SWIGGY GENIE CLONE APPLICATION

Project report submitted in partial fulfilment of the requirement for

the degree of Bachelor of Technology
in

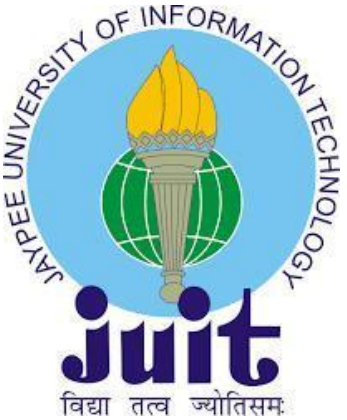
Information Technology

By

Nitika Sharma (191517)

Under the supervision of
Prof. (Dr.) Pradeep Kumar Gupta

to



Department of Computer Science & Engineering and
Information Technology

**Jaypee University of Information Technology Waknaghat,
Solan-173234, Himachal Pradesh**

CERTIFICATE

This is to certify that the work which is being presented in the project report titled "Swiggy Ginie Clone Application" in partial fulfilment of the requirements for the award of the degree of B. Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by "Nitika Sharma, 191517" during the period from January 2023 to May 2023 under the supervision of Dr. Pradeep Kumar Gupta, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Nitika Sharma (191517)

The above statement made is correct to the best of my knowledge.

Dr. Pradeep Kumar Gupta
Professor
Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat, Solan, HP.

I

PLAGIARISM CERTIFICATE

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ……..

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ __Department: _____ Enrolment No _____

Contact No. _____E-mail. _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ……(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

……

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

II

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to Almighty God for his divine blessing that makes it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor **Dr. Pradeep Kumar Gupta**, **Professor**, Department of CSE & IT Jaypee University of Information Technology, Wakhnaghat. Deep Knowledge & keen interest of my supervisor in the field of **"Android development"** to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts, and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to **Dr. Pradeep Kumar Gupta**, Department of CSE & IT, for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patience of my parents.

Nitika Sharma (191517)

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

1. ADK: android development kit
2. DDMS: Dalvik Debug Monitor Service
3. JDK : Java Development Kit
4. JRE : Java Runtime Environment
5. OEM: Original Equipment Manufacturer
6. OS: Operating system
7. MVVM: Model View View-Model
8. SQL: Structured Query Language
9. UI: User Interface
10. OTA: Over-the-Air
11. MVC: Model View Controller

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Using Kotlin to develop Android applications has gained momentum in recent years and is widely used as an easier way to develop applications using Java. Kotlin provides many features for Android developers to easily create interactive apps. This document describes the development of the Kotlin-based Android version of the Swiggy Ginnie application. Users of the app can use it as a food delivery service to order their favourite dishes from various restaurants and have it delivered directly to their door. The application is developed with the help of Kotlin, a popular programming language for creating Android applications, which is modern, clean and expressive.

This article begins with an introduction to the functionality of the Swiggy Ginnie clone app. Next, it describes the program architecture based on the Model-View-ViewModel (MVVM). The document also explores various applications, including components, views, data structures and activities, and the relationships between them.

The following document details the app's features and functions, including user authentication, restaurant search, menu, ordering, and payment. The document also describes how to use key Android libraries and frameworks in app development, including Retrofit, Glide, Room, and ViewModel. The document ends with a discussion of the challenges and lessons learned in the application development process.

In summary, the Swiggy Ginnie Android Kotlin clone created in this article is a simple food delivery application that demonstrates the power of the Kotlin programming language and MVVM design. The application can be developed and customised according to the specific needs of various businesses and customers.

Chapter - 1
Introduction

1.1 Introduction

In recent years, more and more customers have chosen the convenience of home delivery of goods and services, which has increased the attractiveness of the market. Print for delivery when necessary. Demand business, hyper-local business, is a niche business with great growth in cities [8]. Swiggy Genie's customers are aware that the India delivery service can deliver and pick up packages in cities. The service has grown well due to its availability, durability, and affordability. This post will explain how an app like Swiggy Genie works.

We believe this research will be useful to other businesses looking to develop similar applications to improve their hyper-local deployments. feint. Users can order anything for delivery anywhere in the city using the delivery service through our app. The app offers everything you need in one click, including lost keys, documents, groceries and even laundry [2]. The software is easy to use, has a friendly user interface and many features that make the delivery process run smoothly.

The concept, design, construction, testing and delivery of the development phase are covered. Application performance, data security, and integration with third-party APIs and services are challenges we face throughout the development process. The good news is that we managed to overcome all these problems and created an application that works with careful planning and execution.

The technology we use to make this app is Kotlin. Android applications are developed using the powerful programming language Kotlin.

Developers who want to build high-quality Android applications quickly and efficiently should choose it as it is designed to be transparent and interoperable, compatible with existing Java code, as shown in Figure 1.1.

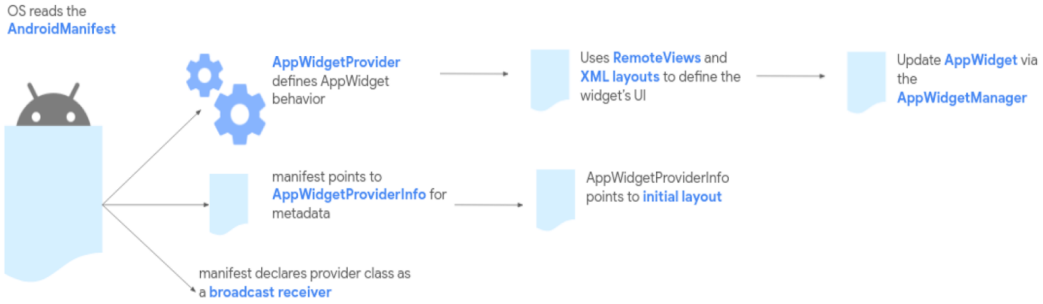


Fig 1.1 Flow of Android Application

One of Kotlin's special features is its security importance, which includes type assumptions and null safety. Errors that can occur when a programmer tries to enter a null value and cause the application to crash are prevented by null security.

Conversely, type inference enables programmers to write shorter, more interesting lines of code, reducing error time and improving readability. Support for function definitions, including Lambdas and higher order, is another important feature of Kotlin. Developers [15] can create modular and reusable code that makes it easy to maintain and update applications over time.

Kotlin also has many useful tools and libraries for developing Android applications. For example, Firebase, a popular backend platform-as-a-service offers many features such as authentication, real-time data, and cloud messaging. With Kotlin, programmers can quickly and easily integrate Firebase into their applications, allowing them to build complex and powerful applications.

Firestore, a cloud-based NoSQL database that allows developers to store and access data in real time, is another essential tool for developing Android apps. Kotlin allows developers to use Firestore to create dynamic, data-driven applications that are updated with real-time, polluting and engaging user experiences.

In addition to Firebase and Firestore in Figure 1.2 [19], Kotlin supports many APIs that can be used to improve application performance.

For example, location-based functionality can be added to an app using the Google Maps API, while social media content can be added using the Facebook API.

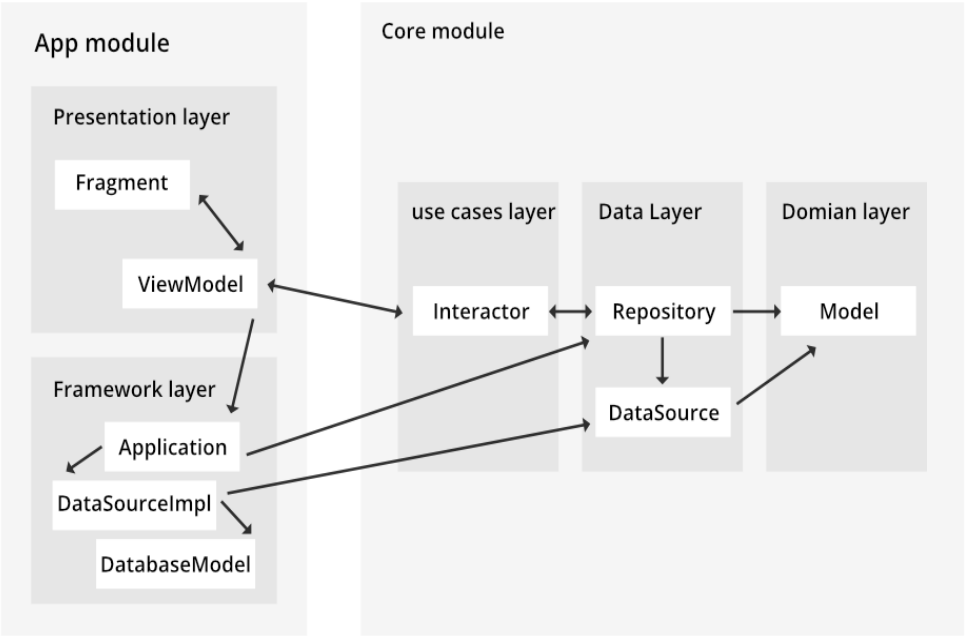


Fig 1.2 Android App architecture

Android Studio is the official Integrated Development Environment (IDE) for application development and an essential tool for Kotlin developers. Developers can build Kotlin applications more easily with the help of Android Studio, which provides many features and tools for building, testing and debugging Kotlin applications.

Kotlin is a powerful and flexible framework for developing Android applications. Due to its security features, performance support, and many tools and libraries [2], it is the best choice for developers who want to build quality, fast and efficient services. Kotlin developers can create powerful, data-driven apps that provide connectivity and enhance user experience by leveraging resources like Firebase, Firestore, and APIs.

1.2 Problem Statement

The shipping industry is growing rapidly because more and more people rely on them to move goods from one place to another in major cities. However, current delivery services often fail to meet customers' needs for convenience, speed, and security.

The aim of this project is to create a delivery application using Android Kotlin to solve these problems and provide customers with a connected, efficient and secure delivery [1]. This initiative specifically aims to address the following issues:

- **Limited distribution options/ Delivery Options:** Consumers' delivery options are currently limited to a small number of service providers, resulting in limited delivery options. This can increase the customer's need for quality and on-time delivery.

- **No real-time tracking:** Export models often lack real-time tracking, making it difficult for customers to track their products and get real-time updates.

- This is done in our project. In this way, users can keep track of what they are sending. Track with a delivery partner.

- **Poor communication:** When users interact with delivery personnel, poor communication and poor communication will lead to delays, incorrect deliveries and other problems. This causes app users to distrust delivery partners.

This work is designed to solve these problems and create a delivery application to provide customers with easy, efficient and safe delivery. My goal is to provide customers with a delivery service that meets their needs and exceeds their expectations [11]. For this purpose, I created an application with multiple delivery options, time tracking, effective communication and security measures. Due to rapid changes in the current environment, customers want fast and efficient quality of their products. With the development of e-commerce and online shopping, it has become important to have a reliable delivery service that can provide customers with quality and smooth delivery.

However, delivery apps currently on the market often suffer from slow delivery, inaccurate tracking, and poor customer experience. Therefore, users are not satisfied and the sales of the application suffer. Therefore, there is a need for a delivery application that can overcome all these problems and offer fast, reliable and clear delivery to customers.

Android developers are increasingly using the modern programming language Kotlin because of its many advantages and features. Kotlin's best performance and best performance are its two advantages in application development.

This is because Kotlin uses native objects instead of web objects, helping to make applications faster and more efficient while providing users with a smooth and seamless experience.

Kotlin is versatile as well as efficient and can be used to create unique, personalized user experiences based on project or organizational goals. This has been made possible by Kotlin's architecture, adaptability and versatility, and how easy it is to incorporate certain changes and effects into the application.

Kotlin's syntax and definitions are one of its main strengths as it makes it easy for programmers to write readable and manageable code. This is achieved using a variety of methods including type assumptions, failsafes and continuous operations that help reduce the amount of common code required [7].

