

National College of Ireland

BSc (Hons) in Computing (Software Development)

BSHCSD4

Academic Year 2020/2021

Name: Muhammad Abu Bakar Sani

Student No: x17112044

Student Email: x17112044@student.ncirl.ie

Gill's POS

Android POS (Point of Sale) Software

Technical Report

Contents

Executive Summary	4
1.0 Introduction	5
1.1. Background	5
1.2. Aims.....	6
1.3. Technology.....	7
1.4. Structure	8
2.0 System.....	9
2.1. Requirements.....	9
2.1.1. Functional Requirements.....	9
2.1.1.1. Use Case Diagram	12
2.1.1.2. Requirement 1 <User Sign up & Sign in>	13
Description & Priority.....	13
Use Case Diagram	13
2.1.1.3. Requirement 2 <Forgot Password >.....	15
Description & Priority.....	15
Use Case Diagram	15
2.1.1.4. Requirement 3 <Select Category>	16
Description & Priority.....	16
Use Case Diagram	17
2.1.1.5. Requirement 4 <Make Sales>	18
Description & Priority.....	18
Use Case Diagram	18
2.1.1.6. Requirement 5 <Scanning Codes>	20
Description & Priority.....	20
Use Case Diagram	20
2.1.1.7. Requirement 6 < Manage Inventory>	22
Description & Priority.....	22
Use Case Diagram	22
2.1.1.8. Requirement 7 < Access Old Invoices>	24
Description & Priority.....	24
Use Case Diagram	24
2.1.1.9. Requirement 8 < Google Maps>	26
Description & Priority.....	26
Use Case Diagram	26

2.1.1.10. Requirement 9 < Chatbot>.....	28
Description & Priority.....	28
Use Case Diagram	28
2.1.1.11. Requirement 10 < Manage Profile>	30
Description & Priority.....	30
Use Case Diagram	30
2.1.1.12. Requirement 11 < Accept Payment>	32
Description & Priority.....	32
Use Case Diagram	32
2.1.1.13. Requirement 12 < Calculator>	34
Description & Priority.....	34
Use Case Diagram	34
2.1.1.14. Requirement 13 < Make a call>	35
Description & Priority.....	35
Use Case Diagram	35
2.1.1.15. Requirement 14 < Set Alarm>	36
Description & Priority.....	36
Use Case Diagram	36
2.1.2. Data Requirements	38
2.1.3. User Requirements	39
2.1.4. Environmental Requirements	41
2.1.5. Usability Requirements.....	41
2.2. Design & Architecture	42
2.3. Implementation	44
Functionality: Splash Screen	44
Functionality: Google sign in.....	45
Functionality: Designing Receipts	46
Functionality: Side Nav Bar	47
Functionality: Card Payment.....	49
Functionality: Profit calculation	50
Functionality: Code scanning	50
Functionality: Chatbot	51
Functionality: Dial Number	52
Functionality: Log-out	52
Functionality: Alarm.....	53

Functionality: Maps	54
Functionality: Profile	55
2.4. Graphical User Interface (GUI)	57
2.5. Testing	66
Unit Testing	66
Integration Testing	68
Automate User Interface Testing	70
2.6. Evaluation	72
3.0 Conclusions	75
4.0 Further Development or Research	76
5.0 References	77
6.0 Appendices	80
6.1. Project Proposal	80
6.1.1. Objectives	80
6.1.2. Background	80
6.1.3. Technical Approach	81
6.1.4. Special Resources Required	82
6.1.5. Project Plan	82
6.1.6. Technical Details	82
6.1.7. Evaluation	83
6.1.8. Bibliography	83
6.2. Project Plan	84
6.3. Reflective Journals	85

Executive Summary

The main purpose of this project is to address the greatly important area of business sales with a view to utilize the technologies which is becoming abundant in our lives to assist those with business sales issues through simple lifestyle intervention. This project is based on implementing a mobile application to help and promote new start-ups and small-scale businesses. As it is the most rising issue that people are not finding any good, cheaper, and competitive Point of sales (POS) System, which will help them in managing their businesses and consequently making their life easier.

Gill's POS System will allow operators with business difficulties to improve their sales and the management of their business with number of approaches: social media login, secure admittance, forgot password, make sales, confirm sales, scanning codes, access previous sales, maps, manage inventories, user profile, AI chatbot, alarm, calculator, dial number (the user will be able to place a call via this application), accepting cash & card payments and competitive receipt design.

The application will be able to assist and improve general status, sales, and reliability also through this purpose by concentrating on key aspects:

- Secure logout
- Help Contact
- Maximize data accuracy.
- Quick response
- Establish secure connection.
- Display info stored in machine readable codes.
- Validate card payments.

1.0 Introduction

1.1. Background

The key goal of this project is to focus on extremely important area of business sales with a view to make use of the technologies which is becoming abundant in our lives to support those with business sales problems through simple lifestyle. This project is based on implementing a mobile application to help and promote new start-ups and small-scale businesses. The idea behind my project is I am familiar with the area as I am working in a retail shop for the last 3 years. So, I think I have a pretty good knowledge about the features a good Point of sales (POS) system should have, how the UI should look like for all age groups to use and how to make it more user friendly and attractive. Besides all this the most important issue is price. These days companies are charging too much for providing small features on POS and, maintenance and the devices also have significant costs. For start-ups or small businesses, it is not easy for them to pay that amount.

Cloud technology is getting popular these days. Cloud helps a person to access and retrieve his data whenever and wherever he wants. The main benefit of this technology is that it speeds up the process. There are still not that many POS software solutions that offer cloud feature. The current available solutions who offer are very expensive, difficult to use, setting up is a complex process for nontechnical person.

Square POS (Square, 2020) is getting very famous as It offers very good and reliable service. It offers 3 different plans. Even If we look at its very basic plan where it says “\$0 Monthly Fee”. Monthly fixed fee is \$0 but on the other side they are charging 2.6% + 10¢ for in person payments and 2.9% + 30¢ for Online payments. Square POS does not offer exchange items and reporting functionality on basic plan as it is the most basic need for all retail businesses. However, Square POS also has very good features that inspires my project e.g.

- User friendly UI designs and background colours.
- Simple and easy to use.
- Does not take long to learn how to use.
- Offers sell online functionality.

The main reason why I am focusing more on putting everything on cloud is that the owner of the shop can check sales, do invoicing, update inventories, etc even while sitting at home. He does not actually need to be in shop to do all this. He can check his employees also without letting them know.

1.2. Aims

The main aim of this project is to build a competent Mobile/Tablet Cloud based Retail sales software. This piece of software will mainly work on android phones and Tablets. My main targeted customers will be small or newly developed business because those people are not able to spend thousands of euros to buy POS software from big companies.

The application will be able to help shopkeepers and enhance their user experience by providing them some of these features:

- Admin Registration & login: It will be cloud based. So, user can register and login to access this app whenever and wherever is needed.
- User can also access the app using Google sign in method.
- User can reset their password.
- Shopkeeper will be able to add new and maintain old inventories. All of this will be stored on cloud based real time database.
- He would be able to make unlimited sales.
- All transactions will have a unique number so that user can track them by using those numbers.
- Competent and understandable Receipt format.
- Receipts will automatically be printed, saved to the internal storage, and uploaded to the cloud on just a single click.
- This application does calculate the profit margin by comparing cost and sales price.
- Provides the option to take payment through cash or card.
- BarCode & QR scanner using camera. This feature is implemented using Machine learning (ML-Kit).
- In app built-in Maps. It would help the user to go and meet new customers and suppliers.
- AI Chatbot to help user by providing desired information.
- User can set alarm as a reminder for different activities.
- User will be able to dial phone number directly from this app.
- This app will also have scientific calculator for quick calculations and price checks.
- Card payments gateway will also be implemented.

I believe this app will help a lot of small and new businesses. In that way they can keep track of their sales and profits and prevent themselves from a loss. It's not mandatory to carry big screens along with them. That app on their phones can perform the same function more quickly and reliably. All calculations, entry and retrieving of information, etc will be to and from real time data base. All this will be done in seconds.

1.3. Technology

In the development of my project, I used several tools & technologies, some of them are Android Studio IDE, JAVA, XML for designing layouts, machine learning & AI (IBM Watson) for chatbot, App access authentication, external storage (Firebase), Google maps, Gimp and Razorpay. Below is the complete detail of all technologies used.

Android Studio: “It is an Android's official IDE. It's purpose built for Android is to hasten the development and support developers to build the top-quality apps for every single Android device i.e., Phones, wearable, etc. It offers custom-tailored tools to Android developers, comprising rich code editing, testing, debugging, and profiling implements.”

(Android, 2021)

Java & XML: Android application uses XML files for designing and specifying permissions, and back end uses java or Kotlin code depends on developer preference.

(Java, 2021)

API: API stands for Application Programming Interface. Android delivers various application framework that permits developer to offer unique resources for different device. I have used Maps API, IBM API, Razorpay API, etc.

(REST API | Google Fit | Google Developers, 2021)

Firebase: It's a scalable, real-time cloud data service. It is devised for developing collaborative and real-time applications. Data in Firebase is in standard JSON format, and developers can retrieve it using the REST API or a client library. Once retrieved through a client library, modifications to data are synchronized in real-time to clients within milliseconds.

Firebase API is a cloud backend as a service which is a tech start-up firm acquired by Google. All Google APIs packaged into one single SDK is offered by Firebase. The services include, authentication, cloud messaging platform, Real-time database, storage, test lab, crash reporting and etc. The application will use most from them.

(Firebase, 2021)

Google Maps: It is the most famous and reliable maps platform. Everyone is familiar with its usage and it is also very cost affective.

(Google Maps Platform | Google Developers, 2021)

IBM Watson: It is AI for business. Watson helps businesses predict further outcomes, automate complex processes, and optimize employees time. Watson Assistant offers customers with consistent, fast, and precise responses across any device, application, or channel. Using AI, Watson Assistant learns from client conversations, enhancing its ability to resolve difficulties the first time while removing the frustration of long wait times, boring searches and unhelpful chatbots.

(How to build a chatbot, 2021), (Watson Assistant - Overview, 2021)

Razorpay: It is the converged payments solution company that allows business to accept, process, and distribute payments via its product suite. With Razorpay, users have access to all payment modes, including credit and debit cards, UPI, and popular mobile wallets.

The most important quality of a payments solution is the ability to scale itself to meet users growing demands. Razorpay devoted to ensuring users have the best payments infrastructure in place.

(Razorpay - Best Payment Gateway for Online Payments , 2021)

Gimp: GIMP is an acronym for GNU Image Manipulation Program. It is a freely distributed program for such tasks as photo retouching, image composition and image authoring. I have used it throughout my project for changing backgrounds, sizes, pixels etc of different images and logos. GIMP is a free and open-source raster graphics editor.

(GIMP - About GIMP, 2021)

1.4. Structure

The structure of this document is as follows, Requirement section which include both functional (including UML diagrams i.e., use cases) and non-functional requirements. Design and Architecture which include architecture diagrams, and details about the system architecture, design, and its main components (System architecture, hardware, and software architecture) then Implementation, GUI, Analysis & Design, Testing, Evaluation, Conclusion and Future development.

2.0 System

2.1. Requirements

2.1.1. Functional Requirements

The Functional requirements of Gill's POS System will be imperative sentence expressing high ranked functional requirements.

- **Splash screen:** The app starts with splash screen. It stays for approx. 3 sec and then move to registration screen. Splash screen contains App name and logo.

- **Authentication related features:**
 - **Registration:** If user is new and does not register before then user needs to fill up registration form or can also register using Google sign in. After registration user will be allowed to use this system.

 - **Login:** After completing the registration process. The system shall let the user to Login & Logout anytime anywhere whenever is needed because it's a cloud-based app. Operator will have to Login typing their valid email and password which will be validated and authenticated on the server side. Most notably is that all the data that are going to the server side will be encrypted. And if user is logging in via Google sign in, they should provide their Gmail email and password.

(Authenticate Using Google Sign-In on Android | Firebase, 2021)

 - **Logout:** System will securely logout the user from both(email and google sign in) but before that it automatically saves and upload all data of the user to the cloud and will fetch it when is required.

 - **Forgot Password:** System will allow the use to reset their password by clicking on Forgot password button. Only registered users will be able to use this feature by providing their valid emails and will get reset password link to their emails.

(Manage Users in Firebase, 2021)

- **Home (Dashboard):** Once the user logged in successfully, the app will bring them to Home screen. There user can select an option as per their need. Available option are Sales, Bar & QR code scanner, Sales history, manage

inventory, Log out, Maps. This screen also shows the user present time and date.

- **Sales:** System shall allow the user to make sales for his customers, in this section. User first need to enter customer name and add the items and its quantity. When user finish the sale of his customer and click 'Save & print' button. The app automatically generates and save receipts on internal storage of a device and upload to the cloud as sales history. User would be able to retrieve all old sales receipts whenever needed.
- **Confirm Sales:** Before taking payment, system shall allow the user to check all items he added into the basket. This feature will help him to make sure that he is not missing anything to till.
- **Accepting payment:** It is the most important feature as it will complete the sales process. This feature will give the user 2 options, rather they click cash or card to complete transaction before moving to the next one. If user selects card, the payment gateway will check card details and process the transaction before showing success screen.
- **Bar & QR code scanner:** Here the system shall allow the user to scan and decode Bar & QR codes and will display the information stored in those codes. Machine learning is using in this instance.
- **Sales history:** System shall allow user to check all sales that they have made so far and also provides the functionality to print specific receipt by putting Id only.
- **Inventory:** This section of the system delivers 2 features. First, user can add new item by typing its name, cost and selling prices, product type and quantity. In return the system automatically calculate profit % and save that item to the database. Second, user can edit the details of any item by just typing the product id only.
- **Google Maps:** It is the most famous and reliable map platform. Everyone is familiar with its usage and also very cost affective. System shall allow the user to enter its start and destination location, then select how he wants to commute there i.e., by car, bus etc. the app will return him with best available route of that time. This feature is very important for this kind of POS system. It would let the use to go and meet to his suppliers or customers. All work-

related actions done under a single app. It is a portable app, so its functionality has to be according to that.

- **Chatbot:** It is the most important requirement of every new software as it increases the usability and makes that software more user friendly. I shall also implement an AI chatbot into this application to give user quick solution of their problem. It is also a quick and easy way to contact help assistance team. I will also try to implement Text-to-speech and speech-to-text in this section.
- **Profile:** Once user have successfully created an account and logged in. The system will let the user to manage their details anytime. It will include username, profile picture, phone number and address.

(Manage Users in Firebase, 2021)

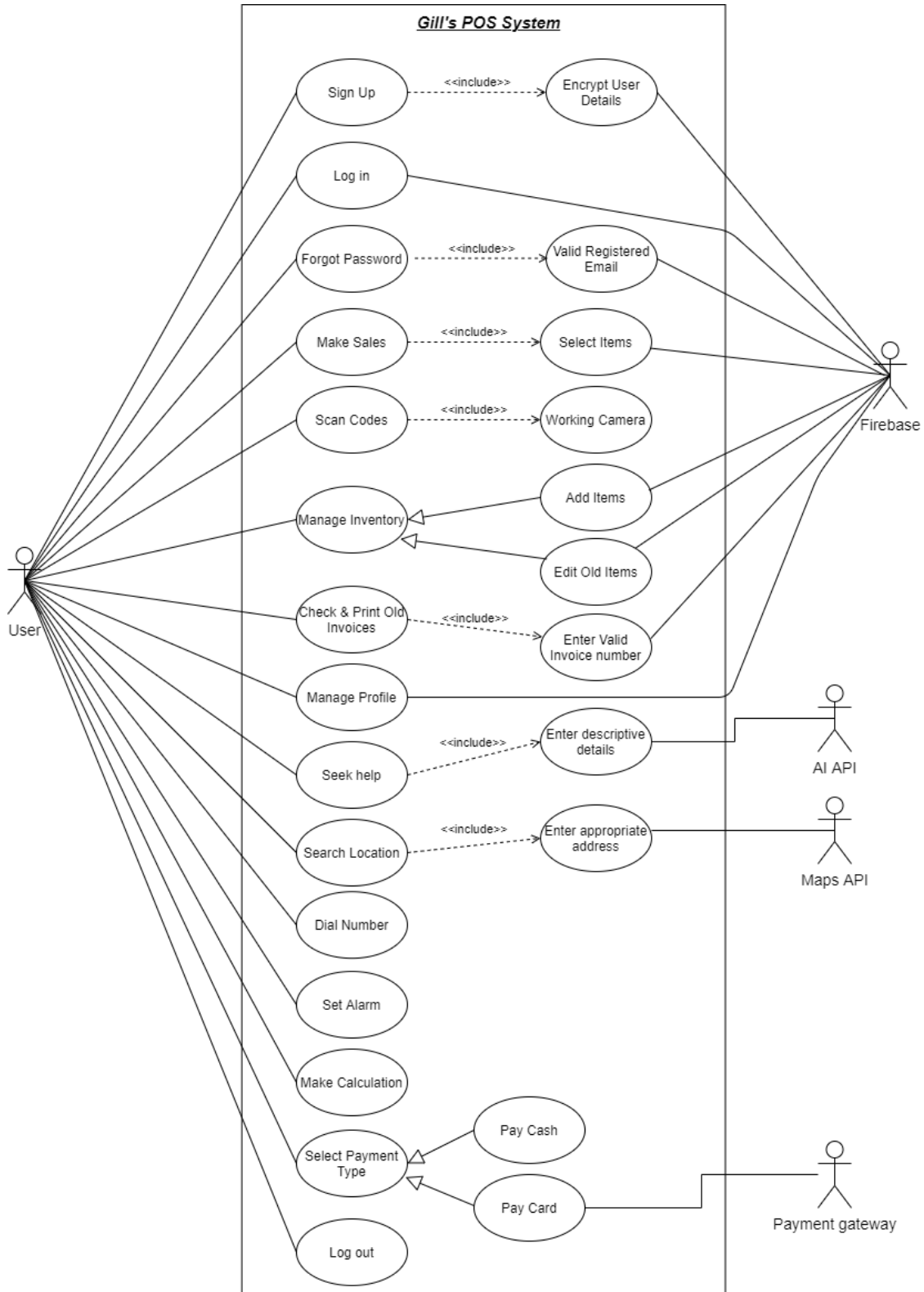
- **Calculator:** As this is a sales app and calculation are the most basic operation of this kind of apps. I will develop inbuilt calculator app for quick calculations like checking profit margins etc. This feature will prevent the user to shift to another app for calculations every time.

(mXparser – Downloads | mXparser – Math Expressions Parser for JAVA Android .NET/MONO/Xamarin – Mathematical Formula Parser / Evaluator Library, 2021)

- **Alarm:** The app will also have alarm functionality. This will let the user to set alarm for various purposes like for wakeup, reminder for start and end of shifts etc.
- **Dial Number:** The app will let the user to dial phone number through this app and make a call to that number when user finished. App will use the phone network to make a call.

2.1.1.1. Use Case Diagram

Figure 1 shows the general use case diagram demonstrating in what way the Gill's POS app will interact with the user and providing an overview of functional requirements.



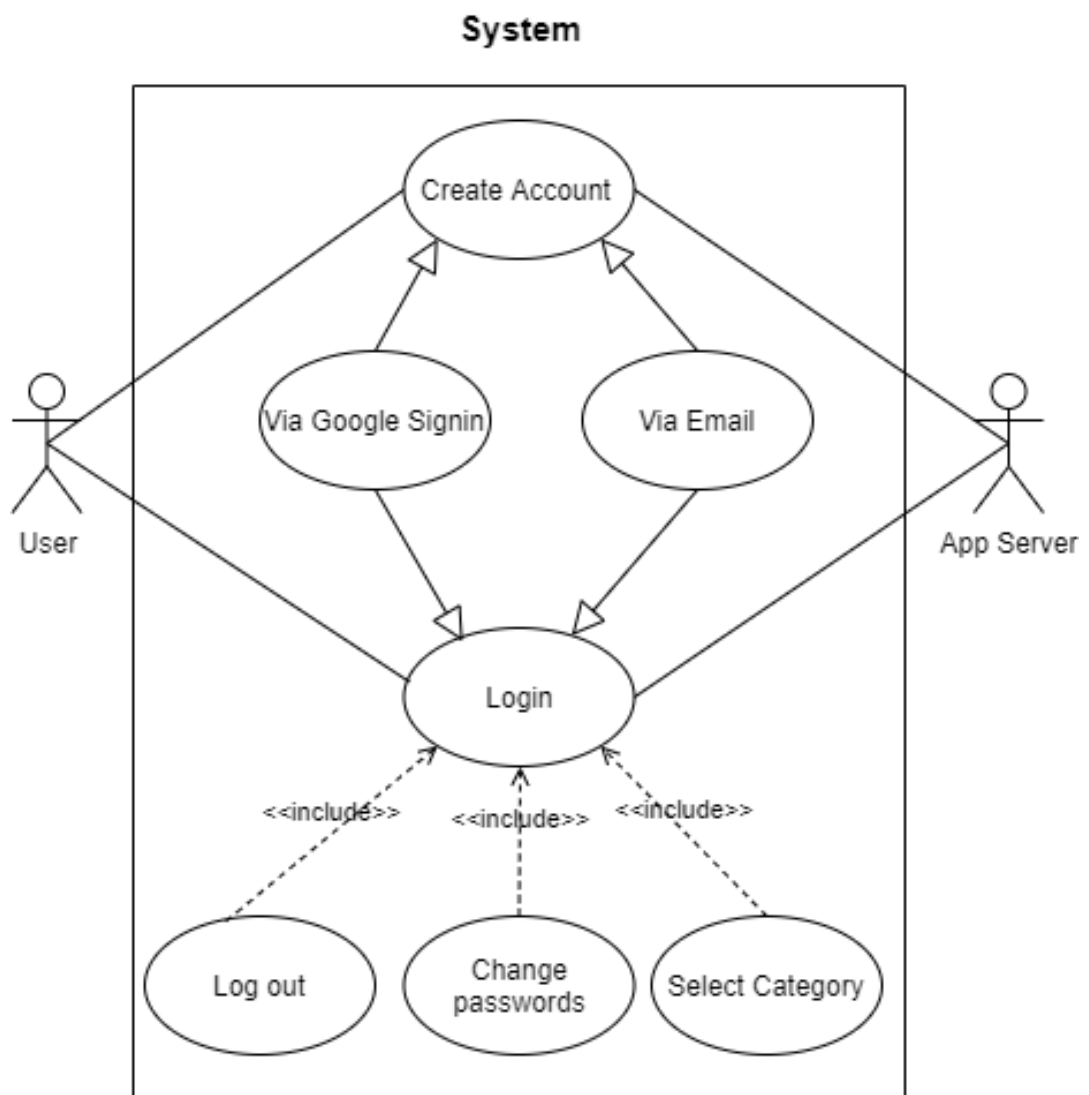
(Figure 1)

2.1.1.2. Requirement 1 <User Sign up & Sign in>

Description & Priority

The system will allow the use to Login or Register to get access to the application. For Registration user must provide his full name, valid email, and a strong password and for Sign in user just need to enter his email and password. User can also access this app by using Google sign in feature. The significance of this use case i.e., Figure 2 is extremely high as the user needs an account to access the application functionalities.

