

# CONTENTS

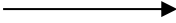
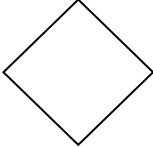






## List of figures

Figure no	Figure name	Page no
3.1	Model diagram	7
3.2	Flow chart	8
4.1	Class diagram	11
4.2	Use case diagram	12
4.3	Activity diagram	13
4.4	Sequence diagram	14

## List of screens

S.NO	Screen names	Page no
5.3.1.1	Login form	19
5.3.1.2	Signup form	20
5.3.2.1	Home page	20
5.3.2.2	Navigation drawer	21
5.3.2.3	Adding products	21
5.3.2.4	Display products	22
5.3.2.5	Edit product	22
5.3.2.6	Edit price	23
5.3.2.7	Search bar	23
5.3.2.8	Customer name	24
5.3.2.9	Select products	24
5.3.2.10	Bill	25
5.3.2.11	Bill notification	25
5.3.2.12	Low stock notification	26
5.3.2.13	Profile page	26

## Symbols

Symbol	Name	Description
	Flow line	Flow line connectors show the direction that the process flows
	Decision	Indicates a question or branch in the process flow. Typically a decision flowchart shape is used when there are 2 options ( yes or no).
	Process	A rectangle represents a process
	Dependency relation	It represents the dependent elements and the direction of dependency
	Association	Describes how the elements are associated
	Generalization	It is a parent and child relationship
	Actor	Specifies a role played by a user or any other system that interacts with system.
	Use case	Describes a set of actions that a system should perform in collaboration with one or more external users of system.

# ***Chapter 1***

## **INTRODUCTION**

### **1.1 Motivation**

The idea of the project was derived from a real-life situation. In rural areas the shopkeepers note down their stocks manually in books and it takes lot of time and work. So, to make their easier we designed this application. It reduces the effort of shopkeeper to note down the stock and reduce the stock manually when purchases are made. It also warns the shopkeeper when a product is low in stock.

### **1.2 Problem Definition**

In some rural areas most of the people in shops still do their billing on paper itself. This may take lot of time and a lot of energy gets wasted. In some cases, either shopkeepers or the customers might loose their bills and this may affect them later when it is required to tally the accounts. To overcome this type of problems stock management application helps the shopkeeper to store the stock and reduce the stock that customer has requested and to generate bill. It also gives a notification under low stock.

### **1.3 Objective of the Project**

By using this application, the shopkeeper will have benefits in keeping track of their purchases and sales. This application is more reliable and easier to work and can be used by anyone who has a basic knowledge on how to use a smartphone. This application is basically designed for rural people to calculate their bills and monthly investments. so, that they can easily calculate how much to spend and how much they have spent in the past.

## 1.4 Limitations of Project

1. Every shopkeeper should have a smart phone.
2. Internet connection is compulsory.

## 1.5 Organization of Documentation

From this document the user can easily understand the aim of this application, how the application works, what it needs, different test cases etc.

**Chapter 1:** In this chapter we discuss about motivation of our application, problems of existing system, objectives of our application and limitations.

**Chapter 2:** Here we introduce a Literature survey. We provide a thorough discussion of stock management, existing system, disadvantages, solutions and proposed system.

**Chapter 3:** We discuss about the software and hardware requirements of this project.

**Chapter 4:** The UML diagrams like use case, class diagram, sequence diagram, activity diagrams are explained here.

**Chapter 5:** Here we discussed about the methods of implementation and their results.

**Chapter 6:** Different test case scenarios are provided here.

**Chapter 7:** Here we gave a conclusion of the project and provided some future enhancements.

**Chapter 8:** We have used some documentations as a reference for this project which are provided in this chapter.

## ***Chapter 2***

### **LITERATURE SURVEY**

#### **2.1 Introduction**

This idea is emerged from our daily life real-time situations. When we go to shops to buy some groceries, cloths, medicines, food items etc., there we can observe that the shopkeepers use manual bills and records during sales and purchases from retailers, which will become a complicated task to them. So, by using this app we can reduce their efforts of maintaining hardcopies by transforming there manual billing system to digital system.

In cities, malls and super markets use software's to manage their stocks and details of their sales and purchases. But in rural areas, the shop keepers note down their stocks, sales and purchases manually. They cannot purchase those software's due to high cost. This project eliminates the paperwork, human faults, manual delay and speed up the process. This is simple and fast Stock management that can be used by anyone who has a Smartphone. By using this project, workers in the shop can also manage the stock without the presence of the owner. The customer can also track his purchases using this app.

#### **2.2 Existing System**

The present existing system is mostly done by using different software's in super markets and shopping malls, these software's are not much known in rural areas. In rural areas there are still many people who are illiterate and don't know to use computers. But now a days each and every citizen is using Smartphone either he is literate or illiterate. They are more familiar with mobile. So, to overcome these problems we have designed an app called Stock management.

In rural areas shop keepers note down their stocks manually and it is time consuming, less accurate when compared to computerized processing.