Besides being efficient, Kotlin is versatile and can be used to create unique and user-friendly experiences based on the specific needs of a project or organization.

The purpose of our application is to make users understand better. The program should be easy to use, with easy-to-understand instructions and a user-friendly user interface that guides users through the deployment process. We set out to create an app that would provide a similar experience for hyper-local delivery, allowing customers to ship and receive products quickly and easily. The objectives of this application are:

Ensure reliable delivery: The program must ensure that the product is delivered as planned, clear notifications and standard monitoring layers to inform users.
Therefore, it provides a good delivery service.

- **Supports multiple delivery methods:** Application, food, medicine, information, tools, etc. to meet the needs of most users. It should provide multiple distribution methods, including

- **Fairly Priced:** Apps must provide customers with fair service at a fair price without sacrificing reliability or performance. Therefore, the delivery cost will vary according to the distance between the place and the place.

- **Delivery Package Status:** Users can see the status of packages delivered on time, and the program allows partners to track their way on time [18]. This allows users to keep track of what they post. See delivery partners.

- **Communication and Customer Service:** Users of this application will be able to chat with delivery people and solve their problems. User can also contact the manager or customer if there is delivery problem, if selected.

1.4 Methodology

Creating a Kotlin project requires some important methods and techniques to develop a good and usable script. Identifying an application's requirements and specifications, including key features, functionality, and user experience, is the first step in development. Creating a specific brand requires the collaboration of stakeholders and end users to identify their needs and preferences [2].

Once the development team has determined the requirements, they can start building and deploying the application using Kotlin. To do this, we use a variety of tools and techniques to build a reliable, scalable architecture that supports application performance and user experience.
One of the many advantages of using Kotlin for Android app development is its integration with various tools and technologies used in app development. For example, Firebase provides many backend services such as authentication, storage, and hosting, which can all be quickly deployed in an application using Kotlin.

Teams can store and manage data in FireStore, similar to how they manage and store user information, orders, and payment details in their apps. For this, a data model must be created in Kotlin that can transfer data from the application to the FireStore database. Various API endpoints need to be created so that users can communicate with data.

While the application is being developed, the team must also test and evaluate the application to ensure that it works well and complies with guidelines and specifications [10]. For this purpose, various tests and tools (such as Espresso, Mockito and JUnit) are used to find and fix bugs or other problems.

Once the app is created and tested, it can be placed in the app store for users to download. This requires packaging the software in Kotlin, submitting it to the app store, and providing ongoing support and maintenance to keep it running and meeting customer needs.

Following these guidelines and making the most of Kotlin and other tools and technologies will enable developers and contributors to create great, user-friendly products.

To do this, we must follow the following guidelines:

- **Keep it simple:** The design should be simple, uncluttered, and easy to navigate. It is recommended to avoid complex interfaces and difficult navigation of products.

- **Consistency:** Consistency in design is very important.

- **Maintaining simplicity**: We must use the same layout, fonts, colors and symbols throughout the program to ensure consistency and a consistent user experience.

- **Use the right colours:** Colour selection is important in design. Using colors that match your brand and intended use. Don't use too many colors and make sure there is enough contrast to make the UI easy to understand and navigate.

XML	Frontend layout file
Kotlin Programming Language	Backend
Postman	API
Stripe	Payment gateway
Firebase	Back-end authentication

Table 1.1 List of technologies stacked for application

All necessary actions are taken after creating the user interface of our program and determining the needs of the users. The second is the selection of the feature [13]. The main technology used to create mobile applications for Android is the Kotlin programming language. Postman for API connectivity and Firebase for backend services are some additional technologies that can be used. Google's cloud-based Firebase platform provides many tools and resources for building and deploying mobile and web apps. Programmers can build apps faster and more efficiently thanks to Firebase's wide array of backend features.

Google's Firebase has many capabilities, including authentication and cloud messaging. Programmers can use real-time cloud communication to send notifications to users on multiple platforms, including iOS and Android, as well as the web. When users select the machine, the most important step is to enter the code. During development, the code of the application is written. according to the scheme 1.1. Building the front-end and back-end of the program, connecting to the API, and testing it for bugs and other issues is all part of the process.
The most important step, entering the code, is the next step users take after selecting a device. The application number is collected as part of the development process. Figure 1.3 shows. For this, front-end and back-end programs must be created, APIs must be integrated, and the application must be tested for errors and other problems. The application is tested throughout testing and quality control to ensure it meets its needs and functions. This includes user validation as well as testing and integration analysis [5].

In this process, it is important to follow recommendations, monitor all work, and ensure that policies are changed and managed in a way that supports future reform and improvement.

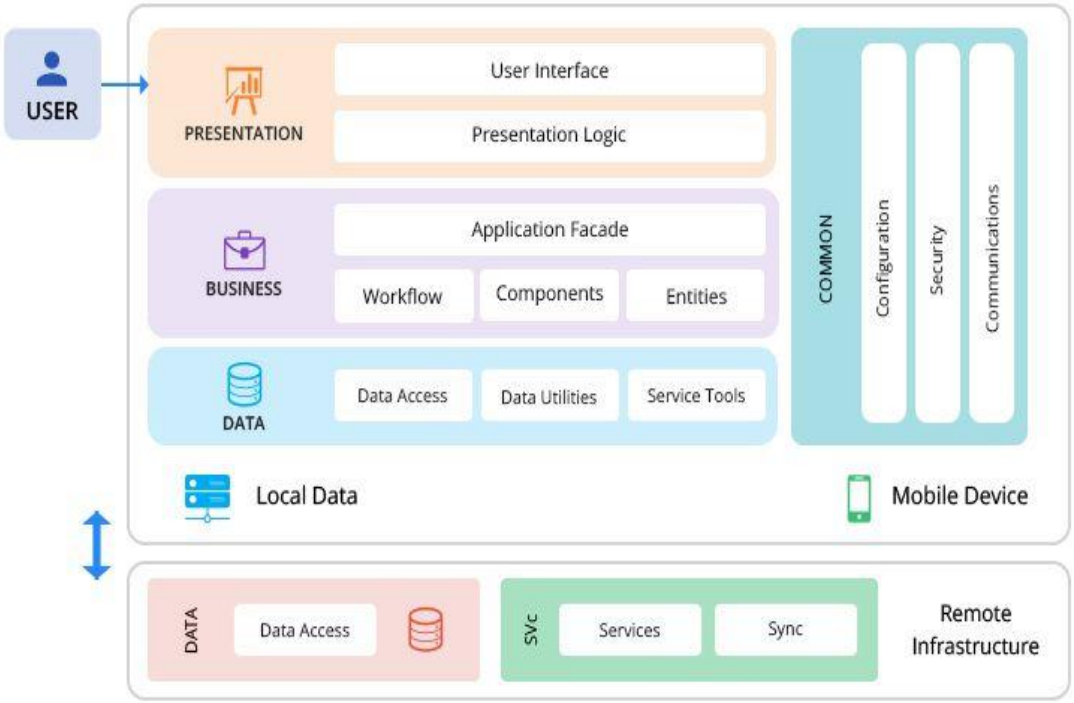


Fig 1.3 Data fetching from back-end and rendering on front-end

Figure 1.4 shows the difference between Kotlin and other technologies. Given that each language has its own strengths and weaknesses, neither Rust, TypeScript, nor Python can be better than Kotlin. TypeScript adds an option such as typing to the big web, Python emphasizes simplicity and readability in many areas, while Rust emphasizes performance, reliability and security for the production of low-cost systems [17]. Kotlin's simplicity, Java compatibility, and Android development support make it useful for many tasks, but each language may be better suited for certain applications. Which language to use in the end depends on the context and the specific requirements of the project.

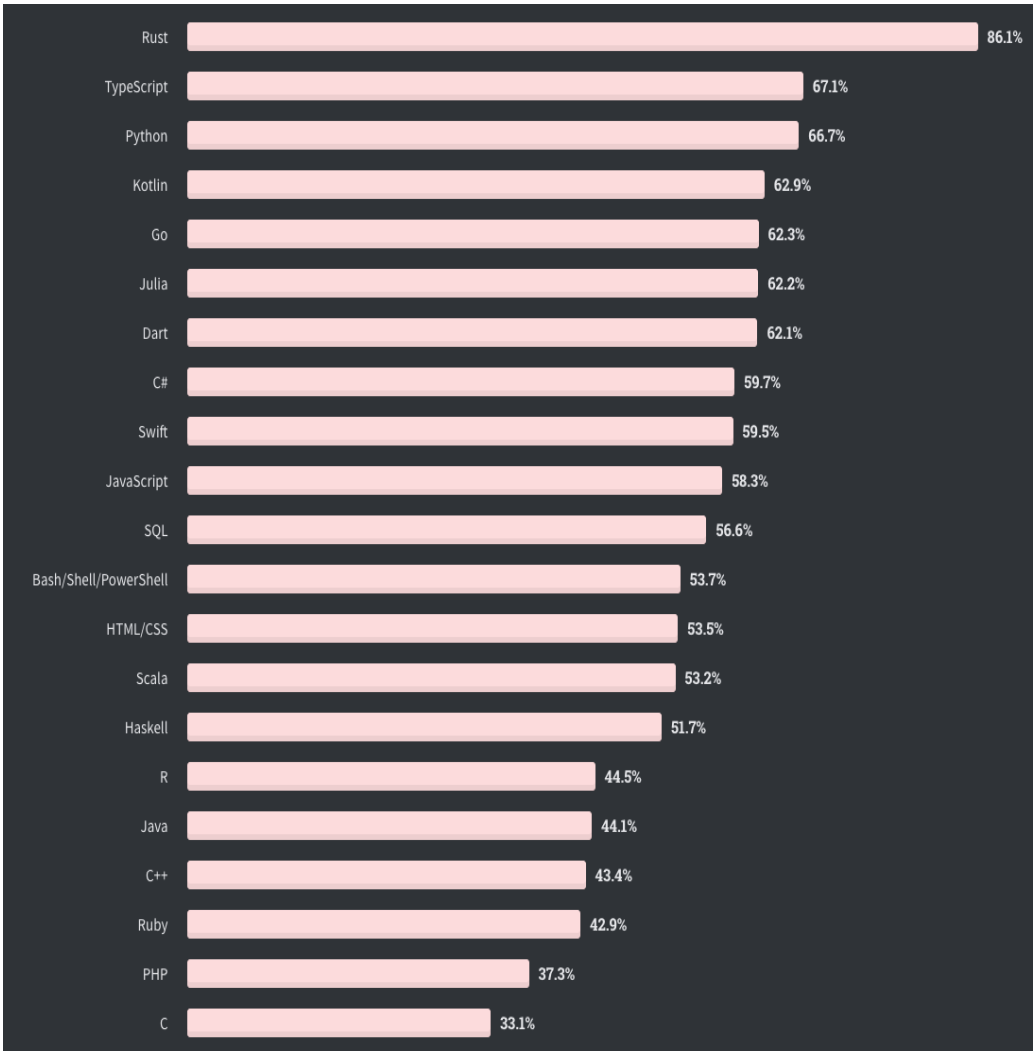


Fig 1.4 Usage of kotlin in comparison with other language

Chapter-2
Literature Survey

The article titled "Developing Mobile Applications for Traffic Monitoring and Management" [1] discusses the development of mobile applications for traffic monitoring and management in India using Android Studio and Firebase.

This article discusses the use of Android Studio and Firebase to develop a mobile application to manage and monitor traffic in India. Users can report traffic violations, accidents and accidents through the application, which also provides real-time traffic information. Managing communication, storage, and authentication, Firebase is the core of the application backend. The user interface, database and API connections, and other services are described in the whitepaper. User opinions are collected while evaluating the functionality and user interface of the application [12]. The app is a useful tool for managing traffic in India and the authors' conclusions are general.