Use Case Diagram



(Figure 2)

Scope

Figure 2 shows the scope of this use case is to permit the user access to the application by signing in or/and register by using user credentials.

Description

This use case describes the logging in and registration.

Flow Description

Precondition

The user must fill all text fields with appropriate information. The fields are: email, username and password.

Activation

The use case starts when user click signup button or click to create an account by using Google sign in.

Main flow

1. The User click on create account button
2. The System shows user options to create an account with Email or social media credentials(Gmail).
3. The User sets connection with his social platform.
4. The System server collects the information to prepare a session with the user
5. The User proceeds to Home screen
6. The User is able to access activities
7. The System encrypts the data provided by the user

Alternate flow

A1: <Registration>

1. The User click on create account button
2. The System displays the registration page.
3. The User fill in needed details.
4. The system confirms the details.
5. The System registers the Users detail to the Database.
6. The System registers the User.
7. The system sends the user to the Home screen.

A3: <Login>

8. The User click on already account button
9. The System displays the login page.
10. The User fill in needed details.
11. The System confirms the details.
12. The System logged in the User.
13. The System sends the user to the Home screen.

A2: <Failed Login>

1. The System cannot find the user credentials stored in the Database.
2. The System displays the error message.
3. The system sends the user back to the registration page.

Exceptional flow

E1 :

1. The user attempts to connect account with Google credentials.
2. The app rejects the credentials provided.
3. The user is displayed with helpful toast to either try again or check internet connection.
4. The app continues at main flow 2.

Termination

The app introduces the user to the home screen to proceed with its activities

Post condition

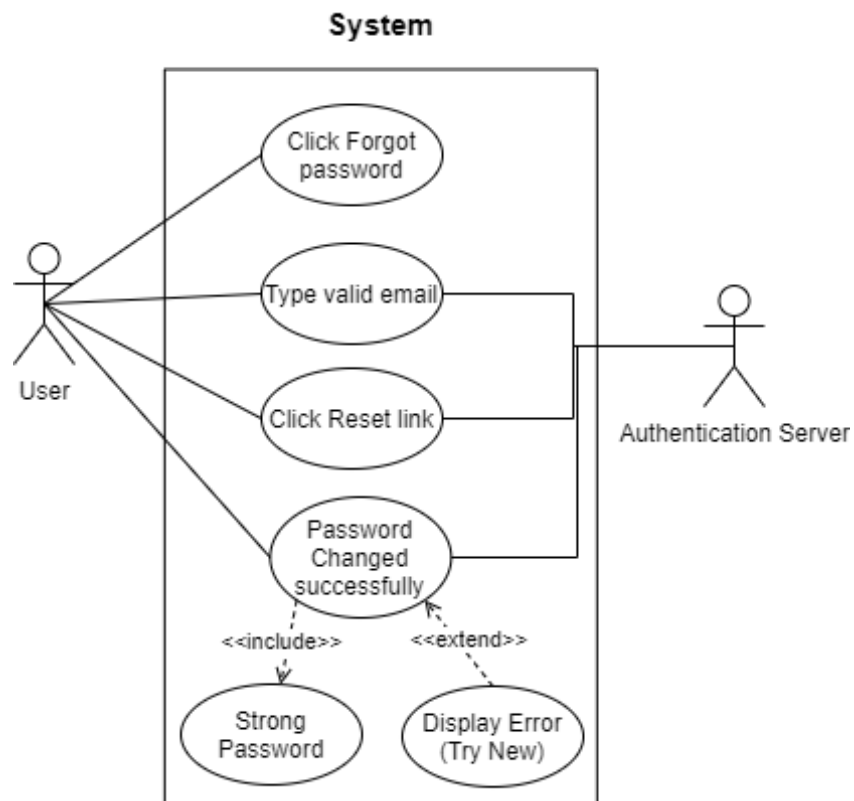
The app awaits user interaction

2.1.1.3. Requirement 2 <Forgot Password >

Description & Priority

The system will allow users to reset their password to get access to the application. A user must enter his valid registered email which he always uses to log-in to the app. The importance of this use case i.e., Figure 3 is high because user as human can forgot passwords and app must provide a feature that enables him to reset it.

Use Case Diagram



(Figure 3)

Scope

Figure 3 shows the scope of this use case is to permit the user access to the application by resetting password.

Description

This use case describes the resetting password.

Flow Description**Precondition**

The System introduces the user to the login screen to select Forgot Password.

Activation

The use case starts when user click Forgot Password button.

Main flow

1. The User click on Forgot Password button
2. The System shows pop-up screen to enable user to type his email.
3. The User enters email and click submit button.
4. The System sends reset link to user email address.
5. The User clicks on that link and type strong password.
6. The System server collects the password and update database.
7. The User will then be able to login using new password

Exceptional flow

E1 :

1. The user typed weak password.
2. The system rejects the password provided.
3. The user is displayed with helpful toast to try again.
4. The app continues at main flow 5.

Termination

The app introduces the user to the success page.

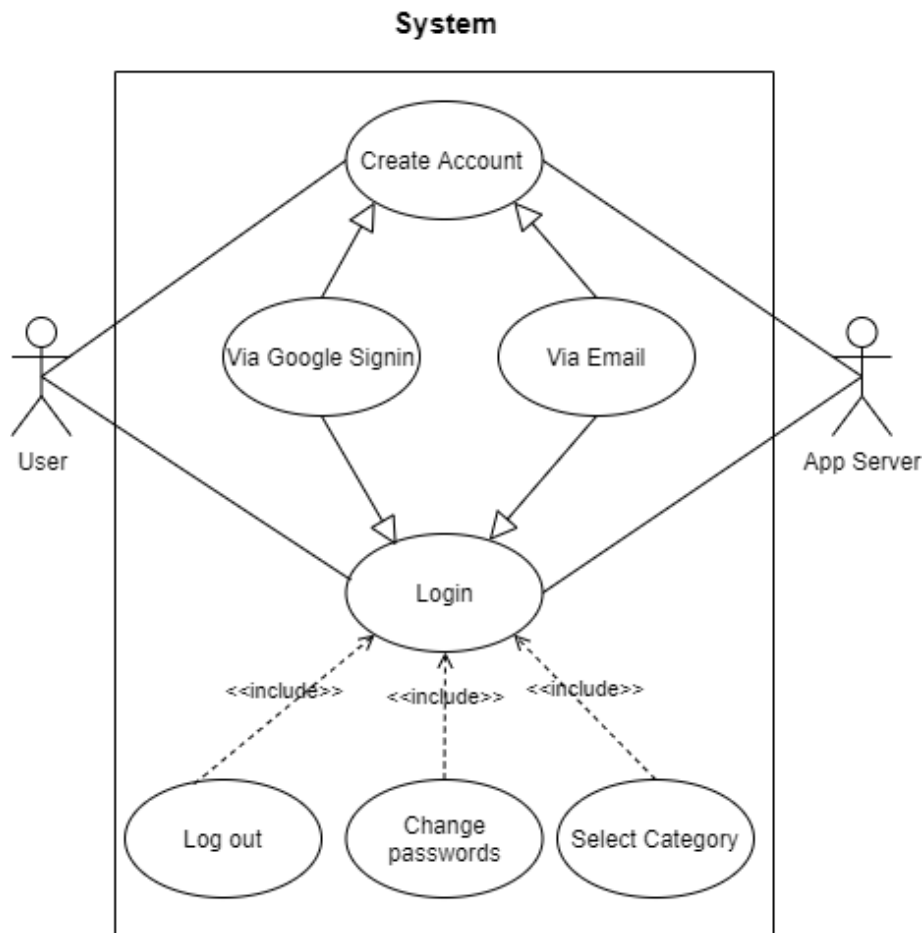
Post condition

The app awaits user interaction.

2.1.1.4. Requirement 3 <Select Category>**Description & Priority**

User must select the specific category they wish i.e., Sales, Scanning, checking history, manage items etc. This functionality allows the System to allow user to select the feature for the specific operation they want.

Use Case Diagram



(Figure 4)

Scope

Figure 4 shows the scope of this use case is to allow User to select Feature they want.

Description

This use case describes the process to select a Feature.

Flow Description

Precondition

The System introduces the user to the home screen to select specific Feature.

Activation

The use case starts when the user moves to Home screen.

Main flow

1. The User selects the feature to use.
2. The System initializes the platform for the user

3. The User attempts to start the chosen platform

Alternate flow

A1 : <Select their category>

1. The User clicks on one of the option: Sales, Scanning, checking history, manage items, logout etc.

Termination

The System permits the user access to click the option category they want

Post condition

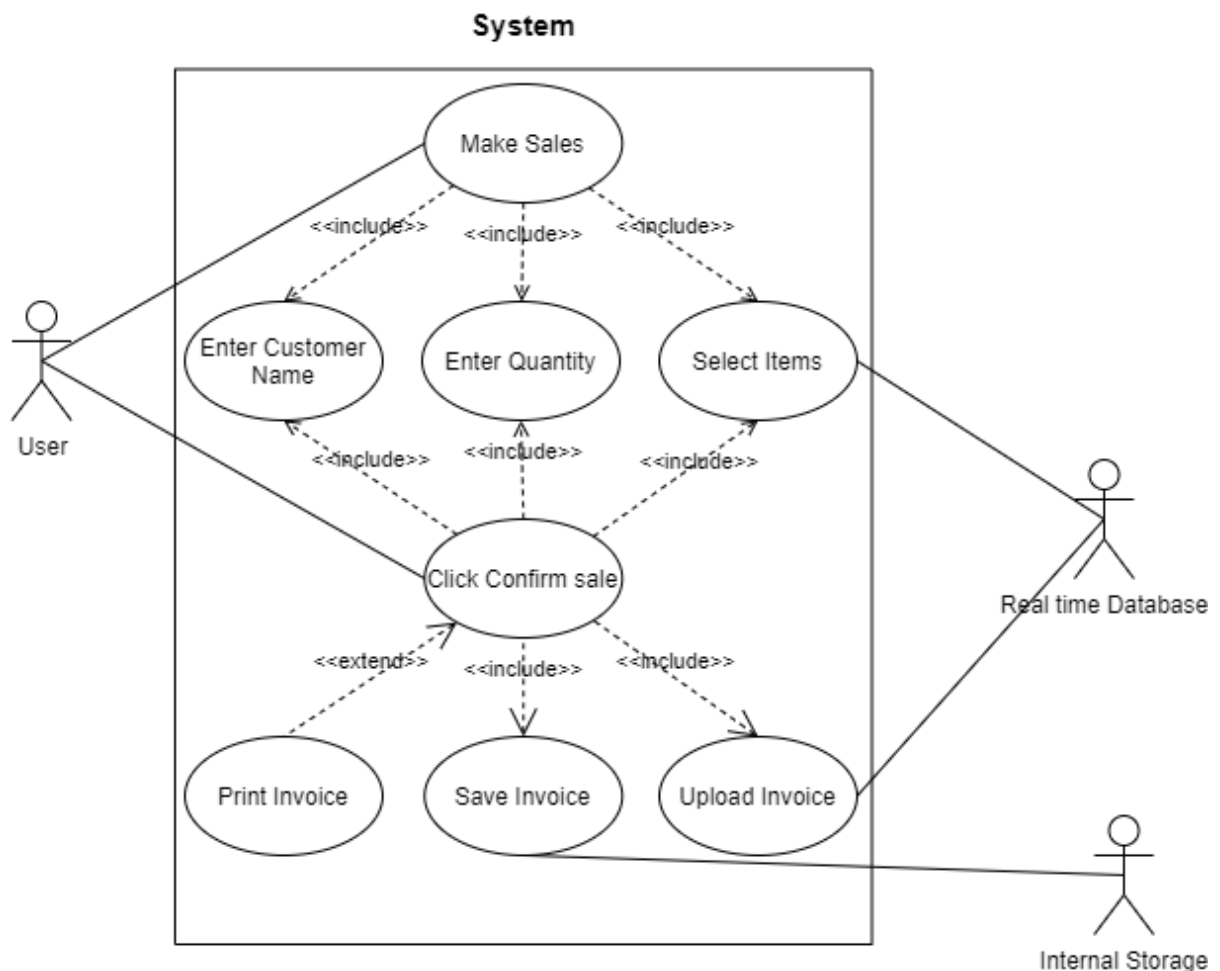
The system will take the User to the category they want.

2.1.1.5. Requirement 4 <Make Sales>

Description & Priority

User must click on Sales icon present in the home page. This allows the system to setup the platform to make sales. This is the most important requirement of the whole system. This feature is affiliated with other requirements also.

Use Case Diagram



Scope

Figure 5 shows the scope of this use case is to allow user to make sales by selecting items and entering quantity.

Description

This use case describes the flow of making sales feature.

Flow Description

Precondition

An android device must be connected to an internet and user must be logged in, In an appropriate manner.

Activation

This use case starts when user click on sales icon

Main flow

1. The User selects the Sales feature to use.
2. The System initializes the platform for the user.
3. The user selects 'Make Sales'
4. The system brings the user to the main screen of make sales.
5. The user fill attributes with appropriate values and options, and then confirm sales.
6. The system collects all the information and generate receipts. That receipts will then upload to the cloud and save to internal storage of device also.
7. The user can access and print it off whenever is required.

Termination

The system saves and upload the transaction automatically when sale is done.

Post condition

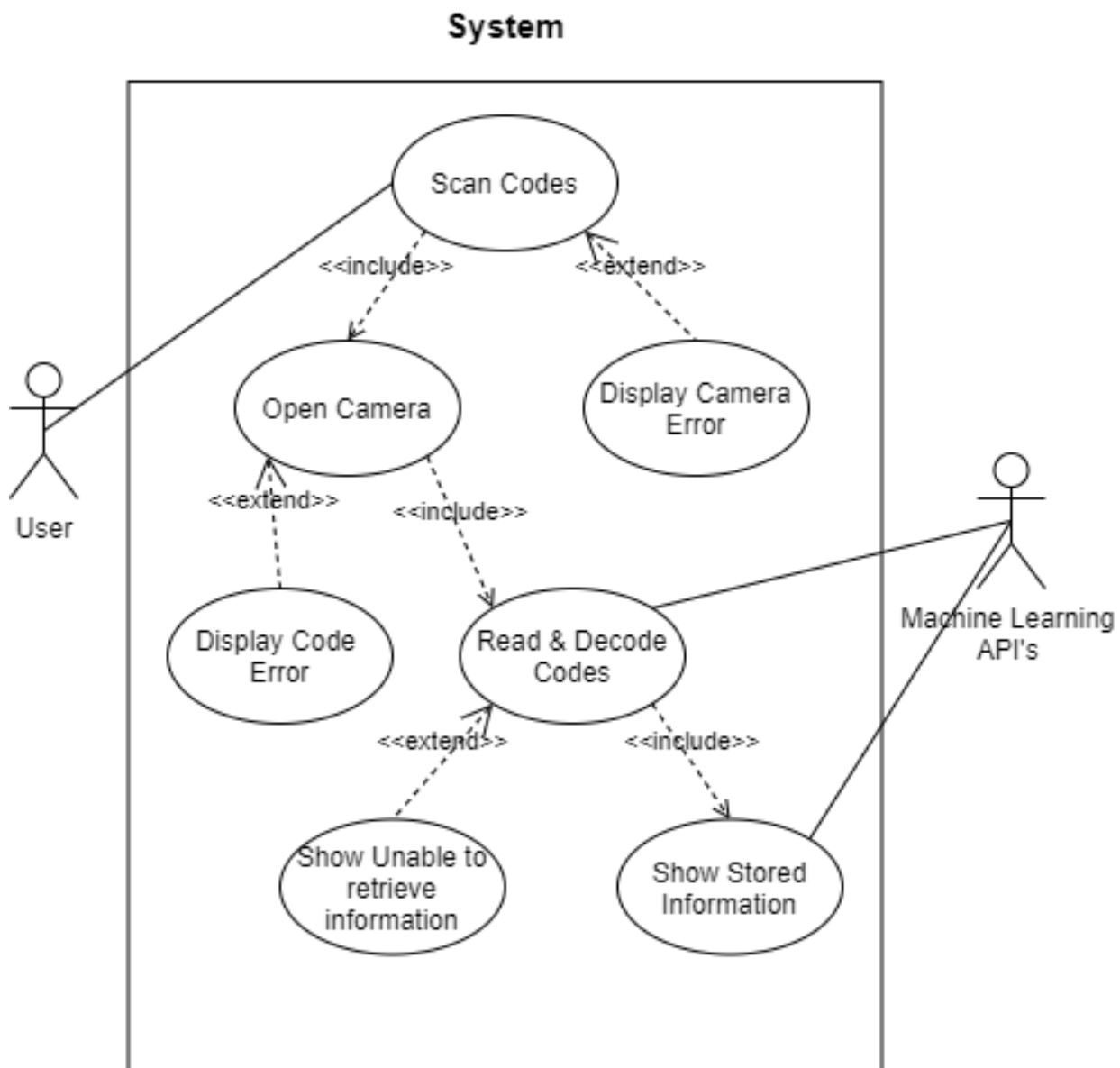
The system goes back to main sales screen.

2.1.1.6. Requirement 5 <Scanning Codes>

Description & Priority

When user selects Scan code feature. This allows the system to setup the platform for scanning codes. This feature is also essential as its implementing machine learning and reading technology.

Use Case Diagram



(Figure 6)

Scope

Figure 6 shows the scope of this use case is to scan Bar & QR codes and extract electronically stored information.

Description

This use case describes the procedure how user can scan codes in an accurate way. And how this decoding thing works.

Flow Description

Precondition

Device should be connected to an internet and user must be logged in. The System presents the user to the home screen to select this feature.

Activation

This use case starts when user click on Scanning icon.

Main flow

1. The User selects the Scanning feature to use.
2. The System initializes the platform for the user.
3. The user selects 'Scanning codes'.
4. The system opens camera and starts scanning.
5. The user attempts to point the camera towards Code.
6. The system reads the code and extract the info stored in it.
7. The user should be able to see that information.
8. The user can repeat the process multiple times

Termination

The system presents the decoded and extracted information.

Post condition

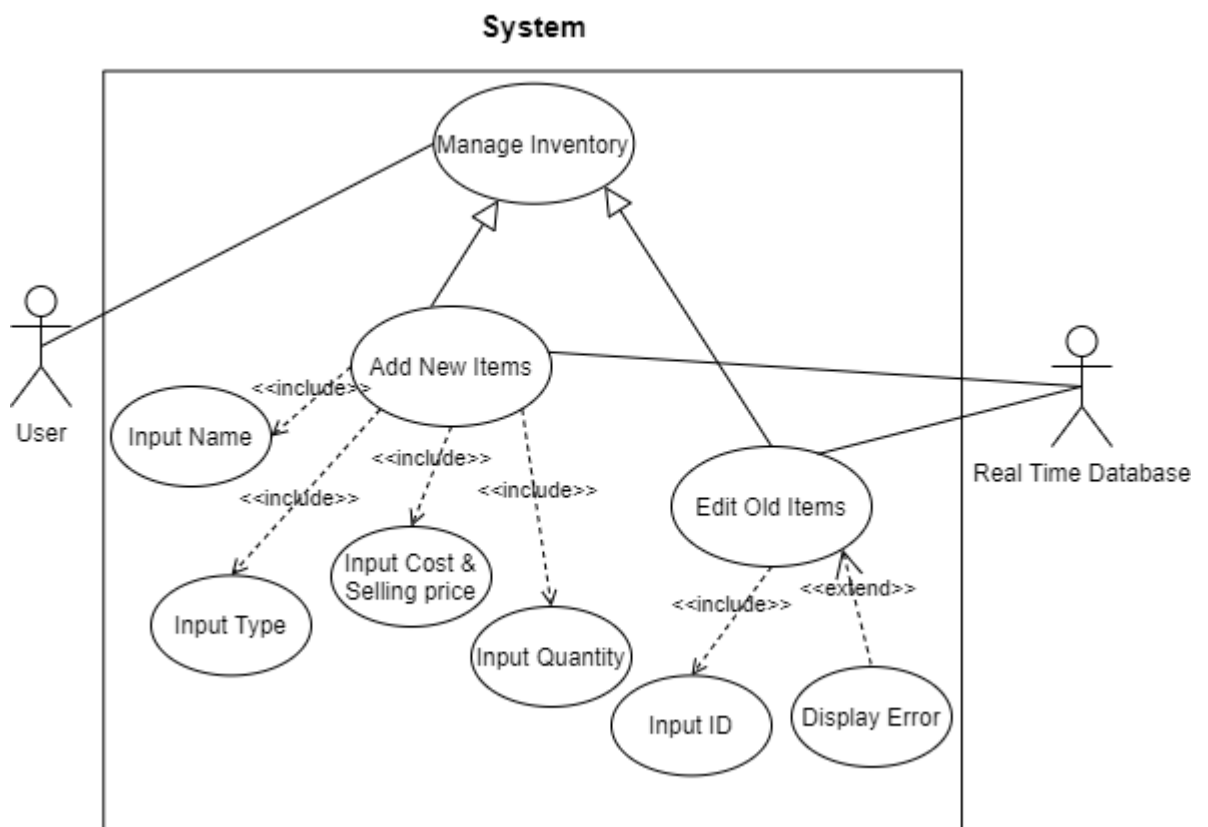
The system scans next code.

2.1.1.7. Requirement 6 < Manage Inventory >

Description & Priority

When user select Manage Inventory option. The system is allowed to setup the platform for adding new and editing old items in the database. This feature has the highest priority, as it is the core part of the app. Because without this feature user cannot drive the sales.

Use Case Diagram



(Figure 7)

Scope

Figure 7 shows the scope of this use case is to add and maintain items to the database. All this is done on real time bases.

Description

This use case describes the flow procedure of Manage Inventory when user selects this. The system will present platform for both Add & Edit items.

