summary, select from various payment options, and complete your purchase with confidence. The sub total is calculated which consist of product price plus gst plus shipping charges. We prioritize the security of your financial information, providing a convenient and hassle-free payment experience.

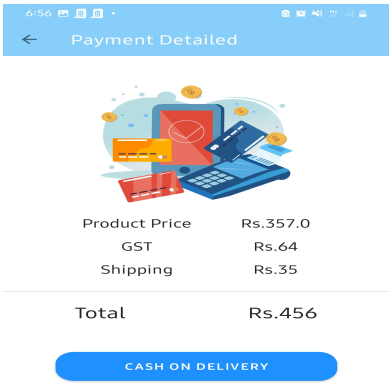

Figure 5.9: Payment Activity

Chapter 6

Conclusion and Future work

SmartMedics, the online pharmacy ordering application, offers a trans-formative solution for individuals to conveniently access and manage their medications. By integrating smart medical assistance features, personalized recommendations, and secure payment and delivery services, SmartMedics enhances medication accessibility, promotes medication adherence, and empowers users to take control of their health. The application provides a user-friendly interface, comprehensive medication information, and real-time monitoring, leading to an improved healthcare experience and better health outcomes. SmartMedics can leverage data analytics techniques to derive meaningful insights from users' health data. This can help identify trends, patterns, and potential health risks, enabling proactive interventions and personalized recommendations.

Implementing AI-powered chatbots can enhance user engagement and provide instant support for medication-related queries, prescription information, and general healthcare inquiries. Continuously refining the user interface and incorporating user feedback would help optimize the app's usability, making it more intuitive and user-friendly. SmartMedics can further expand its services by incorporating additional healthcare products such as medical devices, health supplements, and wellness products. This expansion would provide users with a one-stop solution for their healthcare needs.

References

[1] Google Developer Training, "Android Developer Fundamentals Course-Concept Reference", Google Developer Training Team, 2017. https://www.gitbook.com/book/googledeveloper-training/android-developer-fundamentals-course-concepts/details.

[2] Erik Hellman, "Android Programming-Pushing the limits", 1st Edition, Wiley India Pvt Ltd, 2014. ISBN-13: 978-8126547197.

[3] Dawn Griffiths and David Griffiths, "Head First Android Development", 1st SPD Publishers, 2015. ISBN-13: 978-9352131341.

[4] Bill Phillips, Chris Stewart and Kristin Marsicano, "Android Programming: The Big Berd Ranch Guide", 3rd Edition, Big Nerd Ranch Guides, 2017. ISBN-13: 978-0134706054.