The purpose of the application is to record traffic violations, accidents and accidents and provide real-time updates on the traffic situation. This white paper provides an in-depth analysis of the application's UI design, backend architecture, and functionality. It also shows how Kotlin language is similar to Java in Android application development.

The article ends with general advice on developing a mobile application for management and monitoring in India. The program's features, backend operations and user interface layout are covered. This study also discusses effectiveness and user feedback. The app mainly helps to manage traffic in India and is likely to get more improvements in the future.

The article "A Review of Server-Side Development with Kotlin" summarizes the Kotlin programming language and its server-side development potential [2]. Kotlin's history, development, features and advantages over other programming languages are explained in the opening words of the article. The authors also cover the use of Kotlin in various software development areas, including Android apps, websites, and server-side code development.

After, the article goes into the details of using Kotlin for server-side development, showing how it works with well-known platforms like Spring Boot, Ktor, and Vert.x. The authors examine the benefits of combining these models with Kotlin, including code cleanliness, simplicity, and performance.

The article also explains the advantages of using Kotlin for server-side programming over other languages. For example, Kotlin's type inference and null security features help prevent runtime issues and solidify code. The article also talks about Kotlin's relationship with Java, which enables programmers to use existing Java libraries and tools.

This article also discusses Kotlin development tools such as Gradle build system and IntelliJ IDEA integrated development environment (IDE). Also, the author highlights the advantages of Kotlin by giving some examples of server-side development projects built using Kotlin.

The article ends by highlighting Kotlin's server-side programming potential and its reputation among software developers. The author considers Kotlin to be the best choice for server-side programming due to its clean syntax, type-safe, non-safety, and compatibility with Java. This article gives a good overview of Kotlin's server-side development capabilities and advantages over other programming languages.

The article "A Comparative Study of Java and Kotlin for Android Mobile Application Development" [3] compares Java and Kotlin as programming languages for Android mobile application development. Before starting the comparative analysis, the authors briefly describe the syntax and main features of each language.

The study focused on a few key factors such as development time, code readability, security and performance. The authors concluded that Kotlin has several advantages over Java, including faster development time and greater readability. According to the authors, these results are due to Kotlin's clean syntax, security features, and functionality. Also, Kotlin supports features that Java does not have, such as data classes, continuous functions, and coroutines. Author

also pointed out some of Kotlin's shortcomings, including language incompatibilities with some third-party libraries and tools, and the need for developers to learn new languages.
They also recognize that Java's broad community support and robust environment make it a popular choice for application development.

Overall, the study shows that programmers see Kotlin as an alternative to Java for developing Android applications, especially for beginners who can benefit from its words today. The authors advise developers to monitor the evolution of both languages and their ecosystems, and choose the language that best fits their project's specific needs.

The article "Describing the Migration of Android Apps to Kotlin: A Study on F-Droid, Play Store, and GitHub" focuses on migrating Android apps from Java to Kotlin. To understand the changes affecting Kotlin[4] adoption and transition, the study analyzed more than 120,000 Android apps on F-Droid, Google Play Store, and GitHub.

The study found that Kotlin's popularity has increased since it was introduced in 2011, and since Google announced Kotlin as the Android app development language in 2017. Most of the check references are still built in Java, but the use of Kotlin is increasing, especially on GitHub.

An additional finding of this study is that Kotlin is more likely to be used in large companies and applications with more users. There will also be more Kotlin adoption in applications that use well-known libraries such as Retrofit and Dagger. The author of the article states that Kotlin provides developers with benefits such as better readability, less standard code, and better compatibility with Java .

The results of the study offer suggestions for further research on the use of programming languages and the factors affecting language change.

This article titled "Improving Android Application Performance Using RecyclerView" describes the recycler view used in Android. RecyclerView[5] is a performance-enhancing cross-threading tool for application development. ListView has been replaced by the more complex and flexible RecyclerView, which provides better performance and better memory management. The author compares RecyclerView's performance with ListView and GridView by measuring the time it takes to load and scroll data.

The results show that RecyclerView outperforms the other two methods in terms of memory usage and scrolling performance. Additionally, the author's approach integrates RecyclerView with Model-View-ViewModel (MVVM) design. They discuss the advantages of MVVM in terms of code organisation and separation of concerns.

The authors show how to build a media application using MVVM and RecyclerView in their latest research. They provide a detailed description of the process, including UI design and implementation of RecyclerView and MVVM components. The findings show that MVVM with RecyclerView can improve application usability and performance.

To improve performance and provide a better user experience, the article highlights the value of using RecyclerView and MVVM when developing Android applications. The authors provide case studies to demonstrate the effectiveness of these methods and provide general guidelines for application.

In order to manage schools' data efficiently and effectively, this article describes how to create an intelligent SIAKAD Android application using RecyclerView [6]. The application is designed to provide a user interface for teachers and students to manage educational activities such as enrollment, time management, attendance and grading. The data is stored in a server-hosted MySQL database and the application is built using the Android Studio development environment.

Large files are displayed in a list by the RecyclerView component, allowing users to browse files uninterrupted. The advantages of RecyclerView over traditional ListView in terms of performance and memory usage are also described in the article. To ensure compatibility and reliability, the program has been tested on various hardware platforms with different screen sizes and operating systems.

According to the author, using RecyclerView when creating a Smart SIAKAD Android application can provide better performance, faster load times and a greater user experience. These improvements can increase the efficiency and effectiveness of schools.

This article compares three models (MVC, MVP, and MVVM) for developing Android apps. The authors examine and compare architectural models based on four key metrics: scalability, maintainability, testability, and development time. The case study [7] includes creating a sample application using each model for analysis. The author believes that each of our temples has advantages and disadvantages. MVC is the simplest and most used design, but it has issues with testing and lack of concern. MVP improves MVC by separating view from action and allows for better testing, but still has scalability and control issues. MVVM continues to improve MVP and provide better scalability, scalability, and security by adding data binding and greater separation of concerns. However, it will take more time to develop due to its complexity.

This study attempts to demonstrate the benefits of the MVC model in improving the design and maintenance of Android applications. According to studies, MVC is the most used model in analysis. While it has many differences from the traditional MVC pattern, the way it uses MVC is very different. According to the authors, there are four variations of the MVC pattern in the working application: Model 1, with front controller; Model 2 with monitor; and Passive View, no central control.

This study also explores the impact of the MVC model on quality policies and maintenance. Based on uncomplicated numerical analysis, the authors found a positive association between the use of MVC and policy management. However, there is no clear link between the use of MVC and the flaw, indicating that other factors may play a role when problems occur in Android applications. Overall, the study provided a good overview of the use of the MVC model and its benefits in application development.

Chapter - 3
System Development

3.1 Analysis

This step involves collecting, analyzing, analyzing, compiling and analyzing the code and functionality of the application [5]. It is necessary to find the main content of the application, including shipping, billing and delivery, such as registering a new user, entering the registered user, verifying the delivery and sending the content to the person.

We developed a design based on these specifications, including design, user interface, and database organization. Create document design process specifications.

We need to build a system with core processes and tools like Android and Firebase (for backend services). Implement logical processes, provide backend services and create user interfaces [13]. Then we need to test our application. The main purpose of the implementation phase is to build a platform using Android, Firebase and other programs and models. The UI is created using objects in Android Studio and styling is done using XML.

Cloud database storage and authentication are features provided by Firebase to create backend services. The logic of the system is implemented with programming languages such as Kotlin.

Backend as a Service (BaaS) platform Firebase is well known. Cloud storage, Realtime Database, authentication and many other services are available in Firebase's mobile and web apps. Google's Firebase is a platform that provides many tools and services for developing Android apps.

The following are some advantages of using Firebase in Android projects written in Kotlin:

1. **Realtime Database:** It is easy to build Android apps using Firebase's Realtime Database. This allows developers to easily collect and store data in real time, which is useful for messaging, chat applications, and other applications that need to change over time.
2. **Authentication:** User authentication in Android apps is possible using Firebase's user-friendly authentication solution [20]. The system supports multiple authentication options including social media, phone and email/password login.
3. **Cloud Messaging:** Firebase provides a cloud messaging service that can be used to notify Android app users via push notifications. This will help users interact with the app even when the app is not being used.
4. **Analytics:** Android apps can use Firebase's powerful analytics to monitor user behaviour. To improve the user experience of the application, the technology provides insight into how users interact with the application.
5. **Crash Reporting:** App crashes can be tracked and reported using Firebase's crash reporting system. This makes it easy to find and fix flaws in the app.
6. **Hosting:** Also, Firebase offers hosting that can be used to store static files such as HTML, CSS, and JavaScript. You can use it to host landing pages, business websites and other online content related applications.

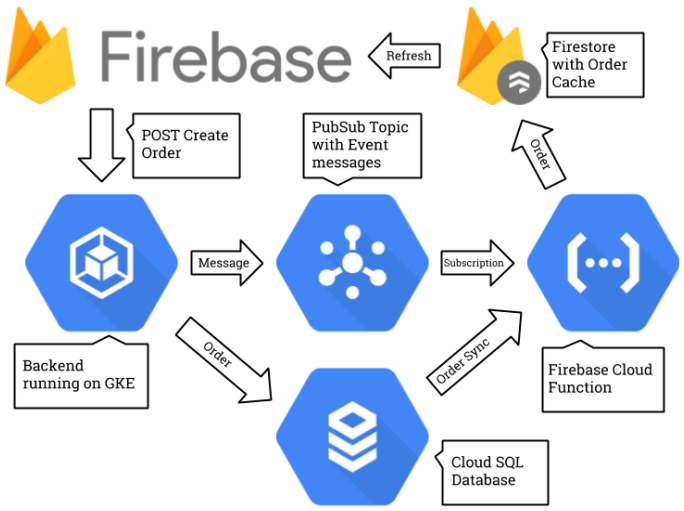


Fig 3.1 Firebase Model

In summary Firebase provides various services and tools for Kotlin-based Android application development, making it easy for developers to build good apps with complex tasks as shown in Figure 3.1.

3.2 Design

Creating a plan for the project submission is part of the design process at this stage. Architecture, user interface, and database schema are covered. The structure of the system plan includes a description of the system's backend services, APIs, and client interfaces, and how these elements work together. The layout of the user interface should follow best practices and guidelines for accessibility and usability. Data generation requires detailed knowledge of data structures, links between sources, and data collection methods [6].

On Android, "XML" is a formatting tool that uses HTML-like syntax. A quick and efficient way to specify the structure for Android app development content is to use XML.

1. **Separation of concerns:** XML provides a clear separation of design and functionality, making it easier for designers and developers to work independently on multiple aspects of a project. As shown in Figure 3 below.

```
<Button
    android:id="@+id/sendVerificationCodeButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/frame_215"
    android:layout_marginTop="442.00dp"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent" />
<TextView
    android:id="@+id/titleTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Sign Up"
```

Fig 3.2 XML layout file

2. **Reusability:** Using XML you can create reusable layouts that can be used across multiple screens and layouts; 3.3.

```
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
```

Fig 3.3 Colour layout file

3. **Flexibility:** The many layout options provided by XML make it easy to create complex and dynamic UI designs for different sizes and orientations, as shown in Figure 3.

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

Fig 3.4 Layout type

4. **Purpose:** XML is easy to learn and use, even for developers without a design background [9]. With the simplicity and intuitiveness of the syntax, developers can quickly create and modify user interface design.

5. **Accessibility:** XML provides accessibility features such as font sizing, screen reader compatibility, and touch target size to make the software accessible to people with disabilities.

Designing of the project is distributed into certain levels. These levels are :

- Authentication
- Product and location selection
- Payment

Figure 3.5 shows the ER model for user authentication. If someone is new to the program, they need to sign up, and if they are already registered, they just need to sign in. Users can verify their identity in both [17] using only their phone number. The OTP will be sent to the registered user's mobile number.