Flow Description

Precondition

Device should be connected to internet and user must be logged in. The System presents the user to the home screen.

Activation

This use case starts when user click on 'Inventory' icon from home screen.

Main flow

1. The User selects the Inventory feature to use.
2. The System initializes the platform for the user.
3. The user has two options.
4. The user clicks Add new<A1>
5. The user clicks Edit old<A2>
6. The system initiates that process.

Alternate flow

A1 : <User click Add> <E1>

1. The user clicks Add new.
2. The system checks if the user fills all the text fields with necessary information.
3. The system will push all that info to cloud-based database

A2 : <User click Edit Old Items> <E2>

1. The user clicks Edit old and enter item Id.
2. The system will fetch all details of this product from the cloud and display it.
3. The user then edits it as per the requirement.
4. The system will push all that changes to cloud-based database to the same Id.

Exceptional flow

E1 : < User click Add >

1. User attempts to upload the content without internet connection.
2. System displays helpful toast that the user is not connected need Internet connection.
3. User initialises Wi-Fi or Mobile data connection
4. System brings user to main flow 4.

E2 : < User click Edit Old Items >

1. User attempts to upload the content without internet connection.
2. System displays helpful toast that the user is not connected need Internet connection.
3. User initialises Wi-Fi or Mobile data connection
4. System brings user to main flow 4.

Termination

The Use case is terminated when system successfully upload the content.

Post condition

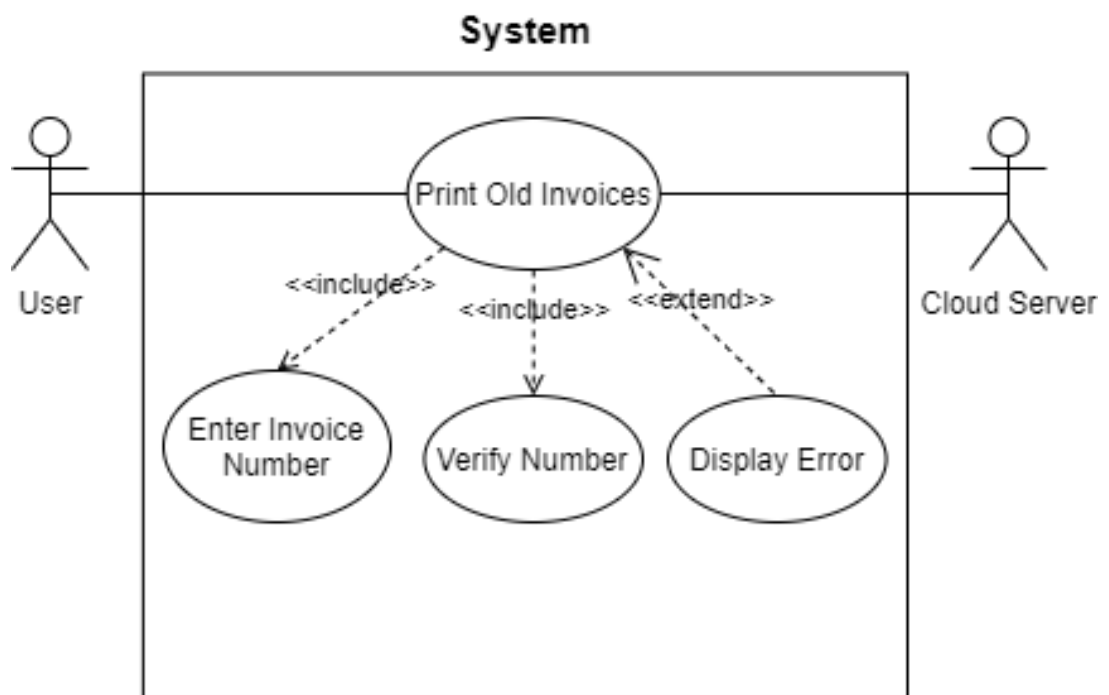
The system ready to take next job.

2.1.1.8. Requirement 7 < Access Old Invoices >

Description & Priority

When user select Old Invoices option. This allows the system to setup the platform for Access Old Invoices. This feature also has great priority, as it helps the user to go back check all the sales he has made so far. And can also access any specific one by entering Invoice No.

Use Case Diagram



(Figure 8)

Scope

Figure 8 shows the scope of this use case is to check sales, mistakes made or to print copies of any old invoice

Description

This use case describes the procedure when user select 'Old invoices' from the Home screen. The system will present platform for it.

Flow Description

Precondition

Device should be connected to internet and user must be logged in. The System presents the user to the home screen.

Activation

The use case starts when user enter invoice number.

Main flow

1. The user selects 'Access Old Invoices'.
2. The System initializes the platform for the user.
3. The system presents highlights of all the invoices
4. The user enters valid invoice number and hit print button.<E1>
5. The system retrieves that invoice from the cloud and print it.

Exceptional flow

E1 : <Invalid invoice id>

1. The user enters wrong invoice id.
2. System displays helpful toast that the user has entered wrong invoice id.
3. The system brings the user to main flow 4.

Termination

The Use case is terminated when the User is giving valid id to the System to print it out.

Post condition

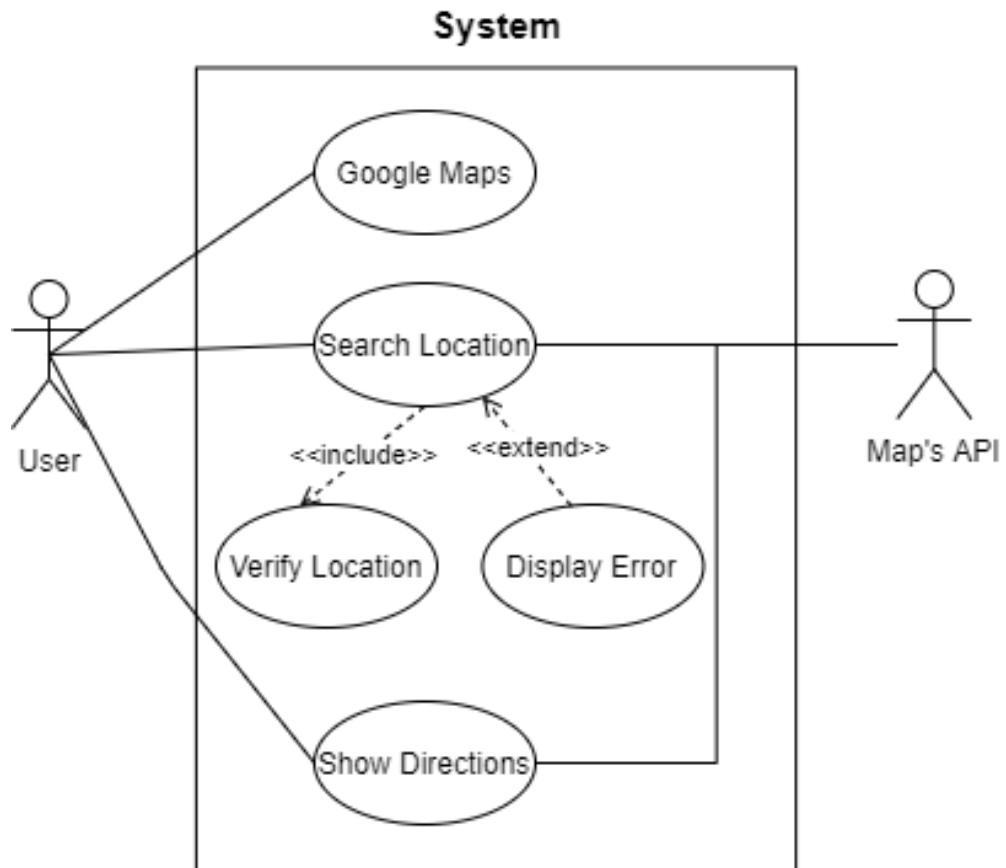
The system waits until User interact with it.

2.1.1.9. Requirement 8 < Google Maps>

Description & Priority

When user select Map's option. The system will open map activity. There user will be able to search for any location by providing the specific address or postcode of that location. This feature also has great priority, as it helps the user to move from 1 location to another, easily.

Use Case Diagram



(Figure 9)

Scope

Figure 9 shows the scope of this use case is to enable user to search for different locations.

Description

This use case describes the procedure when user select 'Maps' from the Home screen. The system will present platform for it.

Flow Description

Precondition

Device should be connected to internet and user must be logged in. The System presents the user to the home screen.

Activation

The use case starts when user click on 'Maps' icon from home screen.

Main flow

1. The user selects the search field.
2. The System initializes the platform for the user.
3. The user types the location and click search.<E1>
4. The system points out the user desired location.

Exceptional flow

E1 : <Invalid location>

1. The user enters wrong location.
2. System displays helpful toast that user has entered wrong location and cannot be found.
3. The system brings the user to main flow 3.

Termination

The Use case is terminated when the User is given valid address to the System.

Post condition

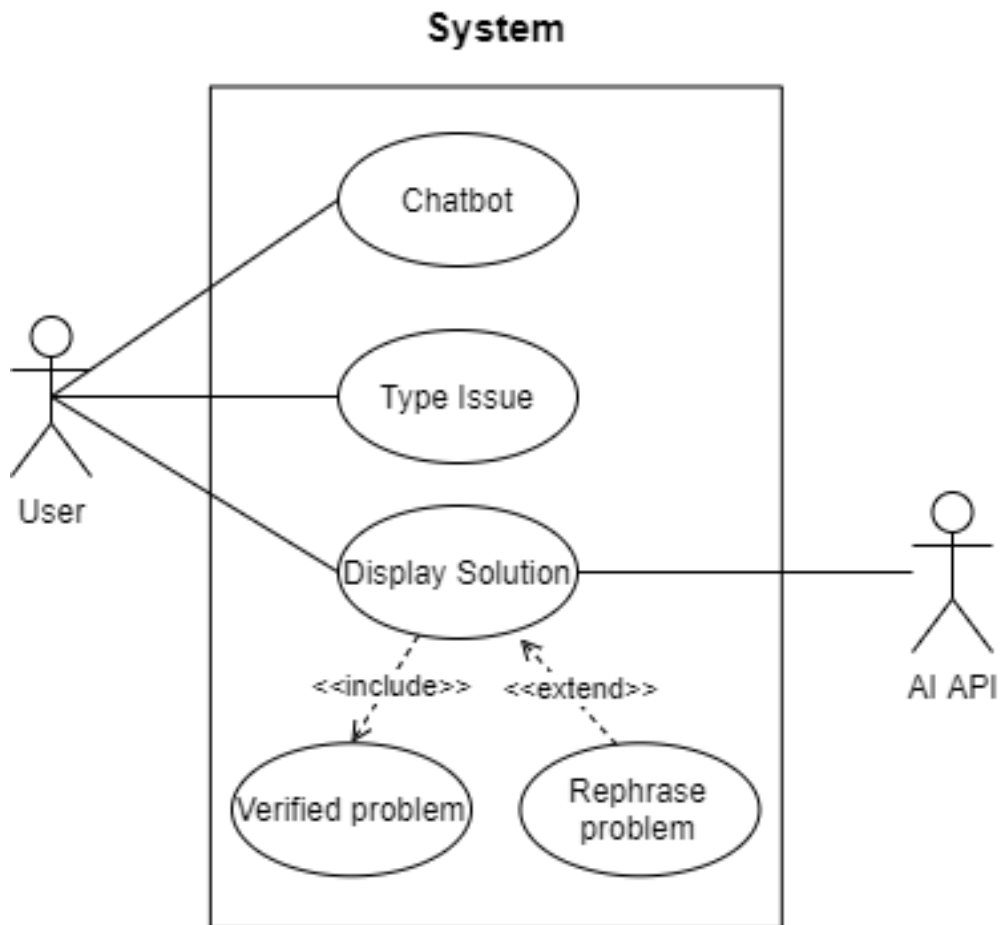
The system waits until User interact with it.

2.1.1.10. Requirement 9 < Chatbot >

Description & Priority

When user clicks on chatbot form side nav bar. The system will open chatbot activity. There user will be able to search for any specific help by providing details of any particular problem. This feature has a great importance, as it drives user out from any issue.

Use Case Diagram



(Figure 10)

Scope

Figure 10 shows the scope of this use case is to enable user to search for verified problems.

Description

This use case describes the procedure when user select 'Chatbot' from the nav bar. The system will present platform for it.

Flow Description

Precondition

Device should be connected to internet and user must be logged in. The System provide nav bar access to the user.

Activation

The use case starts when user click on 'Chatbot' from the nav bar.

Main flow

1. The user selects the search field.
2. The System initializes the platform for the user.
3. The user types the problem and click search.<E1>
4. The system gives brief solution of that problem.
5. The system should speak when user clicks on any text.

Alternate flow

A1 : <Speech-to-text >

1. The user speaks about the problem after clicking on mic's icon.
2. The system hears it and populate that problem on search field.
3. The user click search if he is satisfied.
4. The system brings the user to main flow 4.

Exceptional flow

E1 : <Problem not understanding>

1. The user enters something that system is not able to understand.
2. System displays helpful message that "Unable to understand, please try to rephrase it"
3. The system brings the user to main flow 3.

Termination

The Use case is terminated when User type problem that system understands.

Post condition

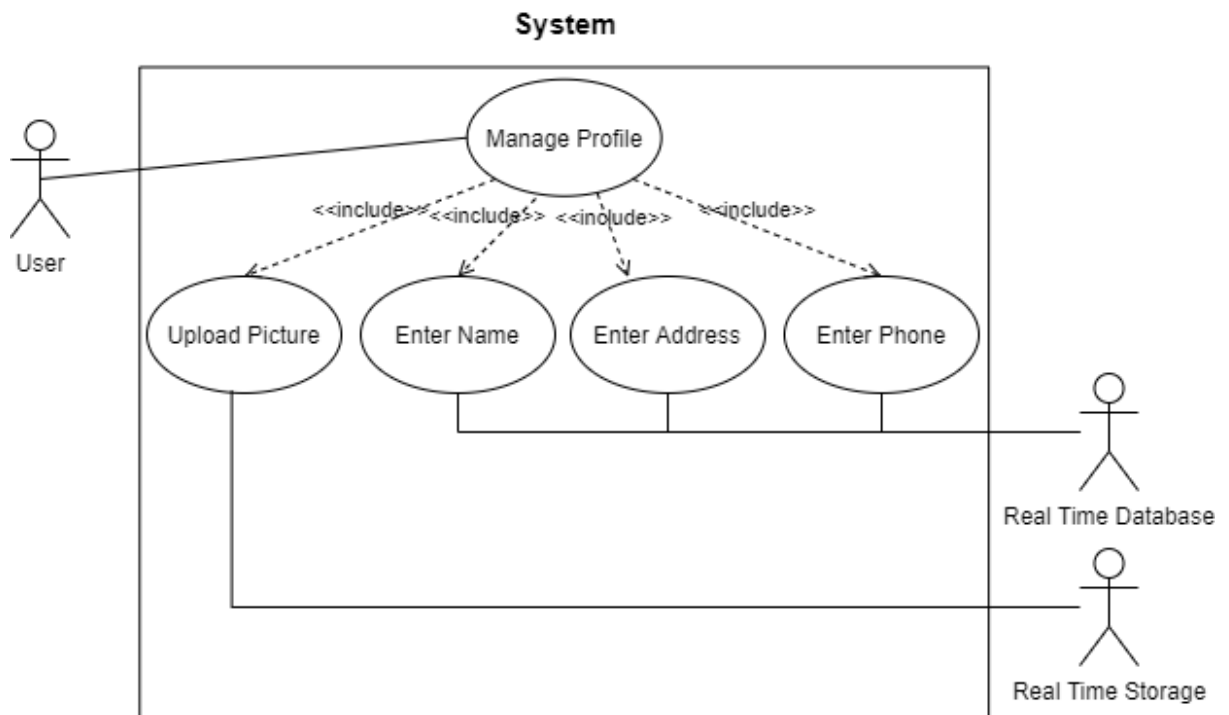
The system ready to take next job.

2.1.1.11. Requirement 10 < Manage Profile>

Description & Priority

When user clicks on Profile present in nav bar. The system will open profile activity. There user should be able to add or edit their personal details any time. User would be able to upload their picture as profile pic, enter their name, address, and phone number.

Use Case Diagram



(Figure 11)

Scope

Figure 11 shows the scope of this use case is to allow user to add or update their personal details.

Description

This use case describes the procedure of managing profile when user selects 'Profile' from the nav bar.

Flow Description

Precondition

Device should be connected to internet and user must be logged in. The System provide nav bar access to the user.

Activation

The use case starts when user click on 'Profile' from the nav bar.

Main flow

1. The user selects their image for profile.
2. The System displays it in preview section.
3. The user enters their name, address, and phone number.
4. The user click finish.
5. The system uploads everything to cloud.

Exceptional flow

E1 : <Problem on reading image>

1. The user selects image that system is not able to read due to non-compatible format.
2. System displays helpful message i.e., "Unable to read file"
3. The system brings the user to main flow 1.

Termination

The Use case is terminated when User updates their profile successfully.

Post condition

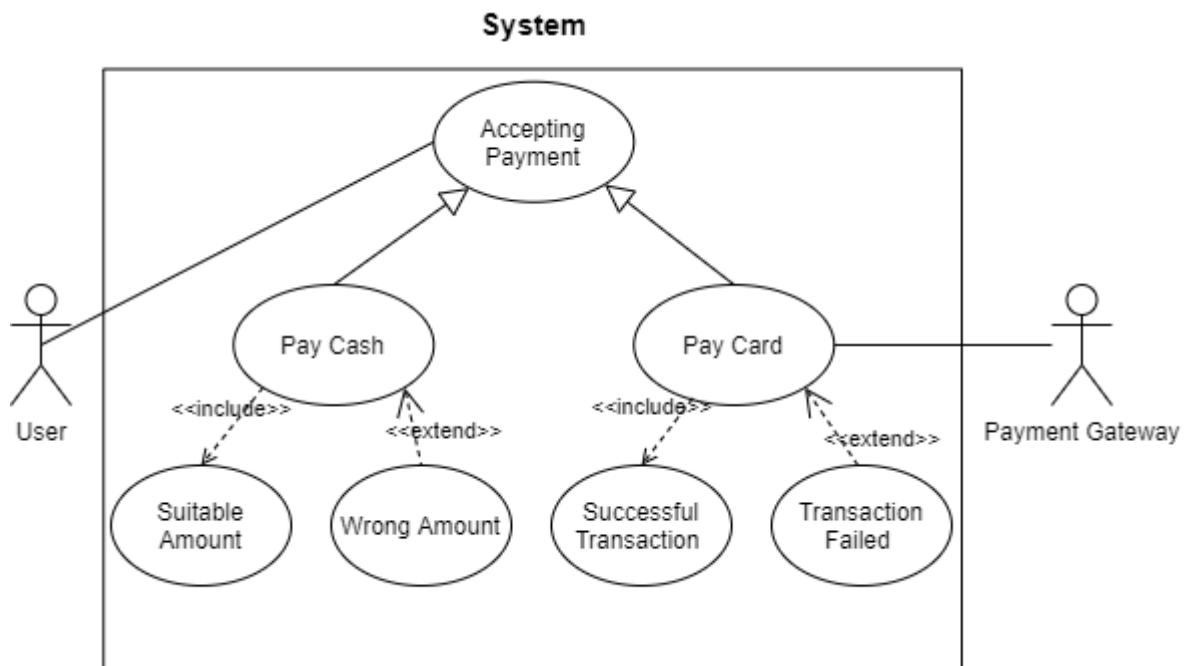
The system will bring the user to success screen.

2.1.1.12. Requirement 11 < Accept Payment>

Description & Priority

When user finish making sales, the pop-up screen will appear. This screen will give user 2 options i.e., Pay Card or Pay Cash. The system will let the user to interact with any of these 2 options. Payment gateway will be involved if user selects Card.

Use Case Diagram



(Figure 12)

Scope

Figure 12 shows the scope of this use case is to allow user to take payment either by cash or card.

Description

This use case describes the procedure of accepting payment from client after finalising the sales order.

Flow Description

Precondition

Device should be connected to internet and user must be logged in. User should have created any sales orders.

Activation

This use case starts when user click on 'Finish' icon on sales activity.

Main flow

1. The System initializes the platform for the user.
2. The user has two options.
3. The user clicks Pay Cash<A1>
4. The user clicks Pay Card<A2>
5. The system initiates that process.

Alternate flow

A1 : <User click Pay Cash > <E1>

1. The user clicks on Cash.
2. The system will bring the user to cash screen and give him some time to count and check cash.
3. The system will print receipt for that transaction.

A2 : < User click Pay Card > <E2>

1. The user clicks on Card.
2. The system will bring the user to gateway screen.
3. The user will then enter card details.
4. The gateway system will check details and try to process the transaction.
5. The system will print receipt if transaction becomes successful.

Exceptional flow

E1 : < User click Pay Cash >

1. User identifies some cash issue.
2. System recalculates the bill.
3. User reinitialises the process.
4. System brings user to main flow 2.

E2 : < User click Pay Card >

1. User waiting for transaction process to complete successfully.
2. System displays decline screen due to wrong card details or insufficient funds.
3. User reinitialises the process.
4. System brings user to main flow 2.

Termination

The Use case is terminated when system successfully accept the payment.

Post condition

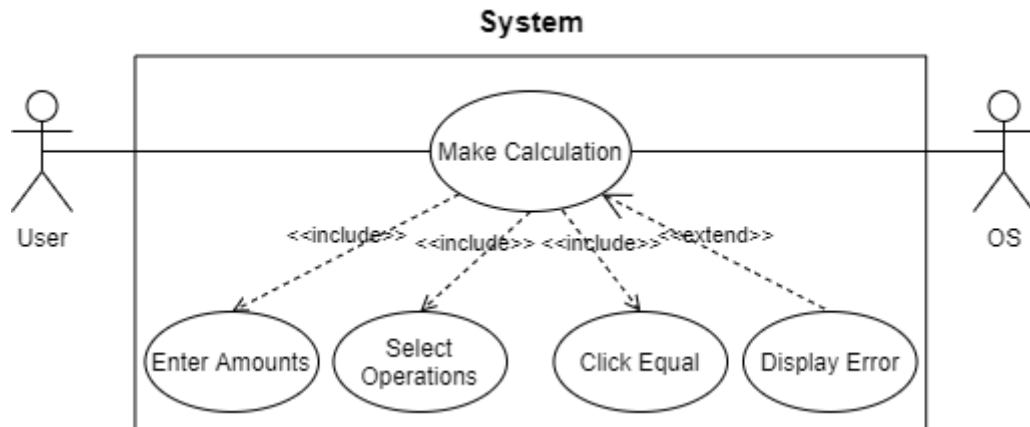
The system is ready to take next job.

2.1.1.13. Requirement 12 < Calculator >

Description & Priority

When user clicks on Calculator present in nav bar. The system will open calculator activity. There user should be able to perform some basic business calculations.

Use Case Diagram



(Figure 13)

Scope

Figure 13 shows the scope of this use case is to enable user to perform some basic calculations.

Description

This use case describes the procedure of performing calculations when user selects 'Calculator' from the nav bar.

Flow Description

Precondition

Device should be connected to internet and user must be logged in. The System provide nav bar access to the user.

Activation

The use case starts when user click on 'Calculator' from the nav bar.

Main flow

1. The user enters the amount with the mathematical operations and click equals.
2. The System performs calculation and display the output.

Exceptional flow

E1 : <Syntax Error>

1. The user enters details in wrong syntax.
2. System displays helpful message i.e., “Unable to perform calculation”
3. The system brings the user to main flow 1.

Termination

The Use case is terminated when User enters details in current syntax .

Post condition

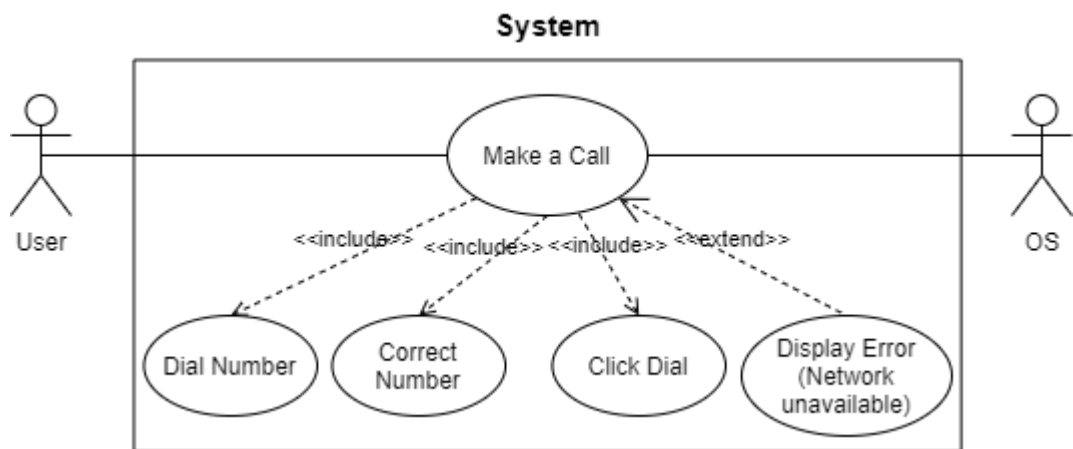
The system is ready to take next job.

2.1.1.14. Requirement 13 < Make a call>

Description & Priority

When user clicks on Dial Number present in nav bar. The system will open this activity. There user should be able to make calls.

Use Case Diagram



(Figure 14)

Scope

Figure 14 shows the scope of this use case is to enable user to make calls using device network.

Description

This use case describes the procedure of making a call when user selects ‘Dial Number’ from the nav bar.

Flow Description

Precondition

Device should be connected to internet & sim network, and user must be logged in. The System provide nav bar access to the user.

Activation

The use case starts when user click on 'Dial Number' from the nav bar.

Main flow

1. The user enters the number where he wants to make a call and click dial.
2. The System starts making a call on this number.

Exceptional flow

E1 : <Network unavailable>

1. The user enters number, but network is unavailable.
2. System displays helpful message i.e., "Unable to make a call"
3. The system brings the user to main flow 1.

Termination

The Use case is terminated when User successfully make a call.

Post condition

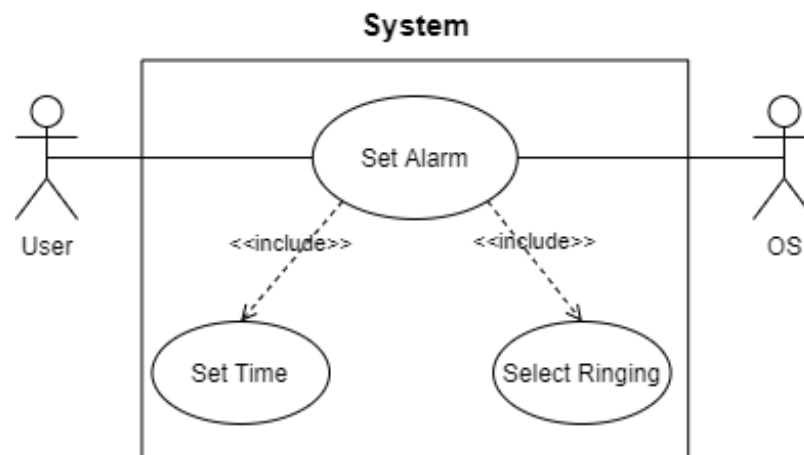
The system is ready to take next job.

2.1.1.15. Requirement 14 < Set Alarm>

Description & Priority

When user clicks on Alarm present in nav bar. The system will open Alarm activity. There user should be able to set Alarms.

Use Case Diagram



(Figure 15)

Scope

Figure 15 shows the scope of this use case is to enable user to set alarm using OS.

Description

This use case describes the procedure of setting up alarms when user selects 'Alarm' from the nav bar.

Flow Description**Precondition**

Device should be connected to internet and user must be logged in. The System provide nav bar access to the user.

Activation

The use case starts when user click on 'Alarm' from the nav bar.

Main flow

1. The user selects the time and click done.
2. The System will set alarm on that specific time.

Termination

The Use case is terminated when User successfully set an alarm.

Post condition

The system is ready to take next job.

2.1.2. Data Requirements

This section will describe the data requirements for Gill's POS system, which are crucial to implement the functionality of this application. The stored data which user needs must be available and accessible at all times.

For user to access this application, the user must sign up by providing name, email and password which will be stored in a database and will be retrieved and authenticated whenever user performs log in. User can also sign up through their Gmail account credentials in that way they must be verified from Google database and if Google approves the user id, then app will move to the home activity.

- User activity record login credentials: Application will store those credentials and let user to use the app without forcing him to login every single time unless he logs himself out every time after using the app. Application will also allow user to reset their password by taking valid email from him and sending him rest link.
- Profile activity: This will allow user to add or edit their personal details. It includes profile picture, name, address, and phone number. Profile pic will be stored on cloud real time storage while rest will be uploaded to cloud real time database. All this will push to cloud when user hits 'Save' button. User can retrieve and perform changes whenever they need.
- Sales activity: When user will click on 'Sales' icon and start performing the sales by picking items available on screen. All this information like customer name, items, quantities, amounts and total bill will firstly be stored on local variable before pushing to the cloud. Then receipt generator will extract the information out from variables and generate pdf. User can also confirm his sales even before generating receipt and taking payment.
- Payment Activity: When user will click pay button after adding all items to the basket. The total bill will then be stored on separate variable and accessible from both cash & card activities for taking payment from the customer. After successful transaction, all sale details will be pushed to the Realtime database.
- Scanner activity: This activity will electronically scan and decode different types of Bar & QR codes by using device camera. After scanning it will extract and display all data stored in a code. All this will happen by using Machine learning(ML Kit).

- Add and Edit inventory activities: These 2 activities will work in quite similar way. Both will store user entered information into local variables and then push all in once to a cloud. The only difference is, in Edit activity after fetching specific item details it will store it in variables and then display it to a user via text fields. Then user will perform some changes and click 'update' button. After clicking, that information will firstly be overwritten in local variables and then will push to the cloud on same position from where it retrieved before.
- Google Firebase – The app will store all the data in the non-SQL back-end system, this will involve user's profiles.
- The back-end system will be link with the application using the dependencies added in the configuration files of application.
- Maps: User needs to provide his start and end location, and then select the option how he wants to travel there. Map activity stores all the inputs and then display the appropriate options.
- Dial Number: The app will store number on local variable before making call on that number.
- Chatbot: After displaying user input to chatbot screen, the app will push that input directly to AI API to get the response.

2.1.3. User Requirements

The user requirements of Gill's POS are to integrate an application to help & improve new and small-scale businesses. Find a way to help people with costs and business management issues using technology to solve these problems. Everyday shopkeeper problems highlight some of the important areas to improve day to day basic. Client expectations of POS application are:

Database: The app must use the real time database for rapid and reliable uploads and retrievals of information.

UI: The User interface has to be simple, understandable, user friendly, and meet today's styles.

Useability: The app should be easy to use. And no need to make it complex by adding random and unusable features.

Receipts: Receipts design need to be simple and more descriptive for the clients better understanding. This feature will help to develop better and strong customer & shop relationship.

Email: The app needs to have signup and login feature using valid email address for security purpose. Without valid email the user will not be able to access to the app. If user will be using Gmail email, then they do not need to sign up. They can directly login by using Google login option. Its more convenient and easy way.

Android Device: The app should run on Android Phone or Tablet and will need a minimum version of 7 or higher for faster & better loading and working. The app needs to be compatible will all Android Phones and Tablets regardless of their company make.

Internet access: The app should use device network and Wi-Fi. User will need to have an Internet access to use the application because it has to connect to a server to send and retrieval of data. The faster the internet the better the server get and post requests.

Camera: The app should work with device camera for scanning codes. Otherwise, user need to have separate camera.

Help: The app should have some kind of support section to help user when they stuck on some point. According to recent times and technology, a successful app should offer live chatbot functionality to resolve their client problems quickly and easily.

Calculation: It would be good if app will offer simple calculator functionality for quick calculations.

Maps: As this app is going to be remote app means user can access it anytime anywhere. These kind of apps should have maps feature because most users of this app would be sales rap and they will be commuting from 1 place to other for work reasons.

Make Calls: App should let a user to dial phone numbers through this app, even though it will use system network.

Payment types: App should offer different payment options to their users means user can pay their bill either my cash or card.

2.1.4. Environmental Requirements

Below are all essential requirements that should be present while developing this application.

Android Device: An android gadget will be mandatory during the operating and testing of the application in the development and testing phases. The mobile application environment will need to be able to push and synchronous to ensure user can gain access to their information. It is the better and quickest way of testing the app and makes it error free. And easier for developer to get feel how his app looks in real device.

Internet Access: Internet access will be essentially required to test features in application and connecting to database. It is also required to get access in the app.

OS (Windows): This product will be coded on Windows PC with android studio as the Android development IDE.

Photoshop/Gimp: Photoshop or Gimp will be used to tailor any images, logos, icons, and graphical assets used during the development of this application.

2.1.5. Usability Requirements

This section will cover and assess the usability requirements for this system application. This will also outline the guidelines and objectives to be met regarding the systems.

Ease of use: The application has to be user friendly and easy to use. So, all different age group people can find it easily to use.

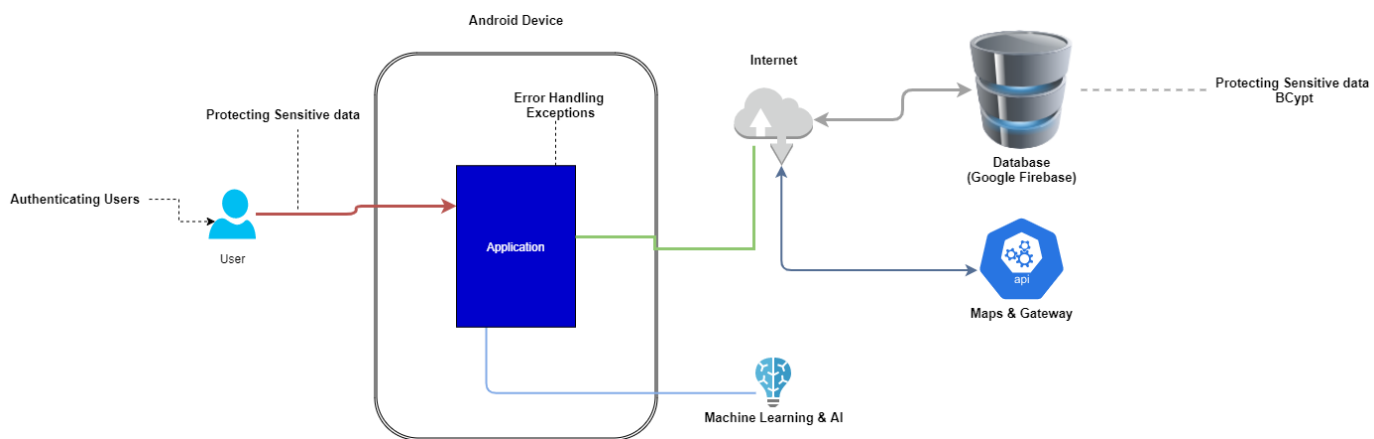
Understandability: The system must be comprehensible to use and easy to follow the functionality. Without understanding user cannot use it.

Operability: The system should act as cited in the requirement. The application should be coherent in terms of functionality.

Attractiveness: The application should be attractive to users (Design, GUI, and Layout). The app should use logos and colours that are comfortable for the eye and commonly understandable.

2.2. Design & Architecture

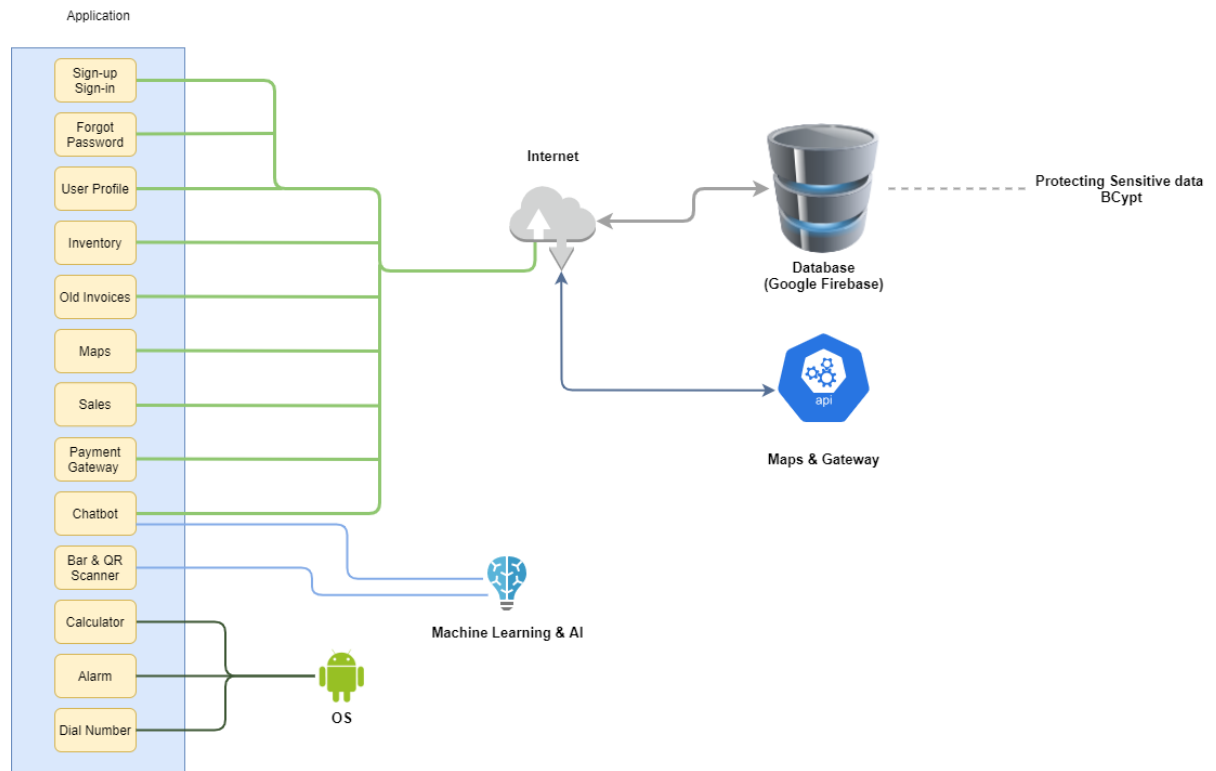
Figure 16 presents the system architecture of Gills POS System. The application is hosted locally on a device running Android OS



(Figure 16)

The Secure System Architecture in Figure 16 shows all the elements the system was made off. This architecture was selected because of its simplicity and also splitting all the major elements which permits better understanding, implementation, and flexibility of Gills POS System. The architecture also employs security throughout the system.

The application is hosted locally on device running Android Operation system. Google Firebase is the database that is used to store user credential, logging, sales history, items, etc. It is also used to store media files i.e., user profile picture. Gill's POS application also uses some API services and Machine Learning kits. Maps and payment gateway APIs are the main APIs that administered in this application. Machine learning & AI is involved in chatbot and scanning codes.



(Figure 17)

Figure 17 shows the complete insight of application features and technologies associated with each of them. As you can see Sales, Old invoices, Inventory etc are in direct contact with Firebase through internet. These features will only talk to database as per user instructions. Map, payment gateway and chatbot activities use API keys to perform certain actions defined by user. Where chatbot activity is also using text-to-speech and speech-to-text machine learning & AI Kits. While the remaining three features (calculator, alarm, dial) will only use OS when they will be called.

2.3. Implementation

Here I am describing the main functions used in my code. I have Considered to show and explain through code snippets:

Functionality: Splash Screen

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_splash_screen);

    Handler handler = new Handler();
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            Intent i = new
Intent(SplashScreen.this, MainActivity.class);
            startActivity(i);
            finish();
        }
    }, 3000);
}
```

(Figure 18 – Splash Screen - code snippet)

I have declared this activity as the starting activity of my app. `Handler()` is a constructor, and with that statement an object of data type `Handler` is created (via the `new Handler()`), which is used to wait on that screen for 3 sec and then automatically call the `Intent` to move on to the Register screen.

Functionality: Google sign in

```
private GoogleSignInClient mGoogleSignInClient;
GoogleSignInOptions gso= new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)

.requestIdToken(getString(R.string.default_web_client_id))
    .requestEmail()
    .build();
mGoogleSignInClient=GoogleSignIn.getClient(this, gso);
public void signIn(){

    Intent signInIntent =mGoogleSignInClient.getSignInIntent();
    startActivityForResult(signInIntent,RC_SIGN_IN);
}
```

(Figure 19 – Google sign in - code snippet)

Figure 19 presents the code for the Google Sign in functionality. Here I am establishing the connection to google library which enables user to have authentication and validation via Google.

```

String pdfname = (invoiceNo+1)+" Gill's POS.pdf";
pdfFile = new File(docsFolder.getAbsolutePath(), pdfname);
OutputStream output = new FileOutputStream(pdfFile);
Document document = new Document(PageSize.A4);
PdfPTable table = new PdfPTable(new float[]{5, 5, 5,5});
table.getDefaultCell().setHorizontalAlignment(Element.ALIGN_CENTRE);
table.getDefaultCell().setFixedHeight(50);
table.setTotalWidth(PageSize.A4.getWidth());
table.setWidthPercentage(100);
table.getDefaultCell().setVerticalAlignment(Element.ALIGN_MIDDLE);
;
Font f = new Font(Font.FontFamily.TIMES_ROMAN, 45.0f,
Font.BOLDITALIC, BaseColor.RED);
Font g = new Font(Font.FontFamily.HELVETICA, 25.0f, Font.NORMAL,
BaseColor.BLACK);
Font c = new Font(Font.FontFamily.HELVETICA, 21.0f,
Font.UNDERLINE, BaseColor.BLACK);
Font h = new Font(Font.FontFamily.HELVETICA, 19.0f,
Font.BOLDITALIC, BaseColor.BLACK);
Font i = new Font(Font.FontFamily.HELVETICA, 17.0f, Font.NORMAL,
BaseColor.BLACK);
Font b = new Font(Font.FontFamily.HELVETICA, 21.0f,
Font.UNDERLINE, BaseColor.RED);
table.addCell((new Phrase("Items",h)));
table.addCell((new Phrase("Quantity",h)));
table.addCell((new Phrase("Price Per Unit",h)));
table.addCell((new Phrase("Extended Value",h)));
table.setHeaderRows(1);
PdfPCell[] cells = table.getRow(0).getCells();
for (int j = 0; j < cells.length; j++) {
    cells[j].setBackgroundColor(BaseColor.LIGHT_GRAY);
}

PdfWriter.getInstance(document, output);
document.open();

document.add(new Paragraph("Gill's Shop \n", f));
document.add(new Paragraph("Unit 2, Trinity
CourtYard", g));
document.add(new Paragraph("Clondalkin,
Dublin", g));
document.add(new Paragraph("Email:-
Gillventuresltd@gmail.com", g));
document.add(new Paragraph("Tel:-
014542844 \n\n", g));
document.add(new Paragraph("Invoice No: "+(invoiceNo+1), c));

```

```

document.add(new Paragraph("Customer: "+name.getText()+"\n\n",
c));
document.add(table);
document.add(new Paragraph("\nTotal Bill: € "+amount, b));
document.add(new Paragraph("\n
Thank You!",f));

document.close();

```

(Figure 20 – Designing Receipts - code snippet)

This is just a slice of my Designing receipts code. I am using pdfFile() for designing and exporting it to pdfs. it helped me to specify colour codes, size, and pattern for the receipts. As you can see 'name' is the variable that stores Customers name. It means pdfFile enabled me to extract the stored information.

Functionality: Side Nav Bar

```

nav.setNavigationItemSelectedListener(new
NavigationView.OnNavigationItemSelectedListener() {
    @Override
    public boolean onNavigationItemSelectedListener(@NonNull MenuItem
menuItem) {

        switch (menuItem.getItemId()){

            case R.id.menu_home:
                Toast.makeText(getApplicationContext(),"Home
Page",Toast.LENGTH_LONG).show();
                drawerLayout.closeDrawer(GravityCompat.START);
                Intent h= new
Intent(SalesActivity.this,HomeActivity.class);
                startActivity(h);
                break;

            case R.id.menu_call:
                Toast.makeText(getApplicationContext(),"Call
Activity",Toast.LENGTH_LONG).show();
                drawerLayout.closeDrawer(GravityCompat.START);
                Intent i= new
Intent(SalesActivity.this,Dialer.class);
                startActivity(i);
                break;

            case R.id.menu_profile:
                Toast.makeText(getApplicationContext(),"Profile
Activity",Toast.LENGTH_LONG).show();
                drawerLayout.closeDrawer(GravityCompat.START);
                Intent j= new
Intent(SalesActivity.this,Profile.class);

```