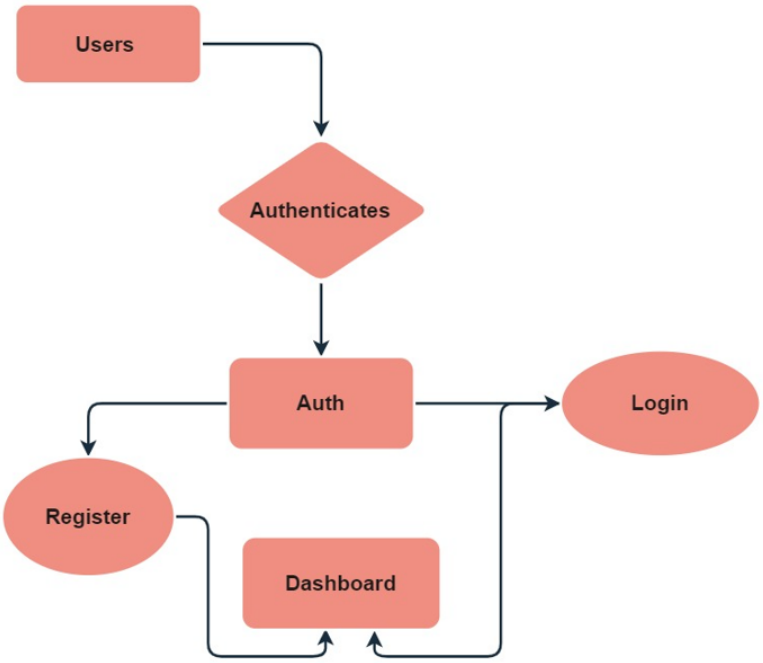


Fig. 3.5 ER diagram of user authentication

Choice of Product and Location

After signing in and scrolling through the application, the user can choose a place to wait and send the location for the post. The user must enter the following information in this form of our application:

- Location for pickup
- Destination
- Information on the deliveries that are required
- Some guidelines on how to deliver the product to the delivery partner.

The process is shown in Figure 3.6. After selecting the destination, a map will appear for the user to select the destination. I created this map using part of the Google Maps API on Google Cloud Platform [1].

Use the same method to select the location. Then the user has to add what he wants to send. Additionally, users can specify the instructions delivery partners should follow when making deliveries. He told me where to go, following the roadmap for the delivery man.

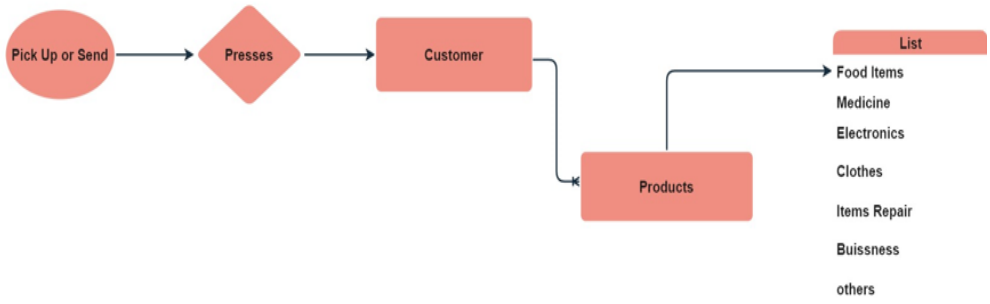


Fig. 3.6 ER diagram of product and location selection feature.

After choosing a location and location, Google Maps API will determine the best and fastest route to get there and provide the distance between the two locations. The shipping cost depends on this distance [14]. Once the location and details are confirmed, the user will receive a confirmation and a complete checkout page. The compensation ER diagram is shown in Figure 3.7. Users can add new cards to complete transactions.

The user chooses the card they want to use each time they want to make a payment; if he wants to add a new card, a dialog will pop up in the middle of the screen where he has to enter the card information. Evaluate the front of the card if the card information is correct. The background uses Stripe to determine if the card information matches an existing card and whether the card is usable. We will examine this process in more detail in the next section.

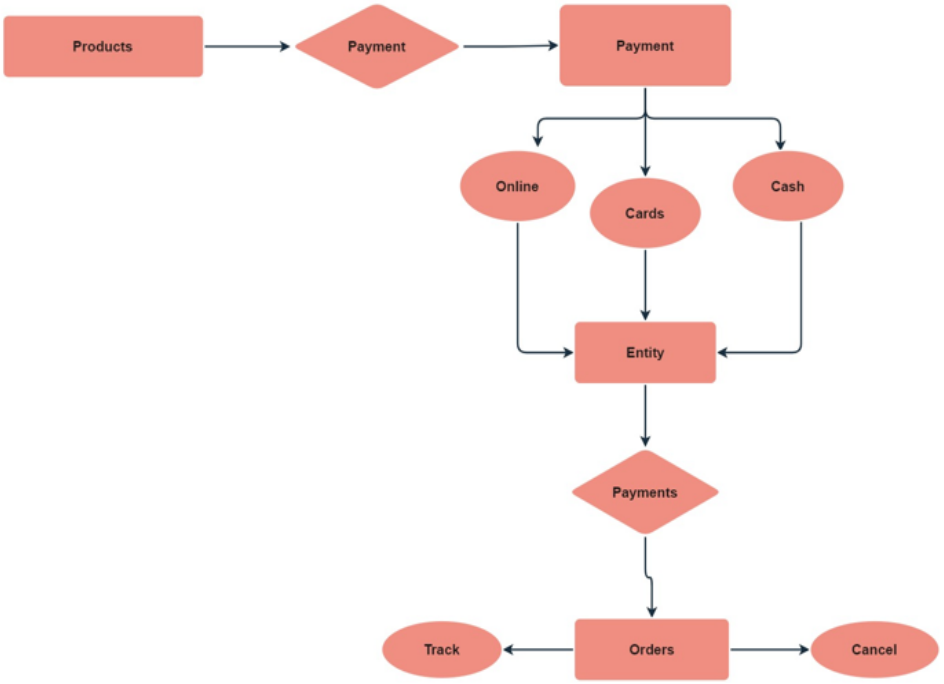


Fig 3.7 Showing the flow of payment methods

In addition, the user of this application will be able to access and change their profile, rate and review the delivery partner, view purchase history, and receive notifications[8]. The user will be notified that the package has been delivered as soon as it arrives at its destination. Following that, the user has the option to grade the driver according to their behaviour and timing of deliveries. They can also leave feedback for the delivery partner. The user has the ability to view every review and rating he has ever given a delivery partner as in Fig 3.7.

The user will be logged in as a former user or the most recent user each time he or she launches the application. The application's welcome screen will be returned to the user after using the logout feature, where the user can sign in again using a different cell phone number or create a new user[3]. The entered cell phone number must be functional for the user to get an OTP on that specific number as in Fig 3.8.

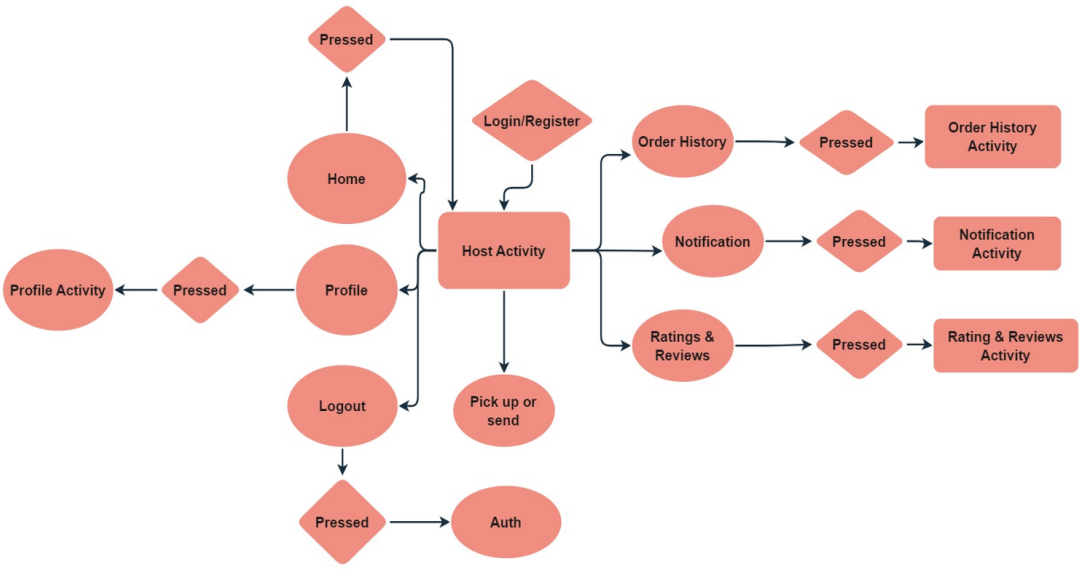


Fig 3.8 ER diagram of application

3.3 Development

This section focuses on examining the structure of our application in detail. This part of the report on Android application design also includes the tools and techniques I used and the problems I encountered [12]. Details of the strategy and architecture used in this project are available here.

- Android Studio
- XML
- MVVM
- Firebase
- Stripe
- Gradle

We will now delve into each technology and how we use it in our projects, as well as running and using them from the first development sessions of the app. to use. in different technologies used on stage.

Android Studio

The official Integrated Development Environment (IDE) for developing Android applications is called Android Studio. It is based on IntelliJ IDEA developed by Google. Developers can create, test, and publish high-quality Android apps using the tools and resources provided by Android Studio, specifically designed for developing Android apps.

Android Studio provides developers with a comprehensive set of tools, including a code editor, layout editor, debugger, emulator, and many additional tools [19]. It also supports many programming languages like Kotlin, which was recently approved by Android developers for its type inference, simple syntax, and compatibility with Java.

Kotlin programming is a statically typed language that uses the Java Virtual Machine (JVM) and can also be translated to JavaScript or native code. Simplicity, security, and compatibility with Java are Kotlin's design goals. Due to its ease of use and new features, it has grown in popularity among Android developers since it was released in 2011 by JetBrains, which also developed IntelliJ IDEA.

Developers can use Android Studio features to build Android apps using Kotlin, including:

1. **Code Editor:** A powerful code editor with Android Studio support for Kotlin syntax highlighting, code completion, and refactoring. Additionally, the code editor includes live templates, shortcodes, and templates to help developers write code faster and more efficiently.

2. **Layout Editor:** Developers can use the [8] layout editor in Android Studio to visualize the UI design of Android applications. The layout editor supports drag and drop widgets and also provides a preview of the user interface.

3. **Debugger:** The debugger in Android Studio allows users to drill down into every line of Kotlin code. Additionally, the debugger provides the ability to define variables, create breakpoints, and evaluate instructions.

4. **Emulator:** Android Studio comes with an emulator that allows programmers to test Android applications on an Android-based computer. The emulator is compatible with various Android versions, hardware setups and screen sizes.

5. **Build system:** Gradle is the design framework used by Android Studio that gives developers flexibility and efficiency in building, testing, and deploying Android apps. Creating different versions of your app for different environments is easy because Gradle supports different builds, types and types.

6. **Android Jetpack:** Google provides libraries, tools, and instructions for developing Android apps, called Android Jetpack. Data storage, UI, navigation and other features are all included in Android Jetpack[3]. It provides backward compatibility to previous versions of Android and aims to make Android development faster and easier.

There are many advantages of using Android Studio and Kotlin for Android application development:

- **Faster development:** Android Studio provides many tools and resources to help developers build Android applications faster and more efficiently. Kotlin's type inference and compact syntax also make coding easier.

- **Better code:** Kotlin is designed to be safer and more extensible than Java, which can result in fewer lines and better code. The Code Analysis feature in Android Studio also helps developers find and fix bugs in their code.