```

        startActivity(j);
        break;
    case R.id.menu_alarm:
        Toast.makeText(getApplicationContext(),"Alarm
Activity",Toast.LENGTH_LONG).show();
        drawerLayout.closeDrawer(GravityCompat.START);
        Intent k= new
Intent(SalesActivity.this,MainAlarm.class);
        startActivity(k);
        break;
    case R.id.menu_bot:

Toast.makeText(getApplicationContext(),"ChatBot",Toast.LENGTH_LON
G).show();

        drawerLayout.closeDrawer(GravityCompat.START);
        Intent b= new
Intent(SalesActivity.this,ChatActivity.class);
        startActivity(b);
        break;
    case R.id.menu_calculator:

Toast.makeText(getApplicationContext(),"Calculator",Toast.LENGTH_
LONG).show();

        drawerLayout.closeDrawer(GravityCompat.START);
        Intent c= new
Intent(SalesActivity.this,CalculatorActivity.class);
        startActivity(c);
        break;
    }
    return true;
}
});

```

(Figure 21 – Side Nav Bar - code snippet)

Above code is for the side nav bar that is present on all activities. It includes the buttons that helps the user to move from 1 screen to another. It has sliding feature and toast message display also .

Functionality: Card Payment

```
private void makepayment() {

    Checkout checkout = new Checkout();
    checkout.setKeyID("key");
    checkout.setImage(R.drawable.pos2);
    final Activity activity = this;
    lastRef= lastRef+1;

    try {
        JSONObject options = new JSONObject();
        options.put("name", "Gill's POS");
        options.put("description", "Reference No. #" +lastRef);
        options.put("image", "https://s3.amazonaws.com/rzp-
mobile/images/rzp.png");
        // options.put("order id", "order_DBJ0Wzybf0sJbb");//from
response of step 3.
        options.put("theme.color", "#3399cc");
        options.put("currency", "EUR");
        options.put("amount", dataObj.tBill*100);//pass amount in
currency subunits
        options.put("prefill.email", "email@example.com");
        options.put("prefill.contact", "9988776655");
        checkout.open(activity, options);
        Toast.makeText(SalesActivity.this, "Invoice is Saved &
Uploaded", Toast.LENGTH_LONG).show();

    } catch(Exception e) {
        Log.e("TAG", "Error in starting Razorpay Checkout", e);
    }
}
```

(Figure 22 – Card Payment - code snippet)

As we all know accepting card payments is a crucial part of any sales software and these days card payments are more than cash. I have implemented **Razorpay** gateway that accepts payments from all kinds of ATM cards.

Functionality: Profit calculation

```
Double call=((Double.parseDouble(String.valueOf(sPrice.getText()))
)- (Double.parseDouble(String.valueOf(cPrice.getText()))));

double cal=(call /
(Double.parseDouble(String.valueOf(sPrice.getText())))*100;
inventObj.profit=Double.valueOf(decimalFormat.format(cal));

profit.setText(inventObj.profit+"%");|
```

(Figure 23 – Profit calculation - code snippet)

In this block of code, I am calculating the Profit Margin in percentage (%). Firstly, I am converting an input amount to String and then to Double because java will not allow to convert directly to Double. Then using this formula for calculation: ((Sales price – Cost price) / Sales price) * 100. I have learnt this formula by working part-time in a sales shop.

Functionality: Code scanning

```
private void scanCode(){

    IntentIntegrator integrator=new IntentIntegrator(this);

    integrator.setCaptureActivity(CaptureAct.class);

    integrator.setOrientationLocked(false);

    integrator.setDesiredBarcodeFormats(IntentIntegrator.ALL_CODE_TYPES);
    integrator.setPrompt("Scanning Code");

    integrator.initiateScan();
}
```

(Figure 24 – Code scanning - code snippet)

Here I am using IntentIntegrator for scanning and decoding all different types of codes. Code scanning is done through camera.

```

Thread thread = new Thread(new Runnable() {
    public void run() {
        try {
            if (watsonAssistantSession == null) {
                ServiceCall<SessionResponse> call =
                watsonAssistant.createSession(new
                CreateSessionOptions.Builder().assistantId(mContext.getString(R.s
                tring.assistant_id)).build());
                watsonAssistantSession = call.execute();
            }

            MessageInput input = new MessageInput.Builder()
                .text(inputmessage)
                .build();
            MessageOptions options = new MessageOptions.Builder()

            .assistantId(mContext.getString(R.string.assistant_id))
                .input(input)

            .sessionId(watsonAssistantSession.getResult().getSessionId())
                .build();
            Response<MessageResponse> response =
            watsonAssistant.message(options).execute();
            Log.i(TAG, "run: " + response.getResult());
            if (response != null &&
                response.getResult().getOutput() != null &&
                !response.getResult().getOutput().getGeneric().isEmpty()) {

                List<RuntimeResponseGeneric> responses =
                response.getResult().getOutput().getGeneric();

                for (RuntimeResponseGeneric r : responses) {
                    Message outMessage;
                    switch (r.responseType()) {
                        case "text":
                            outMessage = new Message();
                            outMessage.setMessage(r.text());
                            outMessage.setId("2");

                            messageArrayList.add(outMessage);
                    }
                }
            }
        } catch (Exception e) {
            Log.e(TAG, "Error: " + e.getMessage());
        }
    }
});

```

(Figure 25 – Chatbot - code snippet)

The above code is just a slice of full implementation of IBM Chatbot assistance. As you can see first, I am establishing the connecting then I am sending messages to get response.

This chatbot also has text-to-speech and speech-to-text functionality. It is kind of an implementation of AI & ML as it learns from user inputs.

Functionality: Dial Number

```
mNumber=findViewById(R.id.numberEt);
mBtn=findViewById(R.id.dialBt);

mBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        number = mNumber.getText().toString().trim();
        Intent intent = new Intent(Intent.ACTION_DIAL,
Uri.parse("tel:" + Uri.encode(number)));
        startActivity(intent);
    }
});
```

(Figure 26 – Dial Number - code snippet)

In this block of code, I am allowing user to type number where they want to make a call. This activity will then grab a number and start making call on it using actual “Phone” app of a device.

Functionality: Log-out

```
GoogleSignInOptions gso= new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)

.requestIdToken(getString(R.string.default_web_client_id))
    .requestEmail()
    .build();
mGoogleSignInClient= GoogleSignIn.getClient(this, gso);

btnLogout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        FirebaseAuth.getInstance().signOut();
        mGoogleSignInClient.signOut();
        Toast.makeText(HomeActivity.this, "You are Logged
Out", Toast.LENGTH_SHORT).show();
        Intent inToMain=new Intent(HomeActivity.this,
MainActivity.class);
        startActivity(inToMain);
    }
});
```

(Figure 27 – Log out - code snippet)

When user will click on logout icon this code will be called and It will log user out from their account. This code works for both, google and email log out.

Functionality: Alarm

```
timePicker = (TimePicker) findViewById(R.id.timePicker);

//attaching clicklistener on button
findViewById(R.id.buttonAlarm).setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View view) {

        Calendar calendar = Calendar.getInstance();
        if (android.os.Build.VERSION.SDK_INT >= 23) {
            calendar.set(calendar.get(Calendar.YEAR),
calendar.get(Calendar.MONTH),
calendar.get(Calendar.DAY_OF_MONTH),
timePicker.getHour(),
timePicker.getMinute(), 0);
        } else {
            calendar.set(calendar.get(Calendar.YEAR),
calendar.get(Calendar.MONTH),
calendar.get(Calendar.DAY_OF_MONTH),
timePicker.getCurrentHour(),
timePicker.getCurrentMinute(), 0);
        }
        setAlarm(calendar.getTimeInMillis());
    }
});

private void setAlarm(long time) {
    //getting the alarm manager
    AlarmManager am = (AlarmManager)
    getSystemService(Context.ALARM_SERVICE);

    //creating a new intent specifying the broadcast receiver
    Intent i = new Intent(this, MyAlarm.class);

    //creating a pending intent using the intent
    PendingIntent pi = PendingIntent.getBroadcast(this, 0, i, 0);

    //setting the repeating alarm that will be fired every day
    am.setRepeating(AlarmManager.RTC, time,
AlarmManager.INTERVAL_DAY, pi);
    Toast.makeText(this, "Alarm is set",
Toast.LENGTH_SHORT).show();
}
```


(Figure 28 – Alarm - code snippet)

This piece of code will let the user to setup alarm by using **timePicker**. It's like a clock dial because most users have found it easier to use. "TimePicker is the subclass of **FrameLayout** class".

(Android TimePicker Example - javatpoint, 2021)

Functionality: Maps

```
SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
    .findFragmentById(R.id.maps);
mapFragment.getMapAsync(this);
searchView=findViewById(R.id.Location);
searchView.setOnQueryTextListener(new
SearchView.OnQueryTextListener() {
    @Override
    public boolean onQueryTextSubmit(String s) {
        String location= searchView.getQuery().toString();
        List<Address>addressList=null;
        if(location !=null || !location.equals("")){
            Geocoder geocoder=new Geocoder(MapActivity.this);
            try {
                addressList=geocoder.getFromLocationName(location, 1);
            } catch (IOException e) {
                e.printStackTrace();
            }
            Address address=addressList.get(0);
            LatLng latLng= new LatLng(address.getLatitude(),
address.getLongitude());
            mMap.addMarker(new MarkerOptions()
                .position(latLng)
                .title(location));
            mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng,10));
        }
        return false;
    }
    @Override
    public boolean onQueryTextChange(String s) {
        return false;}
});
mapFragment.getMapAsync(this);
```

(Figure 29 – Maps - code snippet)

The above code shows the implementation of Map's feature. Firstly, I am storing user location request to a local variable then passing it to map API to get response. The response will then display to screen to help user.

Functionality: Profile

```
Dexter.withContext(getApplicationContext())
.withPermission(Manifest.permission.READ_EXTERNAL_STORAGE)
.withListener(new PermissionListener() {
    @Override
    public void
onPermissionGranted(PermissionGrantedResponse
permissionGrantedResponse) {
    Intent intent=new Intent();
    intent.setType("image/*");
    intent.setAction(Intent.ACTION_GET_CONTENT);

startActivityForResult(Intent.createChooser(intent,"Please Select
File"),101);
```

(Figure 30 – Internal Storage - code snippet)

This piece of code will enable the user to select picture from Gallery which he wants to put it as his profile picture.

```
public void updatetofirebase()
{
    final ProgressDialog pd=new ProgressDialog(this);
    pd.setTitle("File Uploader");
    pd.show();

    final StorageReference
uploader=storageReference.child("profileimages/"+"img"+System.cur
rentTimeMillis());
    uploader.putFile(filepath)
        .addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
        @Override
        public void onSuccess(UploadTask.TaskSnapshot
taskSnapshot) {

uploader.getDownloadUrl().addOnSuccessListener(new
OnSuccessListener<Uri>() {
            @Override
            public void onSuccess(Uri uri) {
                final Map<String, Object> map=new
HashMap<>();
                map.put("uimage",uri.toString());

map.put("uname",uname.getText().toString());
dbreference.child(UserID).addValueEventListener(new
ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull
```



```

DataSnapshot snapshot) {
                                if(snapshot.exists())|
dbreference.child(UserID).updateChildren(map);
                                else

dbreference.child(UserID).setValue(map);
                                }
                                @Override
                                public void onCancelled(@NonNull
DatabaseError error) {
                                }
                                });
                                pd.dismiss();