- **Better User Experience:** A better user experience is possible thanks to the layout editor in Android Studio, which allows developers to design beautiful and responsive user interfaces for their Android apps[14]. Additionally, Android Jetpack has elements for handling user input, sharing information, and switching between screens.

- **Marketing:** With over 2 billion active devices, Android is the world's most popular mobile app. Using Android Studio and Kotlin when developing Android applications can reach a wide and diverse audience.

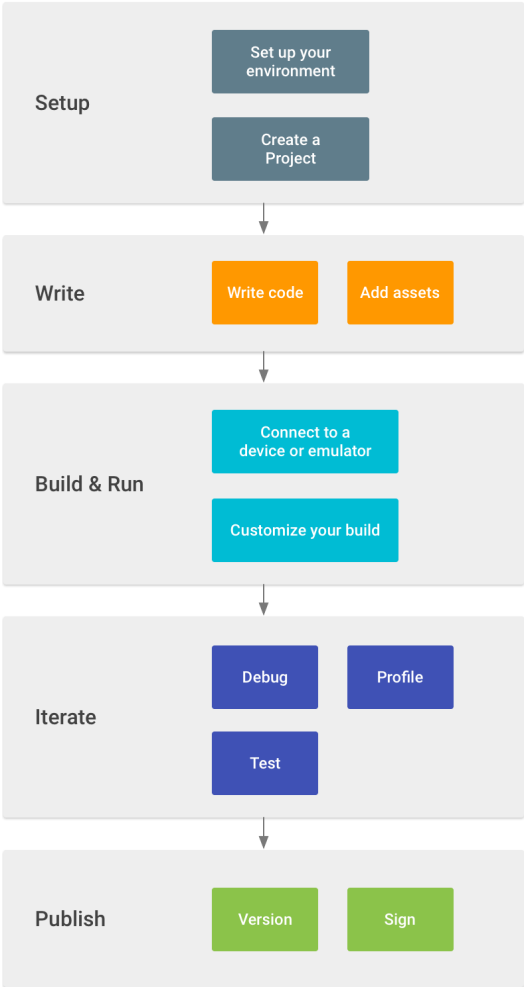


Fig 3.9 Android Studio setup

In summary, Android Studio is a powerful development environment (IDE) for Android application development that provides developers with a variety of features and tools to make good Android applications. As shown in Figure 3.9, today's Kotlin programming language is designed to be secure, extensible, and compatible with Java. Developers can develop applications in an integrated way using Android Studio with Kotlin.

XML

Documents and documents are often created using a markup language called XML (Extensible Markup Language). XML is often used in Android application development to define the application's configuration and user interface. Buttons, text, images, and other visual elements are described in XML files that describe the layout and content of Android app screens.

Tags and attributes are used to denote objects and their attributes in a hierarchical, human-readable language called XML. Text-based XML files can be easily edited using a text editor or an IDE such as Android Studio. The "res" folder in an Android application is usually where XML files are stored.

Here are some key advantages and features of using XML when developing Android apps:

1. **Separation of concerns:** Using XML, developers can store the application UI and structure separately from the logic code. This separation of tasks simplifies the management and maintenance of the code base and improves collaboration between designers and developers.

2. **Declarative syntax:** Because XML uses declarative syntax, programmers can specify how they want the user interface to look instead of writing code to modify it directly. This improves the readability and understanding of the code.

3. **Flexibility:** Using XML, programmers can create user interfaces that can be adjusted to fit different sizes and orientations.

4. **Reusability:** XML layouts can be used for different Android apps and activities [16].
   This reduces code duplication in programs and allows developers to create a uniform user interface.

Developers typically do the following to build Android applications using XML and Kotlin:

- Define the application configuration by creating an XML file that specifies the organization and content of the application screens. To do this, you must define views (such as buttons, text, and images), specify their properties (such as size, position, and color), and place them in a hierarchy.

- Using Kotlin, programmers can create the logic and behaviour of applications by connecting logic to code. Developers use view bindings (the process of creating binding classes that allow developers to view Kotlin code) to interact with the requirements defined in the XML documents paper layout.

- Managing user interaction: Developers use Kotlin to show a view [2] how the program interacts with the user. Callbacks and event handlers are defined to be activated when the user interacts with the view.

- Application testing: To ensure that the application works as expected, developers test the application using benchmarks in Android Studio. The user interface, logic, and behavior of the application are part of the experiment.

Some common XML elements and attributes used in Android app development includes in Table 3.1:

View/ Attribute	Description
Linear Layout	A layout that arranges views in a single column or row.
Relative Layout	A layout that arranges views relative to one another.
TextView	A view that displays text.
ImageView	A view that displays an image.
Button	A view that represents a clickable button.
android:id	An attribute that assigns a unique ID to a view, which is used to reference the view in Kotlin code.
android:text	An attribute that sets the text displayed in a TextView or Button view.
android:layout_width	An attribute that defines the width of a view.
android:layout_hieght	An attribute that defines the height of a view.

Table: 3.1 XML attributes

In a nutshell, application layout and user interface are defined using XML [8], the powerful and adaptive markup language often used to build Android applications. XML, when used with Kotlin, enables programmers to create powerful and user-friendly interfaces that are easy to modify and maintain. Learning how to develop Android applications requires an understanding of XML and how it is used in Android applications.

MVVM Architecture

Modern Android application development mostly uses MVVM (Model-View-ViewModel) architecture. With MVVM, a program's user interface (View), business logic (ViewModel), and data model (Model) are clearly separated into their respective areas of interest [20]. Because of this separation of concerns, developers can make software that is modular, scalable, and easy to maintain.

The main elements of the MVVM architecture are:

- **Model:** The data model describes the data in the application and the functions that can be used for it. You can think of model layers as files that provide information to model view layers.

- **View:** The UI layer represents the visual content of the program and supports user interaction. Information displayed to the user and input from the user are handled by the display layer.

- **ViewModel:** Between the View layer and the model layer, the ViewModel layer acts as an intermediary. User input and other events are handled by the ViewModel layer, which also provides data to the View layer [5]. The application business logic resides in the model view layer, which is also responsible for performing the necessary data transformations or calculations.

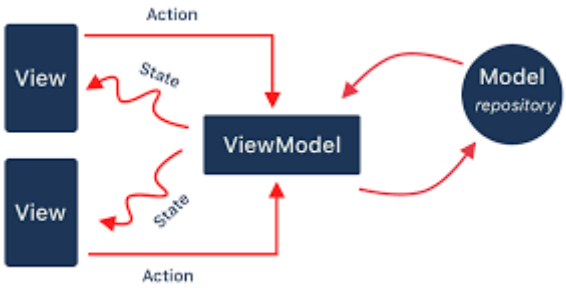


Fig 3.10 MVVM architecture

Here are some advantages of the MVVM architecture in Figure 3.10 for Android Kotlin application development:

1. **Separation of concerns:** MVVM architecture breaks down the concerns The application goes to three additional layers that cause many differences. code, easy to manage. Additionally, this section allows developers to work on many aspects of the project simultaneously [14].

2. **Testability:** Because the business logic of the application is deployed in the ViewModel layer, it is easier to test the functionality of the application alone. This makes it easy to find and fix bugs in application code.

3. **Extensibility:** The MVVM architecture is designed with scalability in mind, making it easy to expand or modify the project as new features are introduced or changes are made. This makes it easier to process the application over time.

4. **Code reuse:** The MVVM architecture divides the program's tasks into several layers, facilitating code reuse for the entire project. Therefore, the programs will be fewer in number, making maintenance easier.

Developers often use the MVVM architecture when creating an Android Kotlin application by doing the following:

- **Defining the data model:** The application developer decides which data to use. To access and use data [2], it is necessary to specify the data model, database process, and API end.

- **Creating the ViewModel layer:** The developer creates the ViewModel layer to house the business logic of the application. The ViewModel layer interacts with the model layer to store and manipulate data, presenting the required data to the View layer.

- **Design View:** The view process, which represents the user interface of the application, is created by the developer. Responsibility for presenting information to the user and receiving input from the user is within the scope of the display process.

- **Using data binding:** To link the view and view model layers, developers use data binding libraries. Data binding libraries allow developers to bind data and events from a ViewModel to a View, making it easy to responsively modify the user interface (UI) for changes in the data structure.

- **Application testing:** To ensure that the application works as expected, developers test the application using benchmarks in Android Studio[17]. This requires evaluation of the application's user interface, business logic, and data interactions.

The MVVM architecture can be implemented in Android Kotlin application development using several well known frameworks. An example of such a library is Android Architecture Components, which provides libraries for managing user interface components, lifecycle management, and data extensions [1].

RxJava, a reactive programming library, is another popular choice for creating the ViewModel layer of an MVVM application. Complex operations can be solved more easily thanks to RxJava operators and APIs to deal with asynchronous data. Dagger can also be used for injection, which includes injection into the ViewModel and View layers of an MVVM application. This simplifies code management and ensures that the code is modular and extensible. Finally, Retrofit simplifies building applications, forwarding requests, and analyzing returned data by managing RESTful API calls in the model layer [19]. Together, these libraries provide powerful tools for using MVVM in Android Kotlin application development.

In summary, MVVM architecture is a useful framework for building scalable, maintainable and testable Android Kotlin applications. The Model, View, and ViewModel layers are the core of the MVVM architecture. By dividing application responsibility into several different layers, developers can create applications that are easy to manage and measure. MVVM architecture can be implemented using many frameworks and technologies such as Android Architecture Components, RxJava, Dagger and Retrofit.

Google Firebase

Google has created a platform called Firebase for the creation and development of mobile and online applications. It is a popular choice among Kotlin developers because it has many tools [15] and services that simplify development and reduce time to market.

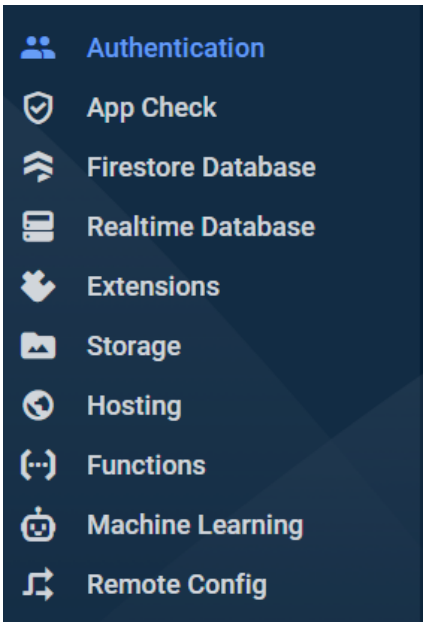


Fig 3.11 List of Firebase features

Here are some features of the Firebase app for Android:

- One option to save and synchronise data instantly is to use Firebase Realtime Database, a database hosted in the cloud. It provides a NoSQL database to easily manage and manage data in a flexible and scalable way. Firebase Live Database can be used to build applications that respond instantly to changes in data, making these applications more interactive and engaging with users.

- Adding authentication to your Android app with Google Authenticator is quick and easy. Social media, phone, email, password and other authentication methods are supported. With Firebase Identification, you can create secure applications that require user authentication and authorization [7]. The focus of this study is phone number authentication, which is only one type of authentication. It generates an OTP and sends it to the given mobile number. For added protection, the OTP has a renewable lifetime.

- Firebase cloud-based storage makes it easy to store and process large volumes of user-generated content, including photos and videos. With local support and storage, data can be accessed from anywhere in the world.

- FCM (Firebase Cloud Messaging) is an important feature of Google Firebase. Firebase's powerful and scalable cloud service makes it easy to message customers on Android, iOS, and the web. Its options, including objectives, planning, and statistical analysis, make it easy for people to communicate in a timely and relevant manner [2]. It is a popular choice among Android developers using Kotlin because of its ease of use, extensibility, and compatibility with other Google services.

Stripe

With a program called Stripe that processes payments, businesses of all types can now accept and manage payments online. It has many products and services that allow companies to pay quickly and efficiently. Stripe is used by many businesses around the world, including independent business owners, small businesses, and large corporations. Stripe accepts all forms of payment, including debit and credit cards, as well as UPI payments. Card payments include all credit and debit cards, including Visa and MasterCard.
The main features of Stripe are listed below:

1. **Payment Options:** Stripe makes online payments easy and secure. A variety of payment methods are accepted, including debit and credit cards, as well as via mobile devices. Stripe manages every step of the payment process, including fraud detection, chargebacks, and currency conversion.

2. **Security:** Security is a top priority at Stripe and has many tools to help businesses keep their transactions secure. Authorized as PCI Level 1 Service Provider, which is the highest level of validity for a payment processor [10]. In addition to providing the ability to detect and prevent fraud, the Stripe platform also provides real-time risk analysis, machine learning algorithms, and multiple authentication features. It also supports subscriptions and payments, allowing businesses to easily manage long-term payments from customers.

3. **Global Coverage:** Stripe supports more than 40 trading countries, making it easy for businesses to accept payments from customers around the world, with approximately 135 confirmed results. Stripe is solely responsible for handling international payments, including currency conversion and compliance with local laws.

4. **Invoicing:** The software process called invoicing simplifies the membership and reimbursement management of businesses. It has the ability to create pricing models, manage customers and payments, and manage payment refusals. Stripe Billing partners with Hawkeye and Payments as well as other Stripe services to provide billing solutions.

With the many products and services offered by Stripe, businesses can easily manage and accept online payments. With solutions for billing, security, global reach, payment processing, productivity tools, and fraud detection, Stripe provides a complete platform to manage payment in your Android app, as shown in Figure 3.12. Businesses of all types choose it for its flexibility, protection and ease of use.

Fig. 3.12 Dependencies included in this project

Gradle

The open source build automation tool Gradle is used to build Android apps. It is used to automate the development process, including code compilation and packaging, development management, and test execution. Gradle is designed with high flexibility and written in Java and Groovy, so it is suitable for many types of development.

Gradle is a tool for building and managing Android Kotlin application dependencies that can include external libraries, plugins, and other resources. An app build configuration is defined by Gradle using an assembly .gradle file, including settings for the app's package name, version number, and dependencies.

Additionally Gradle has a plugin for building and testing Android apps, which provides a framework for building and testing. These tasks can be accomplished using Android Studio or an integrated development environment (IDE) such as the command line.

Gradle supports dependency management, which is one of the advantages of using it to build Android apps in Kotlin. By teaching them to design gradle files, Gradle allows developers to quickly manage their programs' dependencies. This allows Gradle to download and include required libraries throughout the application development process, simplifying the development process and ensuring that the application has access to the latest versions of external libraries [14].

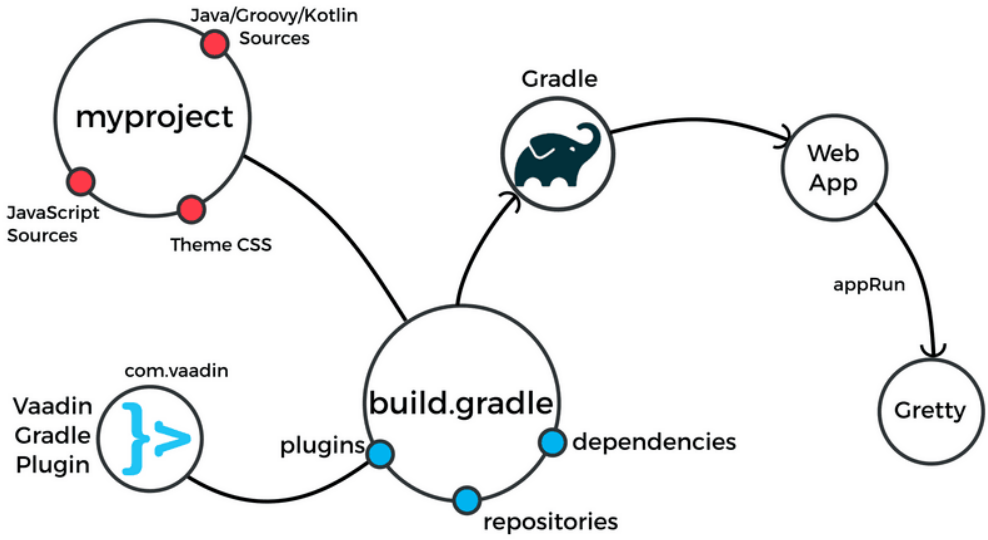


Fig 3.13 Gradle in Android

In summary, Android Kotlin apps are usually built using Gradle, a powerful build automation tool, as shown in Figure 3.13. It is used in the design process, including code collection and packaging, progress control, and test execution [4]. The dependency management provided by Gradle is less complex and suitable for many types of development thanks to the ability to create and add variants and variants. Gradle allows developers to simplify the development process and guarantee that their applications can access the latest versions of external libraries.

Chapter - 4

Experiments and Result Analysis

First we will create a new project in Android Studio. After starting a new project in Android Studio, you can start creating Android Kotlin applications. The first step is to decide on the layout of your application, which will form the basic skeleton of your user interface. Depending on the features and functionality you need to add to your application, you can choose from a variety of configurations, including core functions and empty functions [3]. Using Android Studio's design and editing tools, you can start adding functionality to your app once you've chosen a layout. This will require writing Kotlin code to implement the functionality of the app, as well as creating and modifying UI elements such as buttons, text, and images.

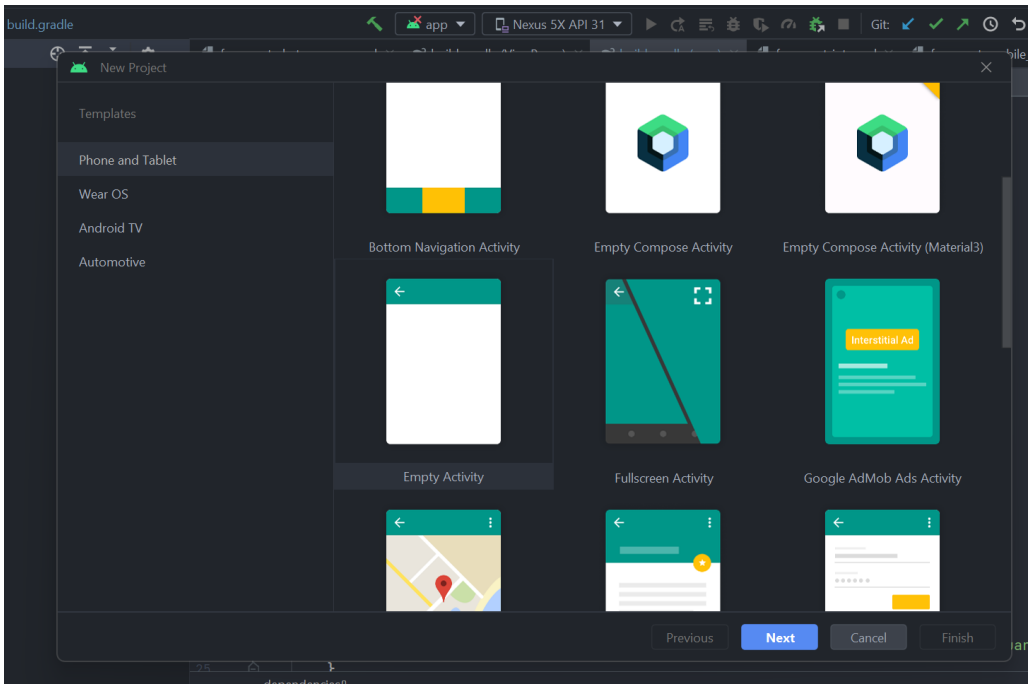


Fig 4.1 Android Studio

Code templates, debugging tools, and integrated emulator are just some of the services Android Studio provides to help you build applications [16]. You can test your app on various devices using the emulator as shown in Figure 4.1.

You can use Android Studio to create an Android Kotlin app that fits your specific needs and make it available to others on the Google Play Store with a little practice and patience. Beginning with our Welcome screen, there are many templates and pages in our work, and often there are many screens or "scenes" that together make up the user program interface. Each screen will have different features and functionality as part of your mobile app. The names of the screens shown in Table 4.1 are given below.

S. No.	Screen Title	Functionality
1.	Welcome Screen	It gives the user an option to register or an existing user to sign-in to their respective accounts.
2.	Login Screen	In this screen the user needs to enter the registered mobile number and press submit and he will move to the next screen.
3.	Register Screen	If the user is not registered (new user), he has to enter a mobile number to get himself registered.
4.	Verification Screen	In this screen there is a user input of 6 digits in which the user needs to enter the OTP which he got via SMS on the entered mobile number.
5.	Home Screen	In this screen the user can press on “Modal” for different functionalities, here he gets the button to set pick up and destination locations.
6.	Task Screen	In this screen the user needs to enter the destination and source locations along with the order that needs to be delivered.
7.	Confirmation Screen	This screen is for rechecking the source and destination location and confirming the address.

8.	Order Details Screen	This screen shows the user the details he has filled in the previous pages along with the delivery charges and proper billing details.
9.	Payment Screen	If the user confirms the details he is then landed on the payment screen where he can add a new card or make payment with already added cards just by filling necessary card details.
10.	Track Order Screen	The user is then able to see the real-time tracking of the delivery partner on a Map.
11.	Feedback Screen	After the item gets delivered the user of the application is then able to rate and write some reviews about the delivery partner.
12.	Edit Profile Screen	This is a feature screen of this application where the user can edit its profile except for his mobile number.
13.	Notification Screen	This screen shows the notification of the items that have been delivered or picked up.
14.	Chat Screen	The user will be able to chat with the admin and communicate about his problems or any other delivery issues.

Table 4.1 List of screens

User Authentication using Firebase

A variety of alternatives are available through Google Firebase for authenticating users and ensuring their security. Because it is simple to install, offers greater security, and enhances the user experience, I have only used mobile authentication in this application[17]. Utilising the mobile OTP in Google Firebase gives mobile applications a secure and practical authentication method that could improve user experience while also reducing the risk of credential fraud. A six-digit OTP is generated, sent to the entered mobile number (if it already exists), and expires after five minutes.

Google Firebase gives us setup content that includes the following fields as shown in Fig. 4.2 after we initialise a new project and select the platform.