Toast.makeText(getApplicationContext(), "Updated
Successfully", Toast.LENGTH_LONG).show();
                                }
                                });
                                }
                                .addOnProgressListener(new
OnProgressListener<UploadTask.TaskSnapshot>() {
                                @Override
                                public void onProgress(@NonNull
UploadTask.TaskSnapshot snapshot) {
                                float
percent=(100*snapshot.getBytesTransferred())/snapshot.getTotalByt
eCount();
                                pd.setMessage("Uploaded :"+(int)percent+"%");
                                }
                                });
}

```

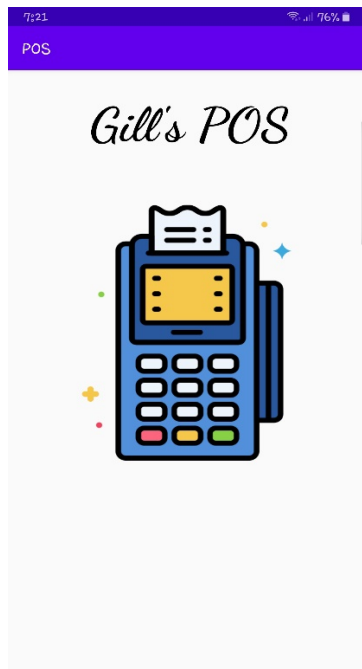
(Figure 31 – Profile - code snippet)

Once user have successfully selected his picture and typed all his details, this code will upload all that details to the cloud and will retrieve whenever he wants.

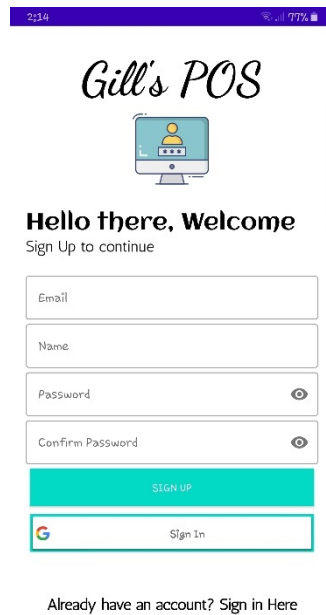
2.4. Graphical User Interface (GUI)

The application is providing the following Graphical Interface and working features to the user.

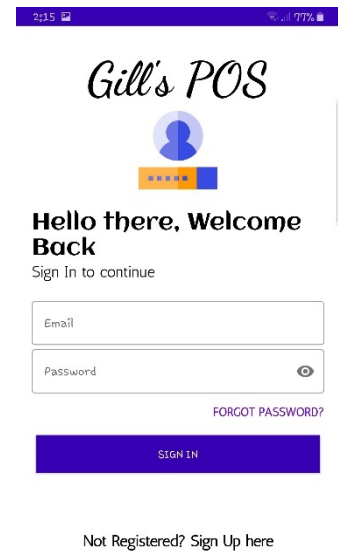
- Splash screen
- Registration page
- Log in page
- Google sign in screen
- Forgot password page
- Welcome page
- Home page
- Sales page
- Confirm order page
- Select Payment page
- Cash page
- Card page
- Invoice Template
- Scanning page
- Add Inventory page
- Edit Inventory page
- Old Invoices page
- Maps page
- Nav bar screen
- Manage Profile page
- Dial Number page
- Alarm page
- Calculator page
- Chatbot page



(Figure 32)



(Figure 33)

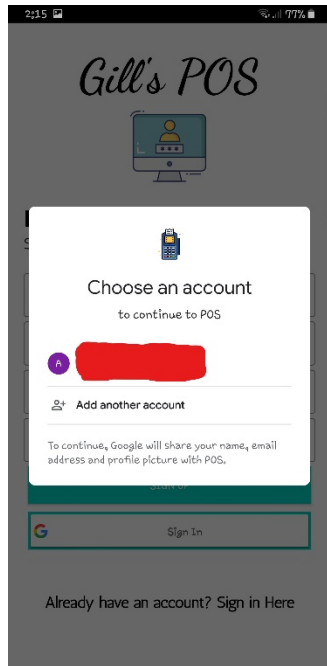


(Figure 34)

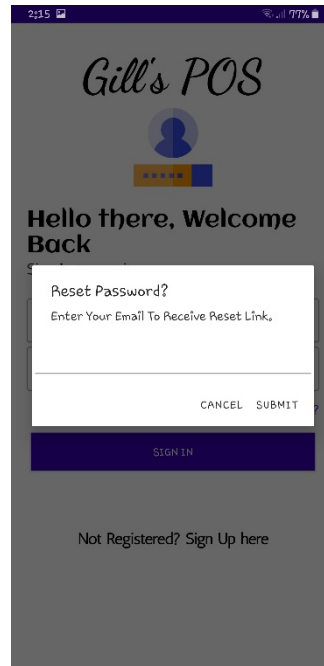
Splash Screen: As figure 32 shows, the app always starts with Splash screen displaying the app name and icon. This screen stays for 3 sec and then automatically moves to the registration page. This screen makes the app more appealing, and it is in trend also.

Registration page: Figure 33 shows the page where a new user can register with their details to create an account. After user registration by providing their details, the application will validate the email and the password that is it meeting the validation of the registration process. This page also tells the user if he forgot to input some information by highlighting specific text field.

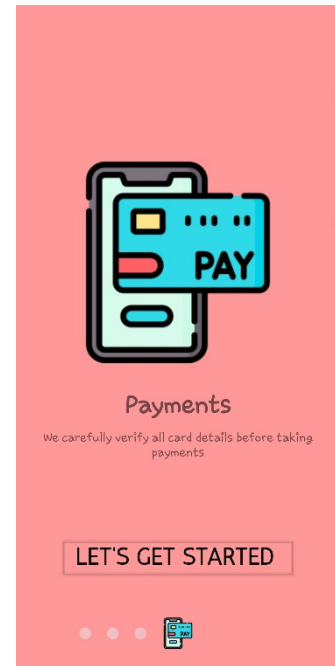
Login page: Figure 34 shows the Login page, where the user enters their info to access their account. There are 2 text fields which has to be fill in before getting access to the main menu (Home page) of Gill's POS system. Once the user fills in all his detail, it will be validated, and the user will be moved to the main menu page.



(Figure 35)



(Figure 36)

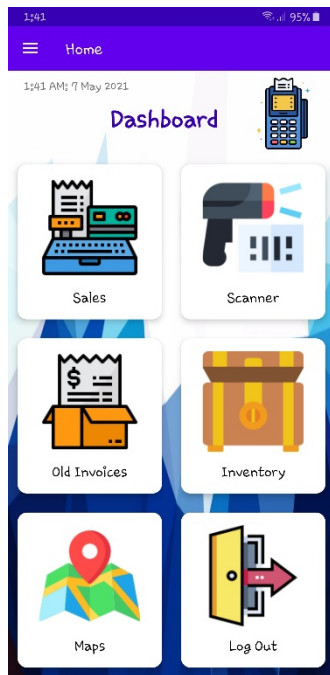


(Figure 37)

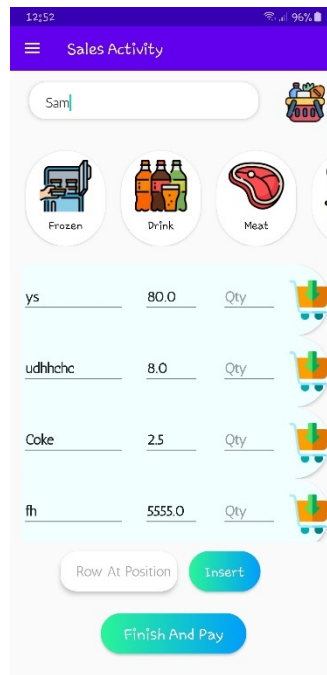
Google sign in Screen: As Figure 35 shows, Google sign in screen. It is another option to access this app. User first need to register by providing their Gmail account credentials and after that the app will automatically read it from phone.

Forgot password page: Figure 36 shows reset password functionality of this application. The user will provide their email address they registered with and the application will send the user reset instructions along with link to their email. Once the user has successfully reset their password, they will be able to login to the system using new password.

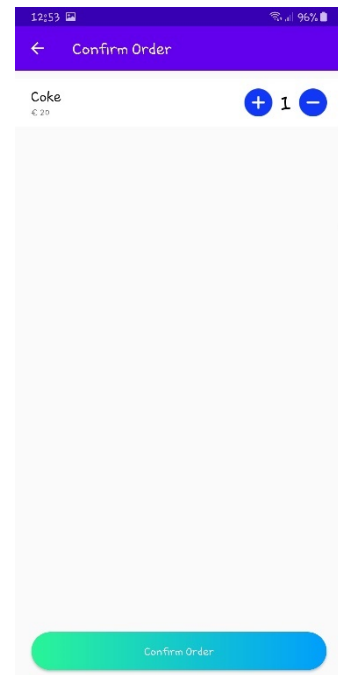
Welcome page: Figure 37 shows the welcome screen. After successful registration, a new user will bring to this screen. It has 4 pages along with Start button. Each page show glimpse of functionalities offered by this app with little animations.



(Figure 38)



(Figure 39)



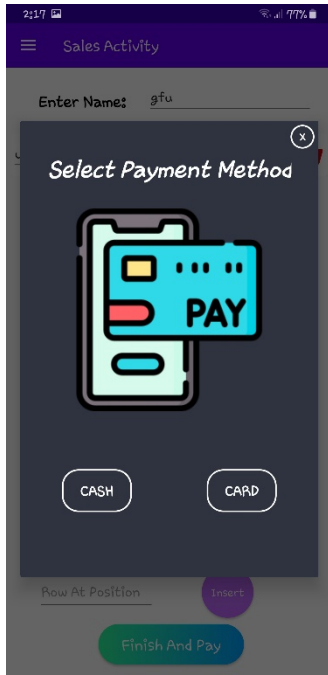
(Figure 40)

Home page: After user has been logged in to the app, the main menu page as described in Figure 38 will be shown. The main menu page will contain following options:

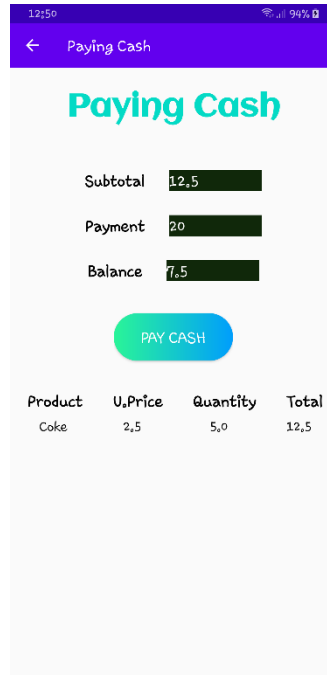
- Sales: By choosing this option user will be brought to the sales page.
- Bar & QR scanner: By choosing this option user will be brought to the scanning page.
- Old Invoices: By choosing this option user will be brought to the old invoices page.
- Inventory: By choosing this option user will be brought to the inventory page.
- Map's: By choosing this option user will be brought to the maps page.
- Log out: By choosing this option user will be logout from the app.
- Nav Bar: By clicking on nav bar more options will be appeared.
- Date & time: Shows the current date & time.

Sales page: Figure 39 shows sales activity. This page will open when user clicks on sales icon from home page. Here user will be able to make sales, insert rows at different positions, check items added to basket etc

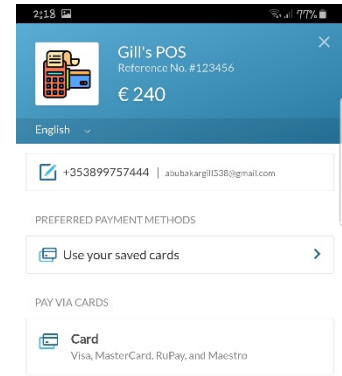
Confirm order page: When user will click on basket icon present on top right-hand corner of sales page (Figure 39), Confirm order activity will get open and will look similar to Figure 40. This page will display all items that user has added into the shopping cart.



(Figure 41)



(Figure 42)

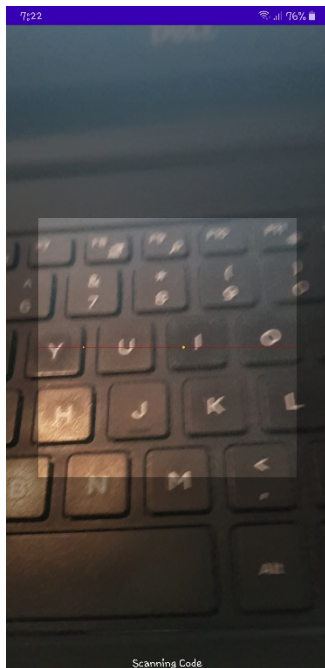


(Figure 43)

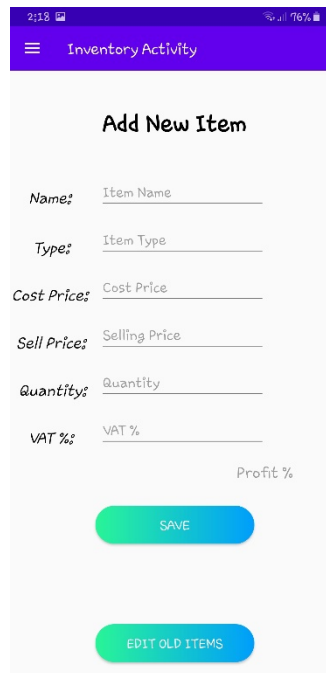
Select payment page: Figure 41 shows the select payment option. This page will enable the user to take payment from his client in either cash or card. This page will appear when user clicks Finish button present on sales page.

Cash page: Figure 42 shows the cash page. This page will appear if customer wants to pay their bill in cash. This page also provides the breakdown of bill.

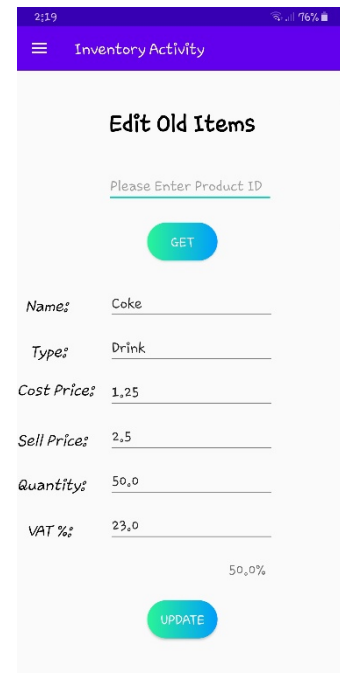
Card page: Figure 43 shows the card page. This page will appear if customer wants to pay their bill through any ATM card. User can also save card details to take more payments in future.



(Figure 44)



(Figure 45)

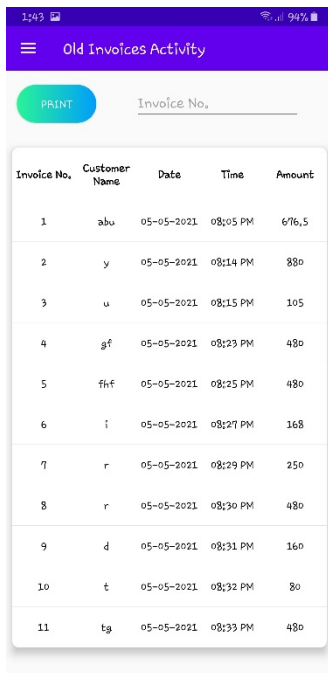


(Figure 46)

Scanning page: This activity will be used for scanning different codes and to extract the stored information as shown in Figure 44. This activity will open after clicking on scanner icon from the home page.

Add Inventory page: Figure 45 shows inventory page. This activity will open after clicking on Inventory icon from the home page. This page provides the functionality of adding new items to the database for Sales purpose.

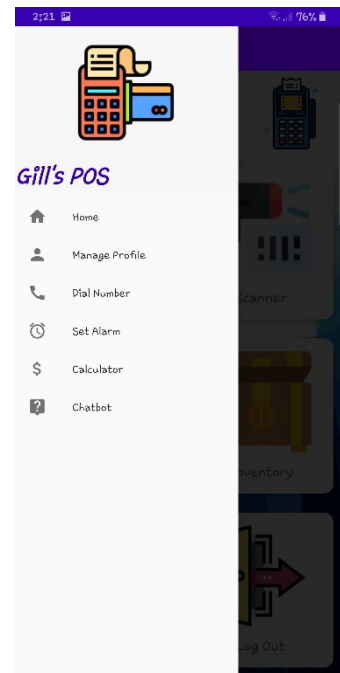
Edit Inventory page: This page (Figure 46) will open after clicking on Edit old items button from Inventory page as shown in figure 45. This page provides the functionality of editing or updating old items from database for Sales purpose.



(Figure 47)



(Figure 48)

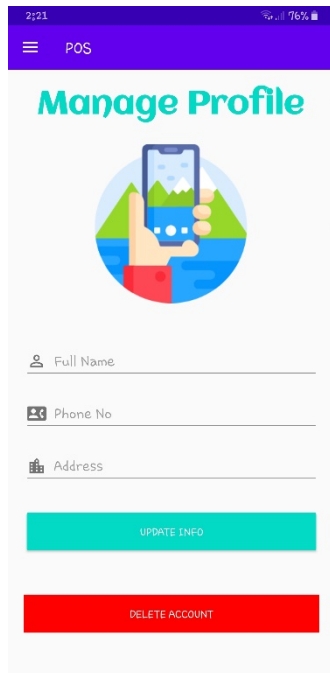


(Figure 49)

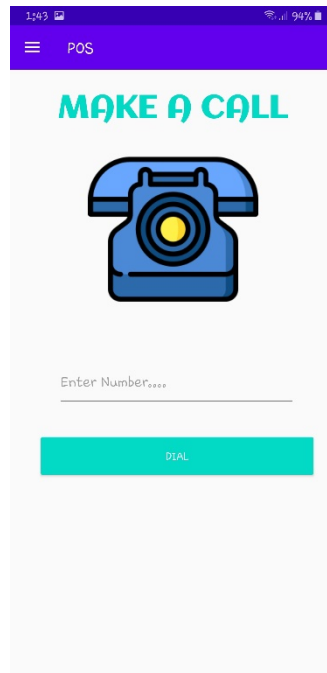
Old Invoices page (Sales History): This page (Figure 47) will show user the summary of sales they have made so far. And they can access any single invoice by providing its Id number and can print it again.

Maps page: Figure 48 shows the maps page. This page will enable the user to type their destination location and then press search. This activity will then point that location using google maps. This feature will help user to commute to different locations for various work reasons.

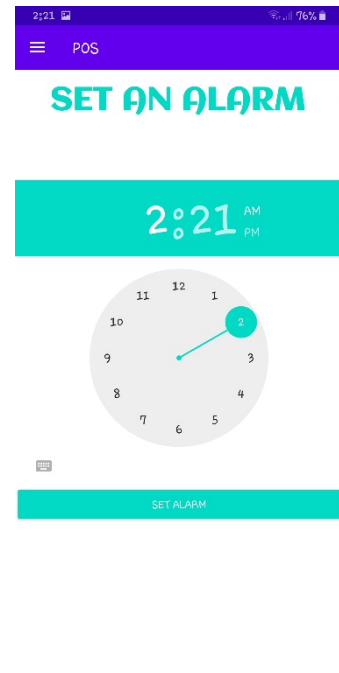
Nav bar screen: Figure 49 shows the nav bar. User can access all those features by clicking on nav bar icon.



(Figure 50)



(Figure 51)

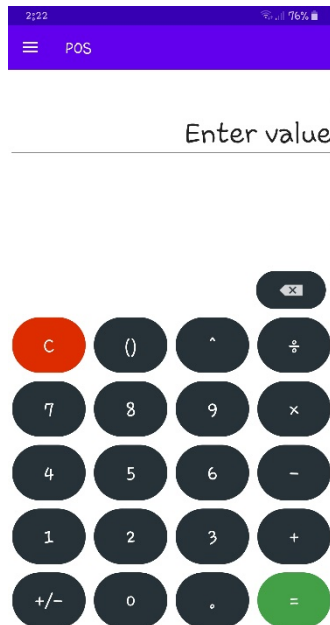


(Figure 52)

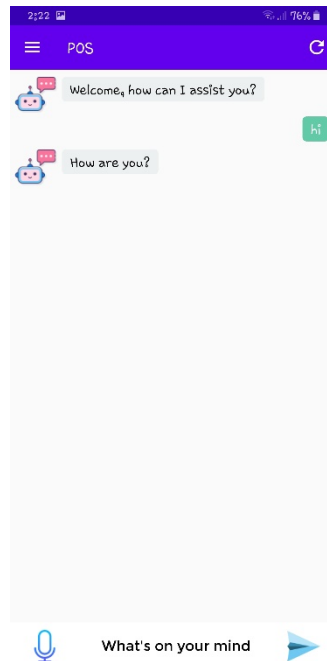
Manage Profile page: As Figure 50 shows, a signed-in user can manage their profile through this page. They can update their personal details and can also delete their account. User can access this page from nav bar.

Dial Number page: This page will let a user to dial phone number where they want to make a call as Figure 51 shows. User can access this page from nav bar.

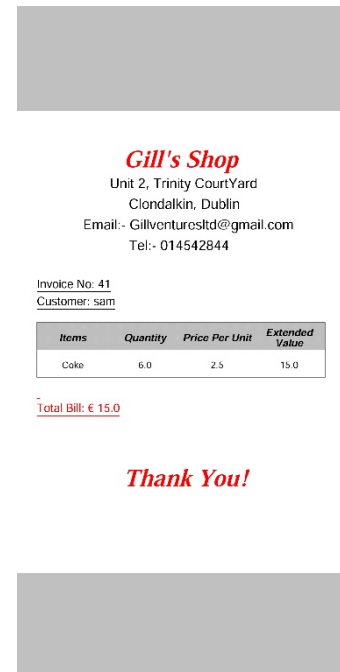
Alarm page: Figure 52 shows the set alarm page. This activity will let a user to set alarm for various purposes like reminder for start or end of his shift, etc . This feature can be accessed through nav bar.



(Figure 53)



(Figure 54)



(Figure 55)

Calculator page: As Figure 53 shows calculator feature of this app. There user can input multiple values to perform different types of calculation operations. User can access this feature from nav bar.

Chatbot page: Figure 54 shows chatbot functionality of this application. User can input text by using keyboard or by using speech-to-text feature. User can refresh this activity by clicking on refresh icon present on top right-hand corner of the screen. User can access this page from nav bar.

Invoice Template: Figure 55 shows invoice template. All invoices will be generated using this template.

2.5. Testing

It is known that testing is one of the most important and required part of development process. Testing help developers to find bugs and also plays a vital role in the popularity of an application among all different users.

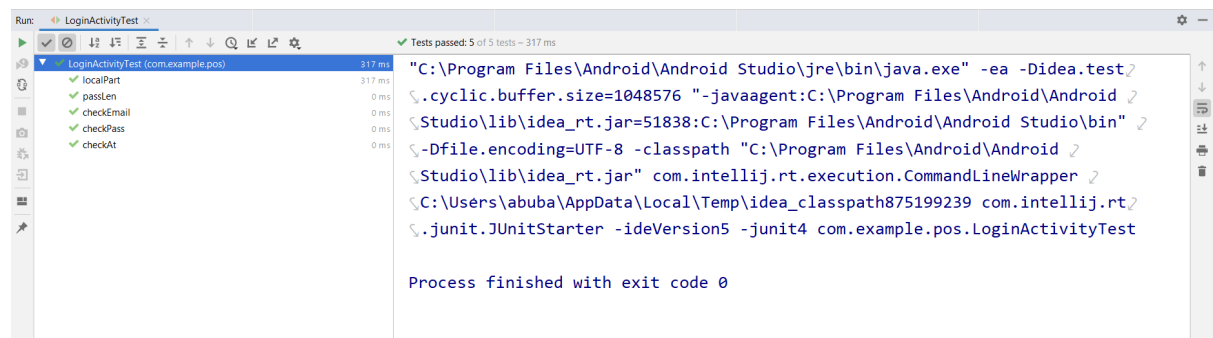
(Android APP Testing Tutorial with Automation Framework, 2021)

I have performed different types of automated testing as follows unit testing, integration testing and user interface testing.

Unit Testing

It is a type of application testing where different individual components of an application can be tested. It's purpose is to validate that each unit of an application code works as expected (GmbH, 2021). I had performed this testing during the development of an application. I had used JUnit test library in android studio to test different app features (Assert (JUnit API), 2021). Logcat in android studio also helped me to find different errors on my project and had given some vital hints on how to resolve them. Below are some examples of how I implemented this testing into my project (unit-testing , 2021).

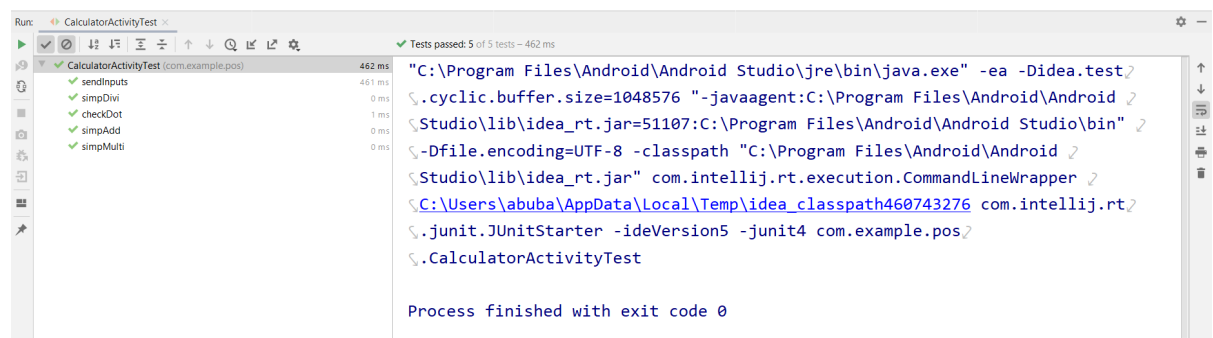
Login Activity Test



(Figure 56)

Figure 56 shows the very first implementation of unit testing. 5 different tests had been written to assess the login functionality of this app. In above tests I was checking length and whether the password and email are valid, etc.

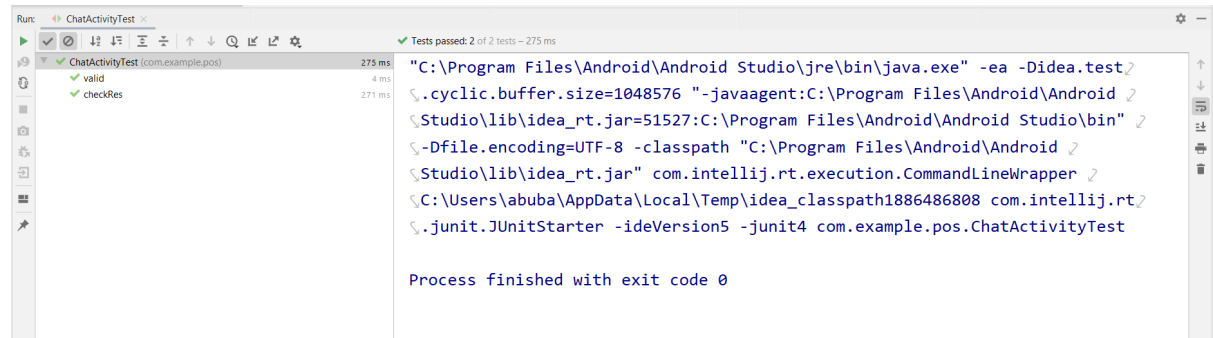
Calculator Activity Test



(Figure 57)

Figure 57 shows the success screen. All unit tests of Calculator Activity had passed and were covering all main features. As above picture shows, tests were developed on different calculator operations like validating inputs, various mathematical functions and checking results.

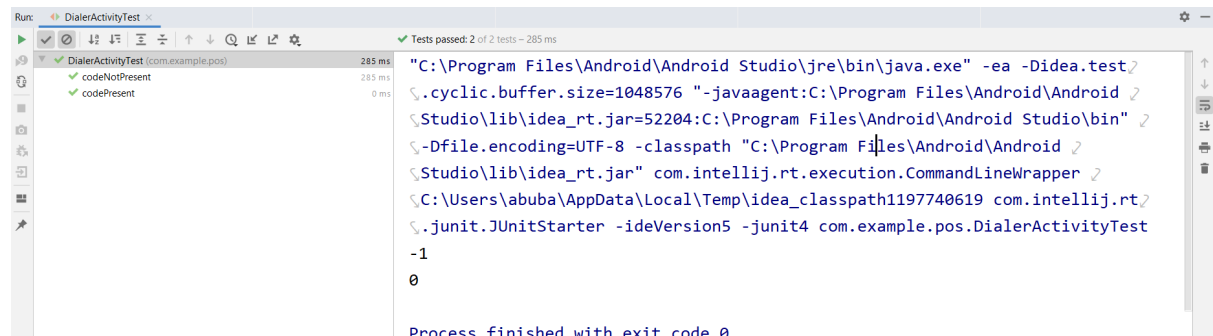
Chat Activity Test



(Figure 58)

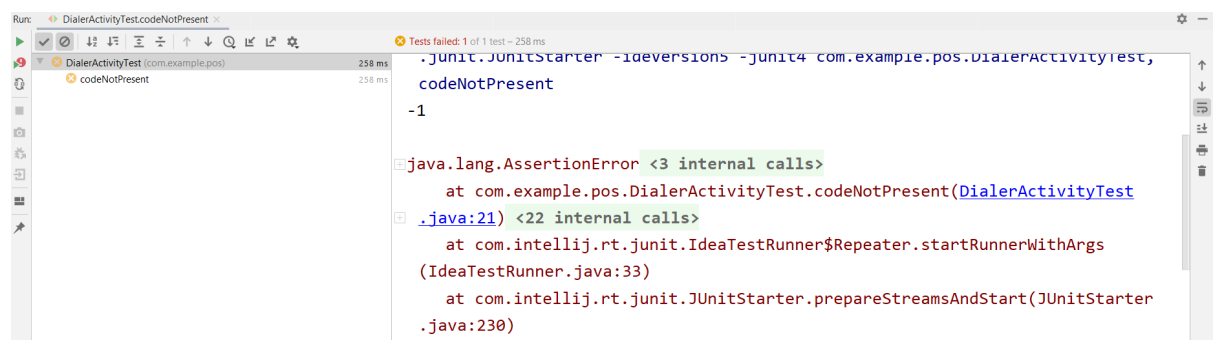
As you can see in Figure 58. All unit tests created for Chat Activity had been passed. On first test I was validating the user's input and seeing how application dealt with it. Then on second test, sending different inputs to actual activity and assessing the responses got back from that activity.

Dial Activity Test



(Figure 59)

In above picture I was testing if county code is present or not. I had written those tests for both numbers (with and without country code) and both passed.



(Figure 60)

I had made this test fail just to show how failed test looks like. If a unit test failed, the JUnit library will output an error, for example in figure 60 the error is given regarding the Assertion. In this test I was checking if country code is not present. This error was fixed by changing the Assertion type from 'assertTrue' to 'assertFalse'.

Summary Table of Unit Testing		
Total No. of Tests	No. of Tests Passed	No. of Tests Failed
17	17	0

(Table 1)

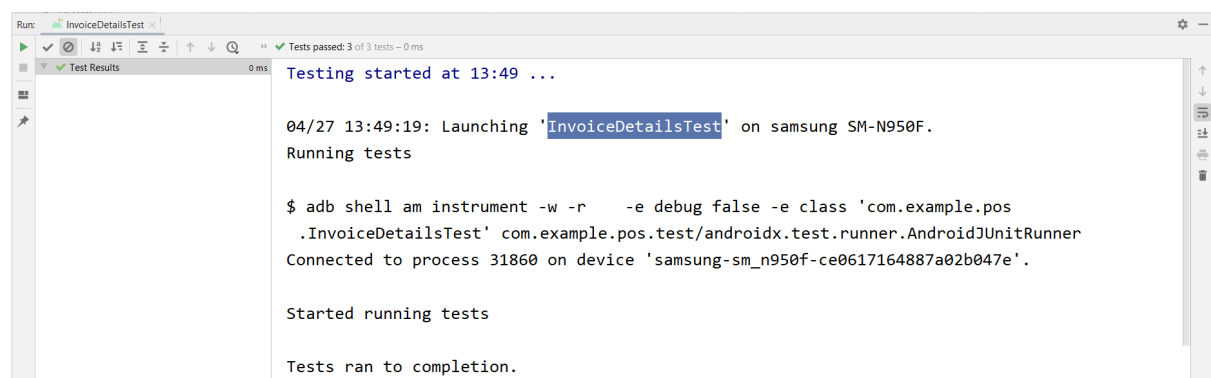
Table 1 shows a summary of the number of unit tests implemented and run for the Gill's POS System. All the tests have successfully passed.

Integration Testing

It is known as System Integration testing it is a type of software testing carried out in an incorporated software and hardware environment to validate the behaviour of an entire system. Integration tests validate how various units work together with one another. Both unit and integration tests increase the coverage faster and increases the reliability of the application. An integration test typically covers a higher volume of system than a single-class test, so it is more efficient. The advantage of performing these tests on a real device or emulator is that it will replicate the exact same environment as the user will have. But there is a major downside here: **speed**. Below are some examples of how I implemented this testing onto my project. I have used real device(*Galaxy Note8*) for this testing.

(A view on testing Android apps, 2021), (Training, 2021), (Integration, 2021)

Invoice Details Test



(Figure 61)

Figure 61 shows the very first implementation of integration testing. I had developed 3 different tests to check the functionality of DataObj class and all tests passed with success screen. DataObj is responsible for storing sales details before pushing everything to cloud.

Inventory Details Test

```

Run: InventDetailsTest
Tests failed: 1, passed: 3 of 4 tests - 31 ms
Test Results
com.example.pos.InventDetailsTest 31 ms
  typeTest 5 ms
    org.junit.ComparisonFailure: expected:<dr[]> but was:<dr[y]> <2 internal calls>
    at com.example.pos.InventDetailsTest.typeTest(InventDetailsTest.java:46) <5 internal calls>
    at androidx.test.internal.runner.junit4.statement.RunBefores.evaluate(RunBefores.java:80)
    <9 internal calls>
    at androidx.test.ext.junit.runners.AndroidJUnit4.run(AndroidJUnit4.java:154)
    <10 internal calls>
    at androidx.test.internal.runner.TestExecutor.execute(TestExecutor.java:56)
    at androidx.test.runner.AndroidJUnitRunner.onStart(AndroidJUnitRunner.java:395)
    at android.app.Instrumentation$InstrumentationThread.run(Instrumentation.java:2156)

Tests ran to completion.
  
```

(Figure 62)

As you can see in Figure 62, 3 tests passed and 1 failed. The test failed because actual outcome was not the same as expected outcome (Item type was not the same as expected ones). Logs provided me the failure reason and line of code where the error was.