1. API Key
2. Storage Bucket
3. Auth Domain
4. Project ID
5. SenderID
6. Application ID

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyCVr12d0Fr5y6ViJRNIn3tWNsTFumQzHII",
  authDomain: "testing-9ec39.firebaseapp.com",
  projectId: "testing-9ec39",
  storageBucket: "testing-9ec39.appspot.com",
  messagingSenderId: "1017245611535",
  appId: "1:1017245611535:web:1aa3b2ed1e32b07f00d57a"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
```

Fig 4.2 Config file generated by firebase

After starting a new project and choosing a platform. The main and most difficult task of is to manage the user's login after logging into their separate account until they exit the application [6]. If the User's Authorization status is not stored in an application, the user is disconnected from the program. To achieve this, the user's credentials must be stored and sent to the application. It generates a one-time password from Firebase to authenticate the user and sends it to the user's phone. Both first-time sign-ups and previous sign-ups go through the same process.

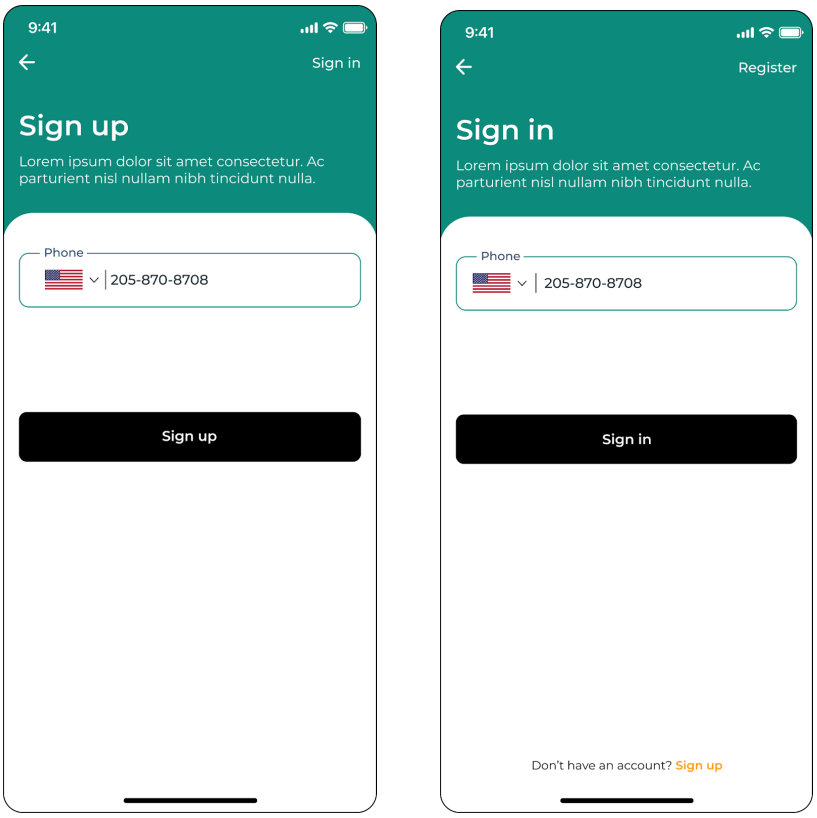


Fig 4.3 Login and Register screens of application

Four different methods can be used to overcome the above challenge: the first is propeller drilling [20], where we send a unique user's Authorization to each mix; the second is to use the Context API and the setting is set as a global variable; the third is to use the Redux Toolkit. I can do this using API content and drilling tools in this app. We can also leverage Android Kotlin's asynchronous storage features to achieve our goal.

It is important to note that AsyncStorage has several limitations, including its inability to handle large files and limited storage capacity. Developers should note that AsyncStorage is not designed as a security solution, but to store sensitive data in a better storage option.

```
import { getAuth, onAuthStateChanged } from "firebase/auth";

const auth = getAuth();
onAuthStateChanged(auth, (user) => {
  if (user) {
    // User is signed in, see docs for a list of available properties
    // https://firebase.google.com/docs/reference/js/firebase.User
    const uid = user.uid;
    // ...
  } else {
    // User is signed out
    // ...
  }
});
```

Fig 4.4 Code for persisting the user

Google Maps

I have integrated Google Maps into my project so that when the user selects a warehouse, Google Maps provides autocomplete to help the user find the warehouse and location more easily [17]. In addition, it is possible to detect the place with signs and to determine its current location. It gives users access to a variety of tools and skills, such as interactive maps, geotagging, directions, and locations. With the help of the Google Maps API in Figure 4.6, users can see the supplier's instructions directly on the Order Tracking screen after payment.

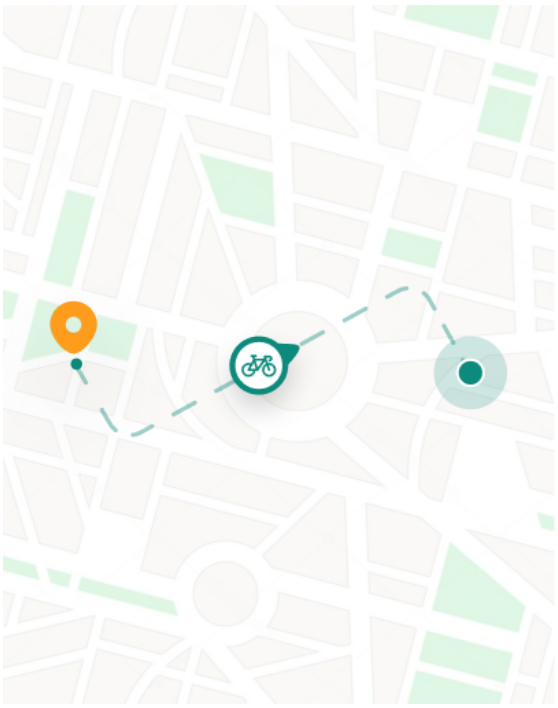


Fig 4.6 Order tracking using MapView

Use Google Maps API in your Android Studio Kotlin application Figure 4.6 as follows:

- Open Google Maps SDK for Android then create an API key and a new project in Google Cloud key Google To enable the Maps API console.
- Include Google Play Services in your distribution: Your app build should now include Google Play Services dependencies. Use the gradle file to access the Google Maps API.
- Google Maps fragments can be included in your application: Use the FragmentManager class to integrate Google Maps fragments into your project's UI.
- Google Maps API Initialization: Use the GoogleMaps class to provide additional settings and initialise the Google Maps API with your API code.
- Customize Google Maps Snippets: Many settings and features, such as markers, polylines, and camera controls, allow you to customize the look and behaviour of your Google Maps widget.

Chat Messaging

A combination of Multi-View Recycler View and Google Firestore, a cloud-based NoSQL database, can be used to implement chat messaging in an Android Kotlin app. Create a RecyclerView adapter, then use multiple RecyclerViews to display the dialog. Multi-View Recycler View is ideal for displaying multiple types of dialog as it can display multiple views in a single RecyclerView. For example, text may appear in one view, while images may appear in another.

Connect the newly designed adapter to the electronic device to receive and display speech immediately [9]. The RecyclerView adapter is updated when new sessions are added using Firestore's snapshot listener, which monitors the collection for changes, as shown in Figure 4.6.

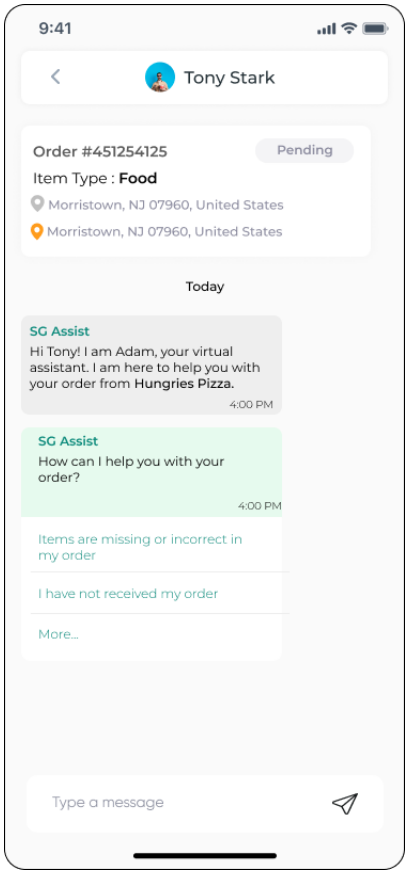


Fig 4.7 Chat Messaging in app

Create text and send buttons to assist in sending messages. Pressing the submit button updates the RecyclerView adapter with new messages as information is added to and added to the Firestore collection as shown in Figure 4.7. can add additional features such as text messages, comments and text icons to enhance the user experience. Using

Multi-View Recycler View and Google Firestore [18], dialogs with real-time updates and UI settings can be added to Android Kotlin applications. However, it is important to ensure that security measures are in place to protect user data and prevent unauthorized access.

Online Payment

Best-in-Class Payments Stripe enables organizations to do business online. When a customer makes an online payment using the Stripe-integrated mobile app or website, several backend transactions are completed. The reverse process of change is shown in Figure 4.8.

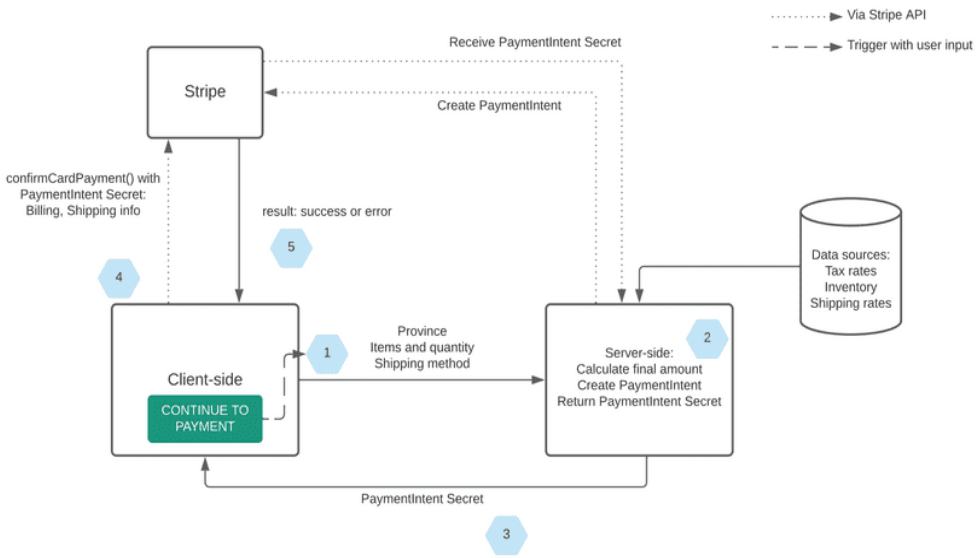


Fig 4.8 Backend of user initiated online payment

I include only payment options in my My Stripe online payments for users to initiate online payments. New users must add a card to the app before starting to pay as shown in Figure 4.8 [3]. You can pay with this card by debit or credit card. The user must enter the following details.

- Card Number ,Card Holder's name, CVV, Expiry Date
- This Information get validated first on the front-end
- If the information is validated by the Stipe.

As shown in Figure 4.9, when the user clicks on the "Add New Card" button, a dialog box will appear showing the card information that the user must provide in order to add a new card for payment.

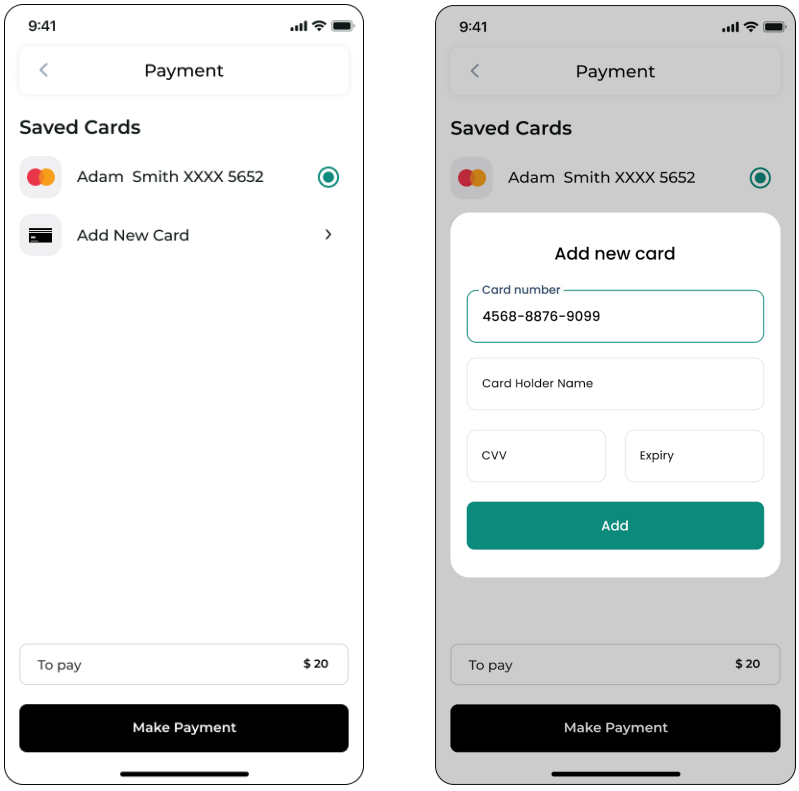


Fig 4.9 Payment Screen of application

After the payment, the user's work with the application is completed [8]. When the truck driver accepts the pickup request, he will receive all the details about the goods to be delivered, including the cargo and the place of delivery.

Chapter - 5
Conclusion

In summary, developing Android applications in Kotlin has many advantages and disadvantages over mobile application development in other programming languages. Kotlin is one of the most used languages for developing Android applications due to its clear and simple concepts, powerful tools, and compatibility with Java.

Also, by leveraging third-party libraries and APIs like Google, developers can easily add features and functionality such as interactive chat and time tracking to their apps, without having to build all the code and power from scratch. Map bunker.

Thanks to the simplicity and ease of Kotlin and the rich tools provided by the Android community, it is possible to create a complete set of Android devices that meet people's current consumption needs [5]. Kotlin provides an easy-to-use and efficient way to build powerful, high-performance applications that can compete with the largest companies in the market, even if you're a beginner developer.

The benefits of using Kotlin go beyond application development. Due to the large developer community supporting Kotlin, there is a wealth of information and resources [13]. It also allows code reuse, which makes it easy to manage and update the app over time.

The use of Kotlin in Android application development will only increase as new challenges and technological advances arise; Therefore, it is the best choice for anyone trying to develop mobile applications in the future.

User data is stored in Google Firebase, which also stores daily reads and writes [2]. Reading and writing numbers are shown in figures 5.1 and 5.2 respectively. The average number of reads and writes per week is also indicated by the dotted line.

Fig 5.1 Reads per day underdeveloped.

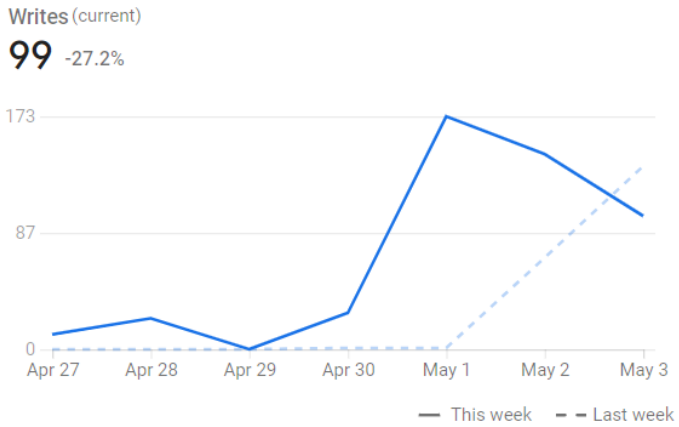


Fig 5.2 Reads per day underdeveloped

First, we will deal with fake data showing how user data is effectively and managed in the background, as shown in Figure 5.3 [19]. Shipping partners are logged in the backend of the app with all their information, including the number of orders they received and how many were successfully shipped.

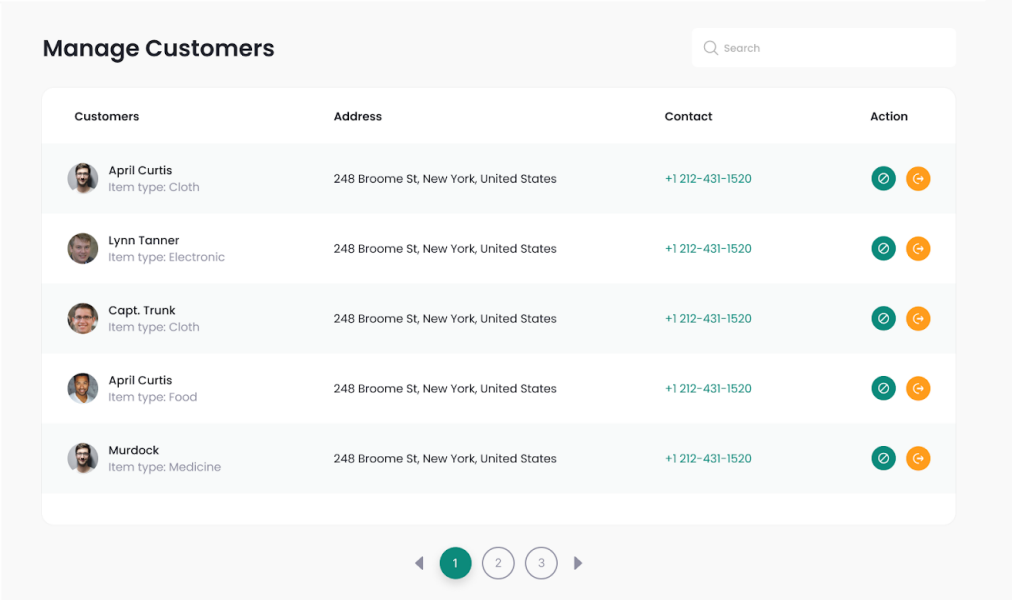


Fig 5.3 Dashboard of admin-side

5.2 Future Scope

Here is the list of features that can be added to our app to improve its effectiveness and user experience:

- **Additional payment options:** Until now, my app only accepts Payment cards (credit/debit). But apart from that, Google Play, Paytm etc. There are other UPI payment methods as well. [7].

- **Support Tutorials:** We can provide animated explanations for the user's first interaction with the application. For new users, this service can be accessed from any Google app.

- **Other login options:** The program currently only allows users to login with their mobile phone number. But in the future, we hope to combine email login with mobile login to give users even greater reach.

- **Integration with blockchain technology:** Another area of potential growth is the integration of blockchain technology. This can provide more security, transparency and responsibility in the delivery process, making it more reliable and trustworthy.

- **Artificial Intelligence and Machine Learning Integration:** Artificial Intelligence and Machine Learning integration allows the Swiggy Ginie clone app to provide users with historical orders, locations, etc.[2] can assist in making recommendations based on Because of this and user experience, customer loyalty will increase.

- **Integration of Augmented Technology:** Integration of augmented reality allows users to see the products by allowing them to see the products before they buy. For example, a user can use the smartphone's camera to see how the furniture will look in the living room before purchasing.

In summary, the Swiggy Ginie clone application developed with Android Kotlin will have a wider use in the future than just delivery. The application can combine technologies such as artificial intelligence, virtual reality, Internet of Things and cloud computing to provide customers with more personalized and efficient services. It can also improve user experience and increase customer loyalty by integrating with payment gateways, chatbots and geofencing [20].

Needless to say, there will be many opportunities for growth and innovation in this space as technology continues to advance.

REFERENCES

[1] Putranto, B.P.D., Saptoto, R., Jakaria, O.C. and Andriyani, W., 2020, December. A Comparative Study of Java and Kotlin for Android Mobile Application Development. In 2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI) (pp. 383-388). IEEE.

[2] Dimitrijević, N., Zdravković, N. and Milićević, V., 2022. An Automated Grading Framework for the Mobile Development programming language Kotlin. International Journal for Quality Research v17, (2), pp.02-01.

[3] Robertus, S., Chandra, J.O. and Andriyani, W., 2020. A Comparative Study of Java and Kotlin for Android Mobile Application Development.

[4] Coppola, R., Ardito, L. and Torchiano, M., 2019, August. Characterizing the transition to kotlin of android apps: a study on f-droid, play store, and github. In Proceedings of the 3rd ACM SIGSOFT International Workshop on App Market Analytics (pp. 8-14).

[5] Sabiyath Fatima, N., Steffy, D., Stella, D. and Nandhini Devi, S., 2020. Enhanced Performance of Android Application Using RecyclerView. In Advanced Computing and Intelligent Engineering: Proceedings of ICACIE 2018, Volume 2 (pp. 189-199). Singapore: Springer Singapore.

[6] Azizah, A., Fitri, I., Fauziah, F. and Hayati, N., 2020. Smart SIAKAD Android based using RecyclerView. SISFOTENIKA, 10(2), pp.192-202.

[7] Lou, T., 2016. A comparison of Android native app architecture MVC, MVP and MVVM. Eindhoven University of Technology.

[8] Daoudi, A., ElBoussaidi, G., Moha, N. and Kpodjedo, S., 2019, April. An exploratory study of MVC-based architectural patterns in Android apps. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (pp. 1711-1720).

[9] Saputra, K., Farhan, K. and Irvanizam, I., 2018, September. Analysis on the comparison of retrofit and volley libraries on android-based mosque application. In 2018 International Conference on Electrical Engineering and Informatics (ICELTICs) (pp. 117-121). IEEE.

[10] Lin, Y., Radoi, C. and Dig, D., 2014, November. Retrofitting concurrency for android applications through refactoring. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (pp. 341-352).

[11] Khawas, C. and Shah, P., 2018. Application of firebase in android app development-a study. International Journal of Computer Applications, 179(46), pp.49-53.

[12] Kong, P., Li, L., Gao, J., Liu, K., Bissyandé, T.F. and Klein, J., 2018. Automated testing of android apps: A systematic literature review. IEEE Transactions on Reliability, 68(1), pp.45-66.

[13] Polese, A., Hassan, S. and Tian, Y., 2022, May. Adoption of third-party libraries in mobile apps: a case study on open-source Android applications. In Proceedings of the 9th IEEE/ACM International Conference on Mobile Software Engineering and Systems (pp. 125-135).

[14] Jinendra, D.R., Bhagyashri, J.R., Pranav, G.Y., Seema, V.U. and Parag, A.N., 2012. Smart travel guide: Application for android mobile. International Journal of Electronics, Communication and Soft Computing Science & Engineering (IJECSCSE), 2, p.115.

[15] Patra, S., Velisetty, K. and Patel, P., 2013. Google Maps and RSS Integration in Android. International Journal of Student Research in Technology & Management, 1(06), pp.614-618.

[16] Ariprasasmita, I.K.W., Piarsa, I.N. and Wibawa, K., 2019. Implementation of Google Maps API and Firebase for Android Based Photographer Marketplace Information System. International Journal of Computer Applications Technology and Research, 8(11), pp.409-414.

[17] Singh, A., Sharma, S. and Singh, S., 2016. Android Application Development using Android Studio and PHP Framework. International Journal of Computer Applications, 975(8887), p.5.

[18] Mallik, R., Hazarika, A.P., Ghosh Dastidar, S., Sing, D. and Bandyopadhyay, R., 2020. Development of an android application for viewing covid-19 containment zones and monitoring violators who are trespassing into it using firebase and geofencing. Transactions of the Indian National Academy of Engineering, 5, pp.163-179.

[19] Kumar, A., 2018. Mastering Firebase for Android Development: Build real-time, scalable, and cloud-enabled Android apps with Firebase. Packt Publishing Ltd.

[20] Rahmi, A., Piarsa, I.N. and Buana, P.W., 2017. FinDoctor-interactive android clinic geographical information system using firebase and google maps API. International Journal of New Technology and Research, 3(7), p.263272.

Report

ORIGINALITY REPORT

4%	3%	2%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Reto Meier, Ian Lake. "Professional Android®", Wiley, 2018 Publication	<1%
2	Submitted to University of Portsmouth Student Paper	<1%
3	repository.nwu.ac.za Internet Source	<1%
4	www.geeksforgeeks.org Internet Source	<1%
5	www.coursehero.com Internet Source	<1%
6	"Proceedings of Third International Conference on Sustainable Expert Systems", Springer Science and Business Media LLC, 2023 Publication	<1%
7	orbilu.uni.lu Internet Source	<1%
8	www.scss.tcd.ie Internet Source	<1%