```

Run: InventDetailsTest
Tests passed: 4 of 4 tests - 27 ms
Test Results
04/27 14:16:39: Launching 'InventDetailsTest' on samsung SM-N950F.
Running tests

$ adb shell am instrument -w -r -e debug false -e class 'com.example.pos
.InventDetailsTest' com.example.pos.test/androidx.test.runner.AndroidJUnitRunner
Connected to process 4601 on device 'samsung-sm_n950f-ce0617164887a02b047e'.

Started running tests

Tests ran to completion.
  
```

(Figure 63)

After fixing that failed test, all 4 test had passed with the success message which you can see in above picture.

Summary Table of Integration Testing		
Total No. of Tests	No. of Tests Passed	No. of Tests Failed
11	11	0

(Table 2)

Table 2 shows a summary of the number of integration tests implemented and run for the Gill's POS System. All 11 tests have successfully passed.

Automate User Interface Testing

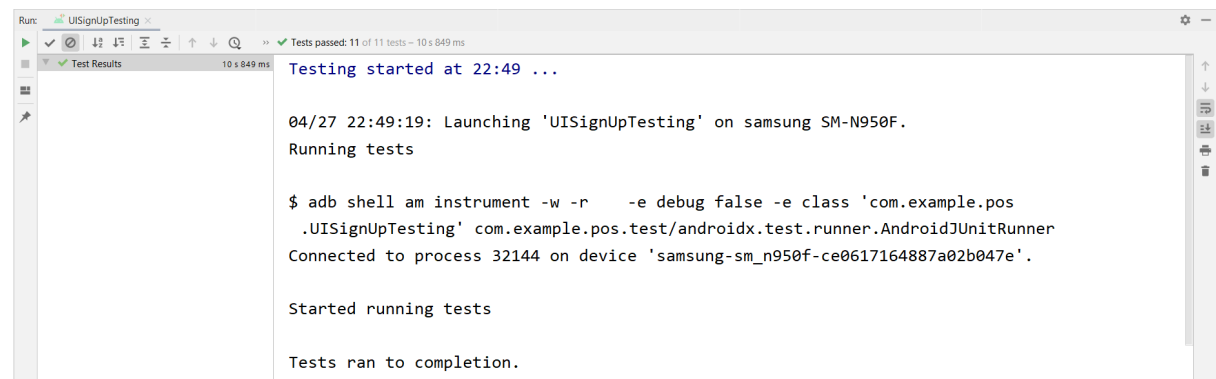
User interface (UI) testing allows a developer to make sure that an application meets its functional requirements and achieves a high standard of quality such that it is more likely to be successfully accepted by clients. One approach to this testing is to simply have a human tester perform a set of operations on the target app. However, this manual approach can be time-consuming, tedious, and error prone. A more effective approach is to write UI tests in an automated way. The automated approach permits a developer to run tests quickly and reliably in a repeatable manner.

(Automate user interface tests | Android Developers, 2021)

I have tested my app in both perspective. “Espresso” library helped me to automate my tests. Below are some from them.

(Espresso-testing, 2021)

Sign Up Test

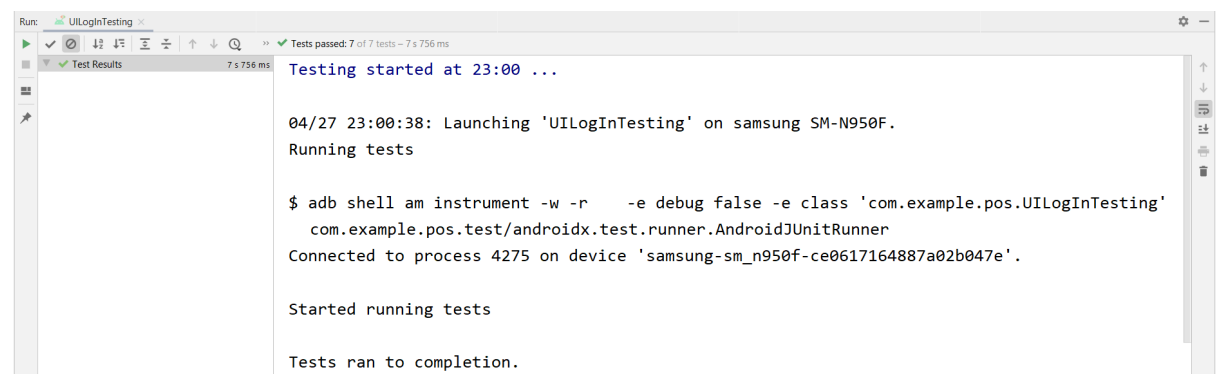


```
Run: UISignUpTesting
Tests passed: 11 of 11 tests - 10 s 849 ms
Test Results 10 s 849 ms
Testing started at 22:49 ...
04/27 22:49:19: Launching 'UISignUpTesting' on samsung SM-N950F.
Running tests
$ adb shell am instrument -w -r -e debug false -e class 'com.example.pos
.UISignUpTesting' com.example.pos.test/androidx.test.runner.AndroidJUnitRunner
Connected to process 32144 on device 'samsung-sm_n950f-ce0617164887a02b047e'.
Started running tests
Tests ran to completion.
```

(Figure 64)

Figure 64 shows the very first implementation of UI testing. I had designed 11 test to check different elements of sign-up page. These tests were covering almost all text fields, buttons, text views etc. I had got few errors before but later I was able to resolve them and then all 11 tests got passed.

Login Test

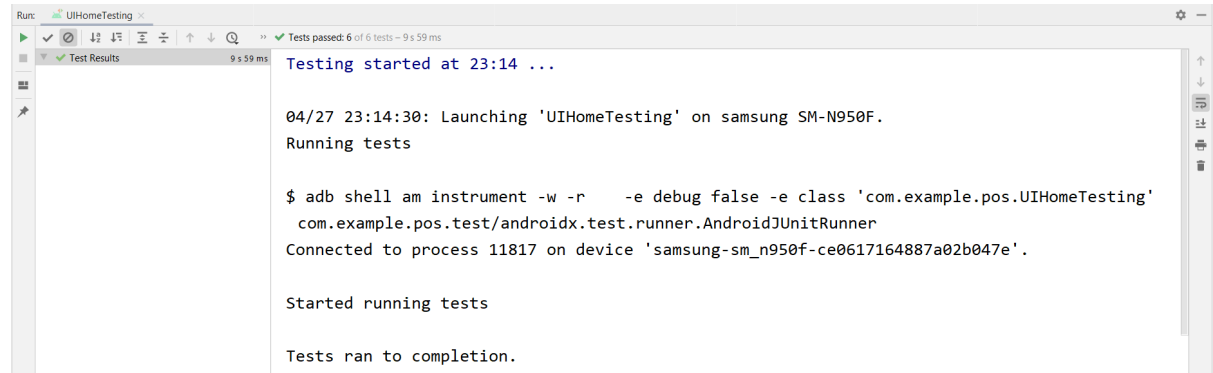


```
Run: UILogInTesting
Tests passed: 7 of 7 tests - 7 s 756 ms
Test Results 7 s 756 ms
Testing started at 23:00 ...
04/27 23:00:38: Launching 'UILogInTesting' on samsung SM-N950F.
Running tests
$ adb shell am instrument -w -r -e debug false -e class 'com.example.pos.UILogInTesting'
com.example.pos.test/androidx.test.runner.AndroidJUnitRunner
Connected to process 4275 on device 'samsung-sm_n950f-ce0617164887a02b047e'.
Started running tests
Tests ran to completion.
```

(Figure 65)

Figure 65 shows the success screen got after running Login UI tests. I had to logout first before running those tests on login page. I had designed 7 tests to check different elements present on this page. Almost all features were being tested under those 7 tests.

Home Test



(Figure 66)

These tests were developed around different features present on home screen. Tests were mainly covering menu buttons and checking how they respond when someone clicks on them. All 6 tests passed means everything was working as expected.

Summary Table of UI Testing		
Total No. of Tests	No. of Tests Passed	No. of Tests Failed
34	34	0

(Table 3)

Table 3 shows a summary of the number of UI tests implemented and run for the Gill's POS System. All 34 tests have successfully passed.

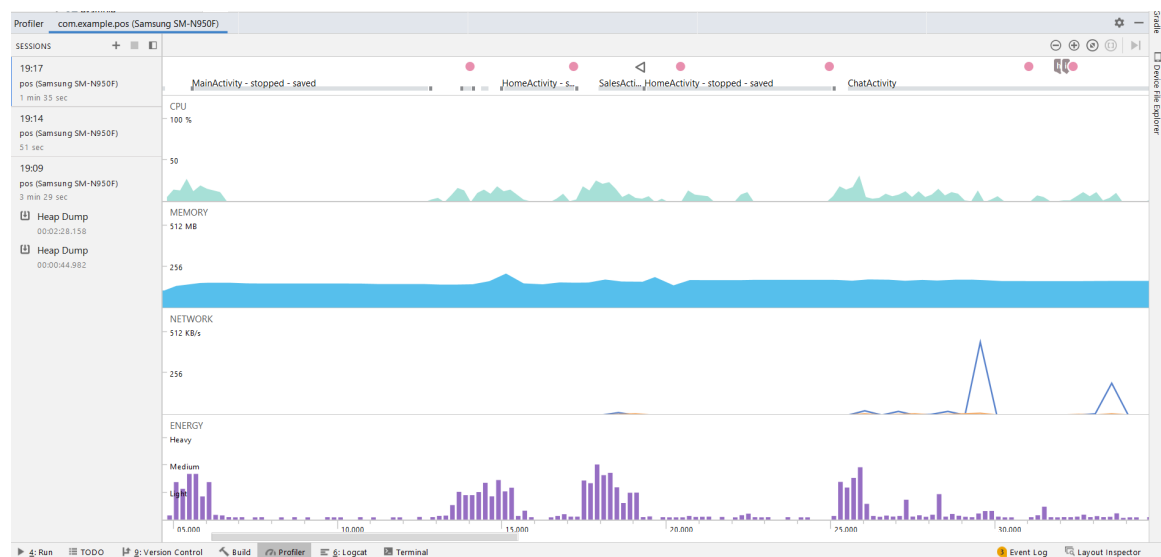
2.6. Evaluation

The results of the testing were useful as they highlighted some issues that would have stopped the project from performing its best. One such problem was missing 'tax' line of code in Inventory activity. It's one of the basic operation of this activity and its absence was causing the app and firebase to crash. This was the missing code:

```
inventObj.tax= Double.parseDouble(String.valueOf(tax.getText()));
```

Profiler tool of android studio was used to test the real-time communication and application performance. It measures for memory, light energy (Display intensity), CPU, and network activity performance. The performance was assessed on Galaxy Note8. Below is just the piece of overall analysis.

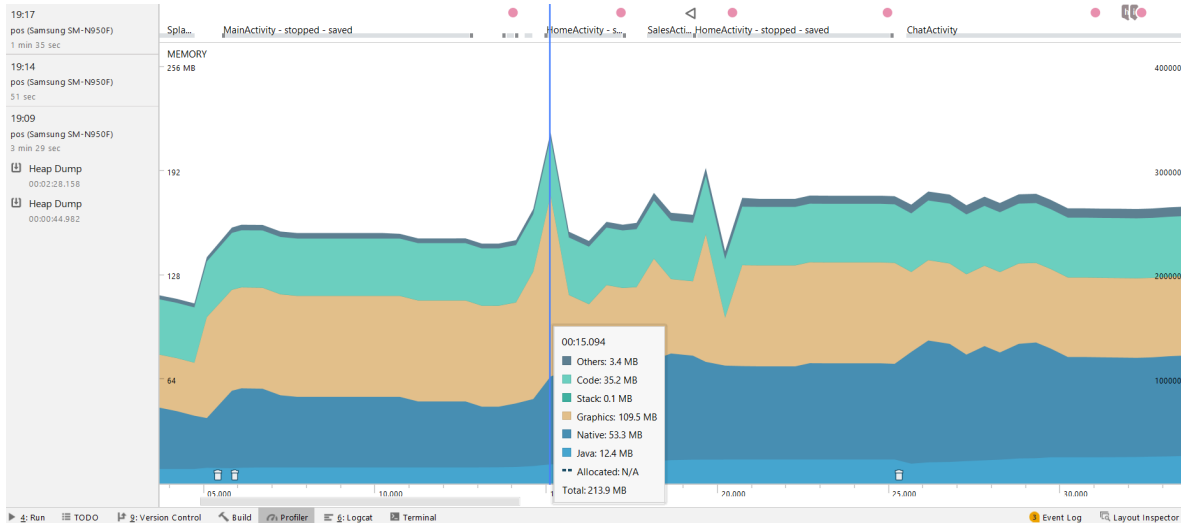
(Android-profiler, 2021)



(Figure 67)

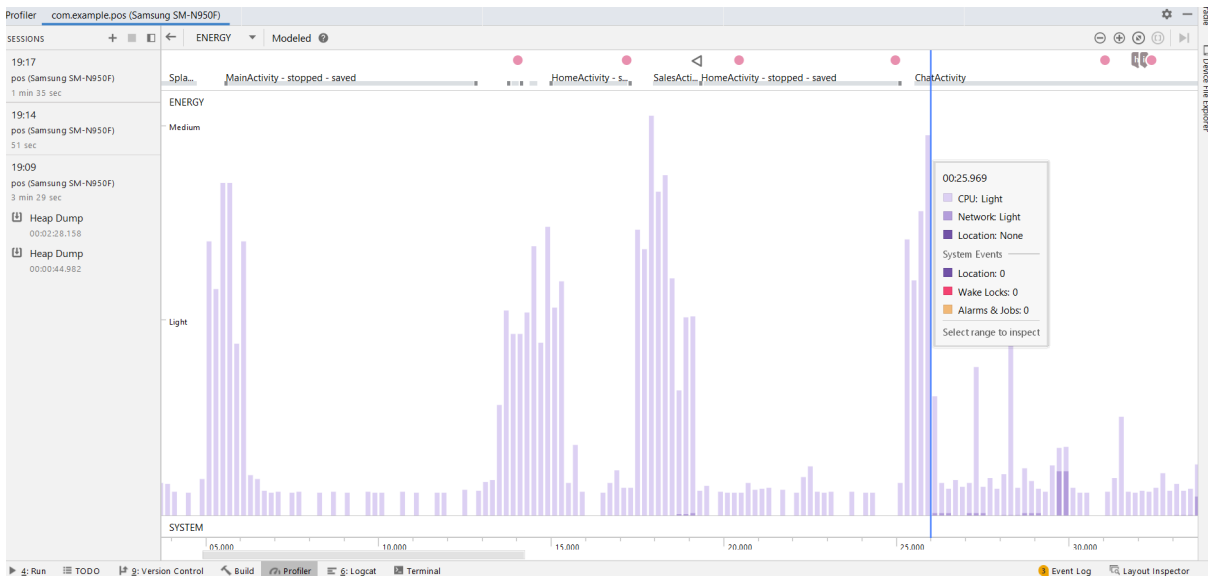
As Figure 67 shows how smoothly Gill's POS was running on Note 8 device. I tested all activities and performed all possible operations to monitor its impact on a moderate android phone. I was happy to see even my old school phone was able to run this app successfully and efficiently. This figure shows the real-time change in CPU, Memory, Network and Energy of when I was using this app on phone. All of them have some maximum and minimum values i.e., CPU: max=35% min=0%, Memory max=213.9 MB min=115 MB, Network max received=448.2 KB/s max sent=11.7 KB/s and Energy max= Medium min= Light.

Below is the further breakdown of memory.



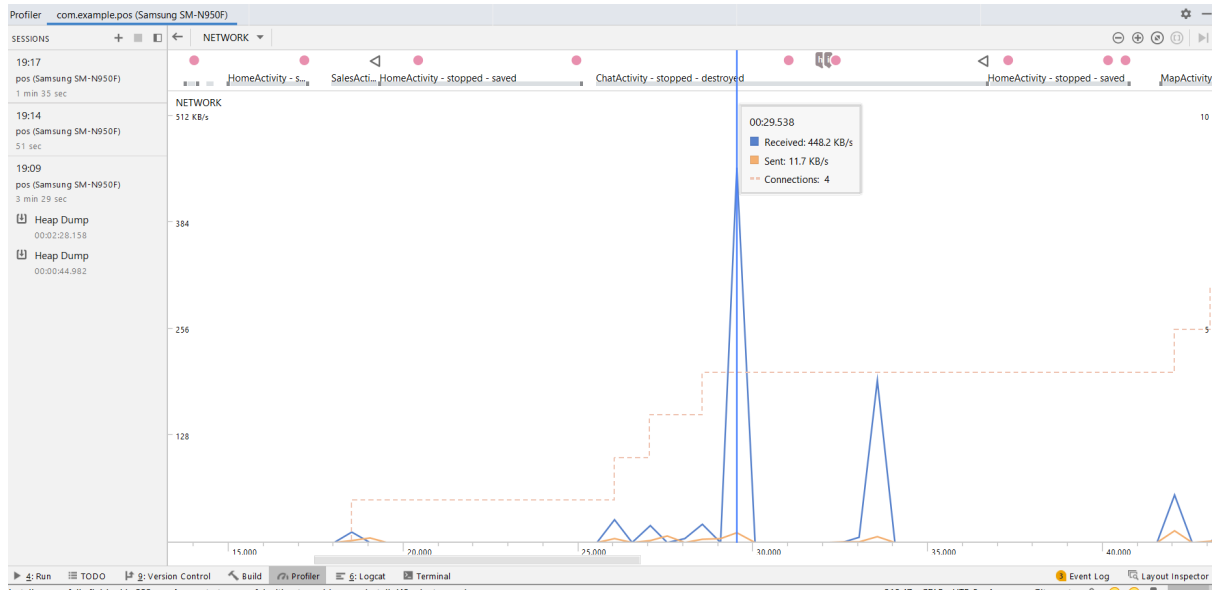
(Figure 68)

The above picture is displaying how phone Memory was utilized by this app. Graphics took max memory followed by Native and so on. Stack is the only one that was consistently lower compared to others, even during peak time it's value was still 0.1 MB. As figure 68 shows, the rate of change of memory subsections was consistent with one another.



(Figure 69)

Energy is the new feature launched by android studio in profiler. It has 3 distributions i.e., Light, Medium, and Heavy. For my project it was ranging between light and medium. It covered different elements like CPU, Network etc as shown in Figure 69.



(Figure 70)

Figure 70 shows the communication between app, APIs and Firebase through the internet by sending and receiving bytes in the form of KB/s. It also shows how many connections had been setup at each instance. The average connections were 4.

3.0 Conclusions

The conclusions I have come to while developing this Application is that most of my problems occurred with the technologies available and their limitations. One such example, as android studio authority like to change dependencies version very often, with the change of their versions they deprecate old features and offer same kind of functionalities with completely new approach and code. This issue had taken quite a lot of my development time to research and implement new versions appropriately. Sometimes technologies were quite finicky to implement.

Advantage: Gill's POS System, was developed with the aim of helping clients and their businesses. This application shows an awareness and helping side that how important is to use technology for managing a business in good and professional way. This application would allow a user to get rid from handwritten invoices which is difficult to write and to maintain old records.

This application also offers a straightforward interface i.e., easy to use and easily understandable to all users.

Disadvantage: People do not want to pay for POS systems, but they want it for free. The aim of this developed app is to help people and their businesses with very little cost.

Limits and Opportunities: The opportunity that may come for this application is that any company that work with POS Systems could support the application and be used to help and improve user experience. Also, that company could be interested in this application and may provide feedback on how to make this application better in the future.

No opportunity to update this application could be the Limits of Gill's POS system.

4.0 Further Development or Research

This application was developed to help people managing their businesses. Therefore, future development will be linked to the users who will be using this application. More features and functionalities could be added to the app.

In the next phase of Gill's POS development, I'll conduct User & UX testing and will take feedback from my clients to consider it while updating this app. I will also be improving the performance and fixing any bugs that may arise during the testing phase of the app.

The following further improvements can be released:

❖ **Notification:**

The application could have the ability to send different notifications to get the users to attract with this app.

❖ **Finger Login:**

The application could also have the ability to let the registered user to access this application using fingerprint rather than typing password every single time.

❖ **Colleagues chatroom:**

The application could have the ability to allow all colleagues of a company to chat with one another through this app

❖ **Category section on sales page:**

The application could have the ability to allow user to select category first and then app will display all items available under specific category. This advancement will make sales process easy and competent.

❖ **Confirm order:**

The application could have the ability to let user to confirm order before taking payment from their client. There user would be able to change quantity and delete any specific item that have been added to basket (Not fully implemented yet).

❖ **More advance UI:**

The app could offer more advance and reliable UI with minimum errors.

As there are no limitations on further development as long as it makes sense i.e., I will try to do more research on next phase of development to identify additional functionalities that would provide the businesses with more information into their day-to-day activities.

5.0 References

Square. 2020. Retail POS (Point Of Sale) System & Software | Square. [online] Available at: <<https://squareup.com/us/en/point-of-sale/retail>> [Accessed 2 November 2020]

Flaticon. n.d. Flaticon, The Largest Database Of Free Vector Icons. [online] Available at: <<https://www.flaticon.com/>> [Accessed 4 December 2020].

Firebase. 2021. Firebase. [online] Available at: <<https://firebase.google.com/>> [Accessed 9 December 2020].

2021. [online] Available at: <https://developer.android.com/studio?gclid=Cj0KCQjwp86EBhD7ARIsAFkgakgQSVSCa5dvkj0hZv1PE8P0Q-HWxVSiZUe09qmw3-LI6Ar4J0WCGsoaAjbWEALw_wcB&gclsrc=aw.ds> [Accessed 12 December 2020].

Java.com. 2021. [online] Available at: <<https://www.java.com/en/>> [Accessed 13 December 2020].

Firebase. 2021. Authenticate Using Google Sign-In on Android | Firebase. [online] Available at: <<https://firebase.google.com/docs/auth/android/google-signin>> [Accessed 23 December 2020].

Firebase. 2021. Manage Users in Firebase. [online] Available at: <https://firebase.google.com/docs/auth/android/manage-users#send_a_password_reset_email> [Accessed 3 January 2021].

Firebase. 2021. Manage Users in Firebase. [online] Available at: <https://firebase.google.com/docs/auth/android/manage-users#delete_a_user> [Accessed 10 January 2021].

Ibm.com. 2021. How to build a chatbot. [online] Available at: <<https://www.ibm.com/watson/how-to-build-a-chatbot>> [Accessed 13 February 2021].

Ibm.com. 2021. Watson Assistant - Overview. [online] Available at: <<https://www.ibm.com/cloud/watson-assistant>> [Accessed 14 February 2021].

Gimp.org. 2021. GIMP - About GIMP. [online] Available at: <<https://www.gimp.org/about/>> [Accessed 17 February 2021].

Razorpay. 2021. Razorpay - Best Payment Gateway for Online Payments - India. [online] Available at: <<https://razorpay.com/>> [Accessed 26 February 2021]

Google Developers. 2021. Google Maps Platform | Google Developers. [online] Available at: <<https://developers.google.com/maps>> [Accessed 5 March 2021].

Mathparser.org. 2021. mXparser – Downloads | mXparser – Math Expressions Parser for JAVA Android C# .NET/MONO/Xamarin – Mathematical Formula Parser / Evaluator Library. [online] Available at: <<http://mathparser.org/mxparser-downloads/>> [Accessed 10 March 2021].

Guru99.com. 2021. Android APP Testing Tutorial with Automation Framework. [online] Available at: <<https://www.guru99.com/why-android-testing.html>> [Accessed 21 March 2021].

GmbH, L., 2021. Developing Android unit and instrumentation tests - Tutorial. [online] Vogella.com. Available at: <<https://www.vogella.com/tutorials/AndroidTesting/article.html>> [Accessed 2 April 2021].

Junit.org. 2021. Assert (JUnit API). [online] Available at: <<https://junit.org/junit4/javadoc/4.8/org/junit/Assert.html>> [Accessed 9 April 2021].

www.javatpoint.com. 2021. Android TimePicker Example - javatpoint. [online] Available at: <<https://www.javatpoint.com/android-timepicker-example#:~:text=Android%20TimePicker%20widget%20is%20used,cannot%20select%20time%20by%20seconds.&text=widget.,the%20subclass%20of%20FrameLayout%20class.>>> [Accessed 15 April 2021].

Medium. 2021. A view on testing Android apps. [online] Available at: <<https://proandroiddev.com/writing-integration-tests-in-android-b0436978ed7b>> [Accessed 11 April 2021].

Training . 2021. [online] Available at: <<https://developer.android.com/training/testing/fundamentals>> [Accessed 13 April 2021].

Unit-testing 2021. [online] Available at: <<https://developer.android.com/training/testing/unit-testing/local-unit-tests>> [Accessed 17 April 2021].

Integration 2021. [online] Available at: <<https://developer.android.com/training/testing/integration-testing>> [Accessed 20 April 2021].

Android Developers. 2021. Automate user interface tests | Android Developers. [online] Available at: <<https://developer.android.com/training/testing/ui-testing>> [Accessed 24 April 2021].

Espresso-testing. 2021. [online] Available at: <<https://developer.android.com/training/testing/ui-testing/espresso-testing>> [Accessed 26 April 2021].

Android-profiler. 2021. [online] Available at: <<https://developer.android.com/studio/profile/android->

profiler?gclid=Cj0KCQjwp86EBhD7ARIsAFkgakhZXYSa_PnPEstPr8zFXrIfXZ4e29TbTOgy5OLxP
M3kjIZ_wPdmhPgaAv7FEALw_wcB&gclsrc=aw.ds> [Accessed 6 May 2021].

Google Developers. 2021. REST API | Google Fit | Google Developers. [online] Available at:
<<https://developers.google.com/fit/rest/>> [Accessed 9 May 2021].

6.0 Appendices

6.1. Project Proposal

6.1.1. Objectives

The main aim of this project is to build a competent Cloud based Retail sales software. This piece of software will mainly work on android phones and Tablets. And my main targeted customers will be small or newly developed business because those people are not able to spend thousands of euros to buy POS software from big companies. The application will be able to help shopkeepers and enhance their user experience by providing them some of these features:

- Admin Registration & login: It will be cloud based. So, user can register and login to access this app whenever and wherever is needed.
- Shopkeeper would be able to add new and maintain old inventories. All of this will be real time.
- He would be able to make unlimited sales.
- All transactions will have a unique number so that user can track them by using those numbers.
- Day to day closing
- Will also tell profit margin by comparing cost and sales price.
- Will also gives the option to take payment through cash or card.
- BarCode & QR scanner using camera

I believe this app will help a lot of small and new business. In that way they can keep track of their sales and profits and prevent themselves from a loss. They don't need to carry big, big screens along with them. That app on their phones can perform the same function more quickly and reliably. All calculations, entry and retrieving of information etc will be to and from real time data base. All this will be done in seconds.

6.1.2. Background

The idea behind my project is I am familiar with the area as I am working in a retail shop from last 3 years. So, I think I have a pretty good knowledge that what features a good POS system should have, how the UI should look like for all age groups to use and how to make it more user friendly and attractive. Besides all this the most important and bigger issue is Price. These days companies are charging too much for providing small, small features on POS and, maintenance and devices also costs huge. For start-up or small businesses, it is not easy for them to pay that amount.

As cloud technology is getting popular, everyone knows what the cloud is and what its benefits are. There are still not that POS software's that offers cloud feature. And the rest

who offers are very expensive, difficult to use, setting up is a complex process for nontechnical person.

Square POS is getting very famous as It offers very good and reliable service. It offers 3 different plans. Even If we look at its very basic plan where it says “\$0 Monthly Fee”. Monthly fixed fee is \$0 but on the other side they are charging 2.6% + 10¢ for in person payments and 2.9% + 30¢ for Online payments. Do not offer exchange items and reporting functionality on basic plan as it is the most basic need for all retail businesses. And its not cloud based. But square also have very good features that inspires me i.e.

- User friendly UI designs and background colours.
- Simple and easy to use.
- Does not take long to learn how to use.
- Offers sell online functionality.

(Square, 2020)

The main reason why I am focusing more on putting everything on cloud is that the owner of the shop can check sales, do invoicing, update inventories etc even while sitting at home. He does not actually need to be in shop to do all this. He can check his employees also with letting them know.

6.1.3. Technical Approach

For the project implementation I will be using Android Studio IDE. Android Studio can develop applications in both programming languages i.e. Java and Kotlin, but I will go with java. I am good in java programming language, but I am looking forward to developing more skills and challenge myself. Along the way, other languages may be use to achieve some part of the project like SQL, JSON, XML etc. It's not my first time developing an app on android platform but have never developed a complex app like this one so I will be learning this technology in more depth as moving forward. I have never used the firebase before, I am watching some good tutorials to get familiar with it and how can I achieve my objective by using firebase. I am taking everything step by step as I do not want to confuse myself and get stuck. It offers very vest variety of functionality and reliability. Firebase is a major part of my project as it offers Authentication, storage, real time database etc.

Here are the steps that I am going to follow:

1. Setting up a project in Android studio and create activities.
2. Link all activities with one another.
3. Design all those activities with appropriate Colours, images, buttons, text fields, text views etc.
4. Will setup a firebase and connect it with a project on Android studio.
5. Then will work on authentication for sign in and signup.

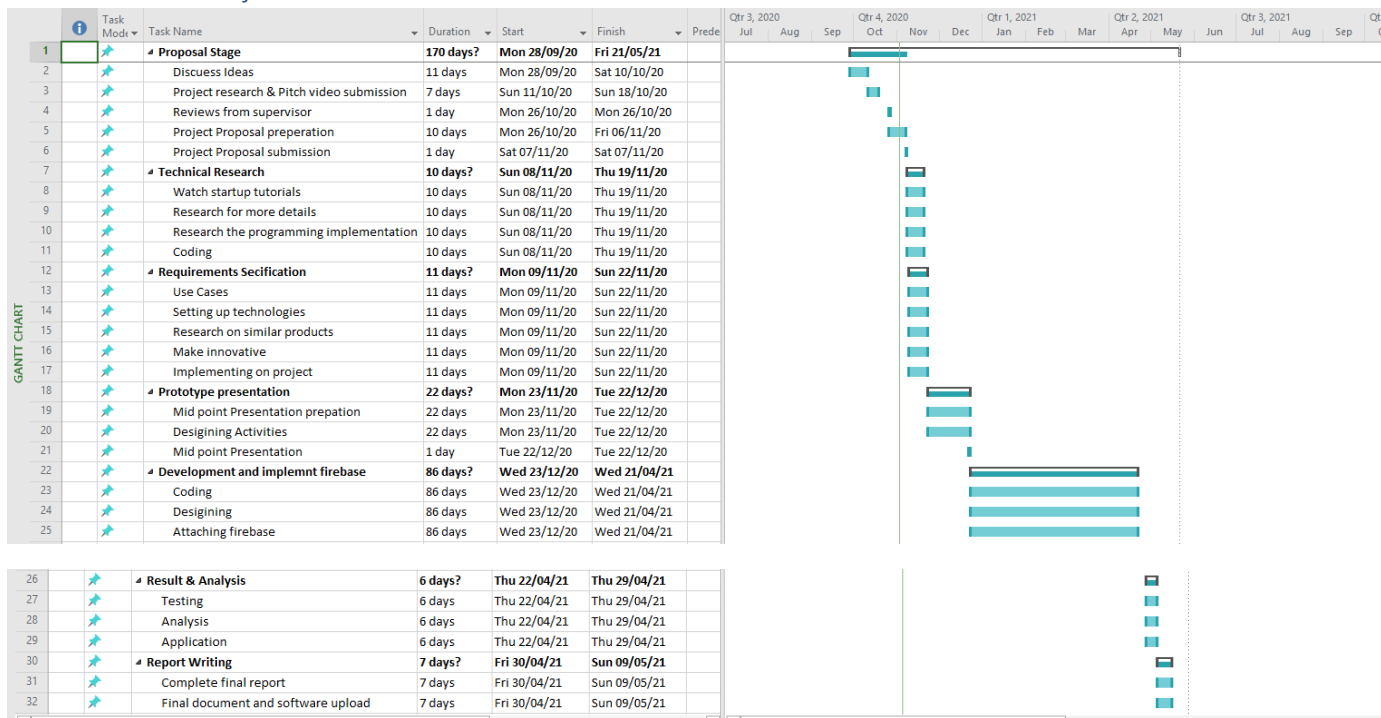
6. Link cloud storage with real time data base for storing and retrieving of information as per need.
7. Will setup barcode and QR scanning through phone camera.
8. Documentation of what I will be doing and how.
9. And so on.

It is just a high-level overview of how I am going to achieve my objective but there is a room for changing as per need.

6.1.4. Special Resources Required

The main external resource that will be required during the development of this project is android device to run the application.

6.1.5. Project Plan



(Figure 71)

6.1.6. Technical Details

The develop of my project will be based on several technologies, some of them are Android Studio IDE, JAVA, XML for designing layouts, Kind of machine learning for reading QR and Bar Codes and external storage (Firebase).

Android Studio: Android Studio is an Android's official IDE. It's objective built for Android to accelerate the development and help you build the highest-quality apps for every

Android device i.e. Phones, wearable etc. It offers custom-tailored tools to Android developers, including deep code editing, testing, debugging, and profiling implements.

Java or Kotlin: Android app is specified by XML files for designing and permissions, and back end uses java or Kotlin code depends on developer preference.

Firestore: Firestore is a real-time, scalable cloud data service. It is designed for building collaborative and real-time applications. Data in Firestore is in standard JSON format, and developers can access it using the REST API or a client library. When accessed through a client library, changes to data are synchronized in real-time to clients within milliseconds.

6.1.7. Evaluation

Testing is an essential quality control measure that I will be taking along with the development of my application. There will be number of test cases run for development life cycle of the project, verifying some errors before deeply rooted within the system. My main goal is to show requirement, design, and proper coding in my program.

Unit tests: One of the vital test in app testing strategy. After building some features or adding changes on last build, unit tests will run to capture and fix software progressing change to the app. Unit testing will verify if the class is in a right state and working as it expected.

- Local test: Unit tests that run on your local machine only (Java Virtual Machine (JVM)).
- Instrumental tests: Unit tests that run on an Android device or emulator.

Integration tests: There will be Integrational testing framework in local unit tests and UI Automator to check user interaction in equipment test. Some components will be Activity and Service testing.

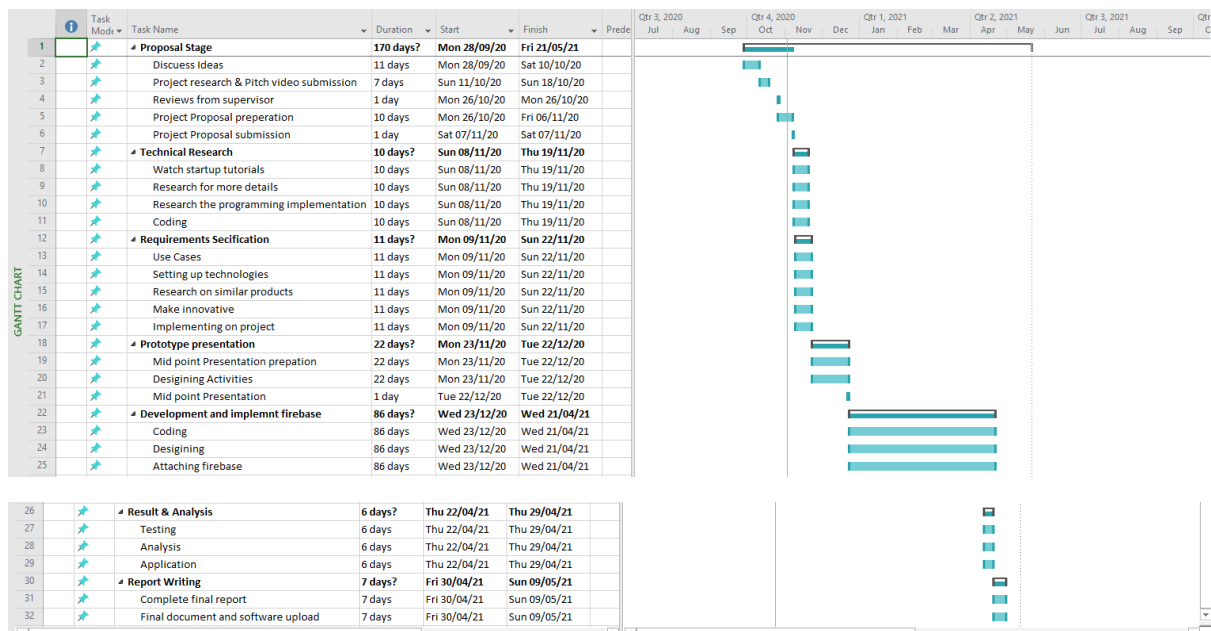
Operation tests: Also called Functional Tests, High level tests designed for checking completeness and correctness of the app. Prices may be used for operation test for the app.

System Tests: It Preformed when the app is done, testing in negation of system requirement. The testing includes GUI, Usability, Performance and Stress tests. Performance test is more focused rather than others.

6.1.8. Bibliography

Square. 2020. *Retail POS (Point Of Sale) System & Software* | Square. [online] Available at: <<https://squareup.com/us/en/point-of-sale/retail>> [Accessed 2 November 2020]

6.2. Project Plan



(Figure 72)

6.3. Reflective Journals

Month: October 2020

What?

In this month of October, we mainly got the introduction of this module i.e. what are the submissions and when is due, what we need to do each month to complete our project on time, how we can save ourselves from plagiarism and filling out Ethic & declaration form and most importantly did good amount of research to come up with some good project idea that I am going to develop in next couple of months. I have also submitted my project pitch video to get green sign on my idea.

So What?

My feeling is bit scared at this point regarding the acceptance of my idea. Project supervisors are currently working on our pitch videos and will give us feedback once they are done. First month was not that bad because we had not done much. Mainly was introduction and submitting project ideas.

Now What?

I am waiting for the feedback on my pitch video. I already sent email to my supervisor for the feedback. As soon as I get the green sign on my idea, I shall start writing my project proposal and will fill out Ethic form also which is due next. And at the same time, I shall start developing the app.

Month: November 2020

What?

In this month of November, my project idea finally got approved but needed to perform bit of amendments. After that I wrote my project proposal and finalise my project plan that how I am going to achieve my aim and how much time I will be spending to get these features working and so on. I have also watched some valuable tutorials regarding Android studio, Firebase, how to design profession looking activities and how can I add some more complex features to my project etc. I have also selected some logos, images, and colours patterns for designing purpose and to make more use attractive. I am also planning to add some machine learning on my project.

So What?

As we all know Android studio is such a good Android app's development platform. But the main issue is they change dependencies very quickly and the result of this is complete change in code. Same thing was happened with me last week which I was trying to do Machine Learning for my app and there is no new tutorial available on internet yet that implements those new dependencies. I am still doing research on this and also started to create and design Activities.

Now What?

For now, I am designing and setting up my Activities. Once I finished with the designing part, I shall start doing some java code for functionality and will try merge Firebase Authentication also. I will also start filling up my technical report document on the same time. I will continue my research for new features and good tutorials which can make my app looks more attractive and user friendly.

Month: December 2020

What?

In this month of December, I have designed and sated up my Activities. I mostly have finished with the designing part; I have also started implementing some java code for functionality and finished Firebase Authentication. This month most amount of time gone in mid-point submission. It contains writing technical report, making presentation slides, and demo of the app features developed so far. But at the same time. I did continue my research for new features and good tutorials which can make my app looks more attractive and user friendly.

So What?

Android studio IDE is such a good Android app's development platform. But I am still facing the same issue of change in dependencies. Also, before few features were free like Google maps API key but now, they are charging for it and requires huge amount of verification too. Same they did with ML-kit Machine learning and have made it more difficult and confusing at the same time. There is no new tutorial available on internet yet that apply those new dependencies. I am still doing research on this and also started to implement real-time database.

Now What?

For now, I am setting up real-time database, storage and connecting it with my app for real-time communication. Once this bit will finish, I will jump on to implement Google sign in feature. I will continue my research for new features and good tutorials which can make my app looks more attractive and user friendly.

Month: January 2021

What?

In this month of January, I have set up real-time database, storage and connected it with my app for real-time communication. I am also done with the coding side of Google sign, but testing is left for this feature. I have designed and set up few more Activities. This month most amount of time was gone in submitting TABA's of other modules. It contains writing reports after doing good amount of research. But at the same time. I did continue my research for new features and good tutorials which can make my app looks more attractive and user friendly.

So What?

Android studio IDE is such a good Android app's development platform. But it has some dark sides also like quick change in dependencies, connectivity issue with firebase. Same they did with ML-kit Machine learning and google maps. There is no new tutorial available on internet yet that apply those new dependencies. I am still doing research on this and also started to implement real-time database on my inventory and sales section.

Now What?

For now, I am testing newly built features like Google sign in, accuracy of database and organising new activities. Once these bits will finish, I will jump on to implement side NAV bar for more options, design invoice for sales and implement Razorpay for card payments. I will continue my research for new and reliable features and good tutorials which can make my app looks more attractive and user friendly.

Month: February 2021

What?

In this month of February, I have refined all the real-time communications between the app and cloud. Also performed some tests to check its efficiency and response back rate. I am also finished with the side NAV bar, designing invoices, and implementing gateway for card payments. I have done some research on implementing Chatbot to my app to make it more user friendly and also getting myself ready to start writing final report.

So What?

Android studio IDE is such a good Android app's development platform. But it has some dark sides also like quick change in dependencies, layer out issues and importing icons. There is no new tutorial available on internet yet that apply those new dependencies. I am still doing my research on this and also started to connect some in app features.

Now What?

For now, I am testing some app features and working on final report template (Understanding it). I have also a plan to implement Chatbot straight after the current work I am doing. I will continue my research for new and reliable features and for good tutorials which can make my app looks more attractive and user friendly.

Month: March 2021

What?

In this month of March, I have started writing Unit and Integration testing. I am still working on it. I have also implemented few bits of IBM Watson assistant but still there is lot more to cover to get it actually working. I have filled my project showcase template and got it approved. Updated my LinkedIn profile as per showcase standards. I have also done research on how to design an attractive project poster.

So What?

Designing images and icons of an appropriate size and pixels is not any easy job. I am struggling to get it done quickly but in well form. Also facing some difficulties while importing up to date dependencies of chat assistance. My research is still carried out on this and also started to refine some in app features.

Now What?

For now, I am writing some tests to make my app looks and works perfect. I have a plan to finish Chat bot implementation ASAP. As soon as all this will finish, I will start working on UX tests and final report. At the same time, I also will continue my research for new and reliable features, and for good tutorials which can make my app looks more attractive and user friendly.

Month: April 2021

What?

In this month of April, I have Finished writing Unit, Integration and Automated UI testing. I have also implemented remaining bits of IBM Watson assistant. I also looked over my project poster design but still need to finalise few bits. I have redesigned my sign-up, login, and home screens because the old screens were not as good and user friendly. While doing testing on my product I had investigated few errors that caused my app to crash. To fix those errors took me quite long time. I also started writing technical report.

So What?

Designing images and icons of an appropriate size and pixels is not any easy job. I am struggling to get it done quickly for my poster. Also facing some difficulties running JMeter for load testing.

Now What?

For now, I am updating my technical report with all the content which I have developed over the past few months. The documentation will go side by side with further enhancement on my project. I will also try to finish designing of my project poster ASAP. For evaluating the product, need to run JMeter and android profiler on my app. Once all this will be done, I will check and read thoroughly my technical document before submitting it, for consistency.