## **2.3 Disadvantages of Existing System**

- Most of rural people don't know to use those software's.
- They cannot buy these software's due to high cost.
- Since most of people in rural areas are illiterate they don't have computers in their shops.
- Lot of paper work.
- Not user-friendly environment.
- Difficult to find old records.
- Add and remove products from stock manually.

## **2.4 Proposed System**

### **2.4.1 How the application works**

Below is the detailed description of how the application works. We will understand this with an example of a retailer selling items to his customer as an example. Suppose you are shopping in a mart. Now when you have asked the items you intend to buy, then the shopkeeper checks whether the item is present in the shop or not, if available he will give the item. Then the shopkeeper will enter the items in application which you have taken. Now the application will check the each of the item and match it with those available in the database of the app. By this procedure the shopkeeper can track the sales of the items from his shop. The application now gives a clear picture to the shopkeeper about the total sales and items available in the stock. It tells him the quantities present in the shop. So now the shopkeeper can decide that which items have sufficient stock or which items needs reordering. After that the application will generate a bill of all the items that are taken by the customer and store the details in

database. In this application there will be certain limit set to the items in the stock if the limit is reached it will send a notification to the shopkeeper that the stock is low. Then the shopkeeper can reorder the stock how much he requires.

The major features in application are:

- Manual system to computerized system.
- User friendly interface.
- Saves time and paper work.
- Generates a bill for customer and updates the stock in database.
- Sends a notification when a product is low in stock.



## ***Chapter 3***

### **ANALYSIS**

#### **3.1 Introduction**

The purpose of the SRS is to provide a detailed overview of our software product, its parameters and goals. A Software requirements specification (SRS), a requirements specification for a software system, is a complete description of the behavior of a system to be developed and many include a set of use cases that describe interactions the users will have with the software. In addition, it also contains non-functional requirements. Non-functional requirements impose constraints on the design or implementation. The software requirements specification document enlists all necessary requirements that are required for the project development. To derive the requirements, we need to have clear and thorough understanding of the products to be developed. This is prepared after detailed communications with the project team and customer. A complete specification of what the proposed system should do! The SRS must correctly define all of the software requirements, but no more. The SRS should not describe design, verification, or project management details, except for required design constraints. To introduce the concepts of user and system requirements, describe functional and non-functional requirements, to explain how software requirements may be organized in a requirements document.

#### **3.2 Software Requirement Specification**

##### **3.2.1 User Requirements**

- User must at least have a basic smart phone.
- User should have a good internet connection.

- User should have a email id to register into the application.

### 3.2.2 Software Requirements

#### 1) Developer

- Operating system: Windows, Linux, Ios
- IDE : Android studio
- Database : Firebase

### 3.2.3 Hardware Requirements

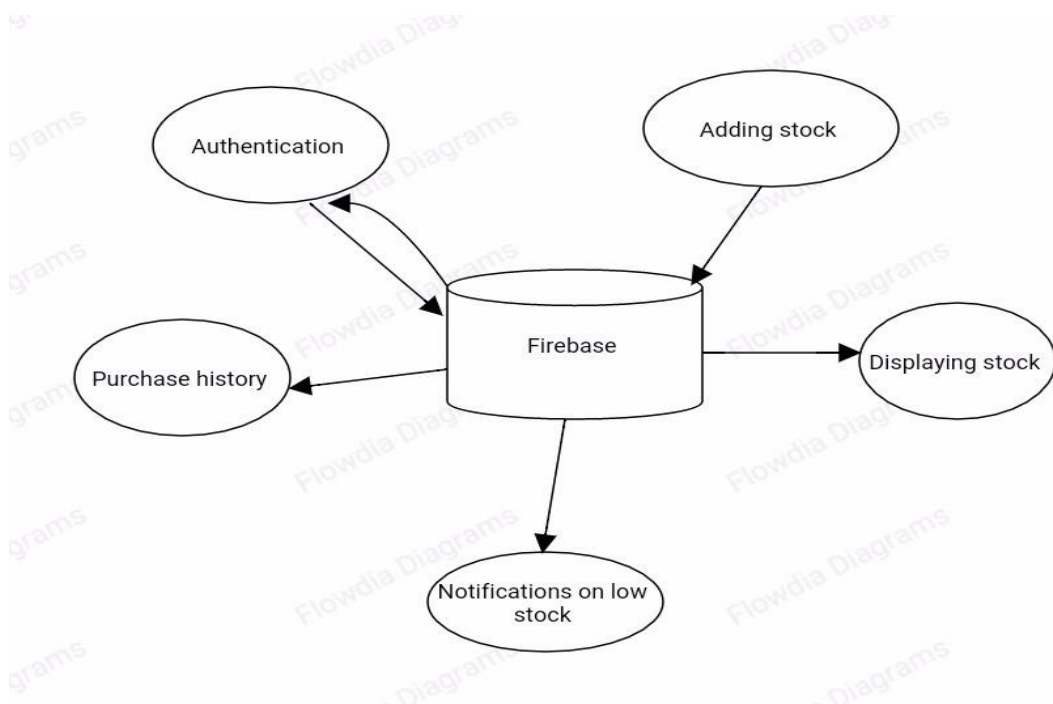
#### 1) User

- Smart phone with android OS

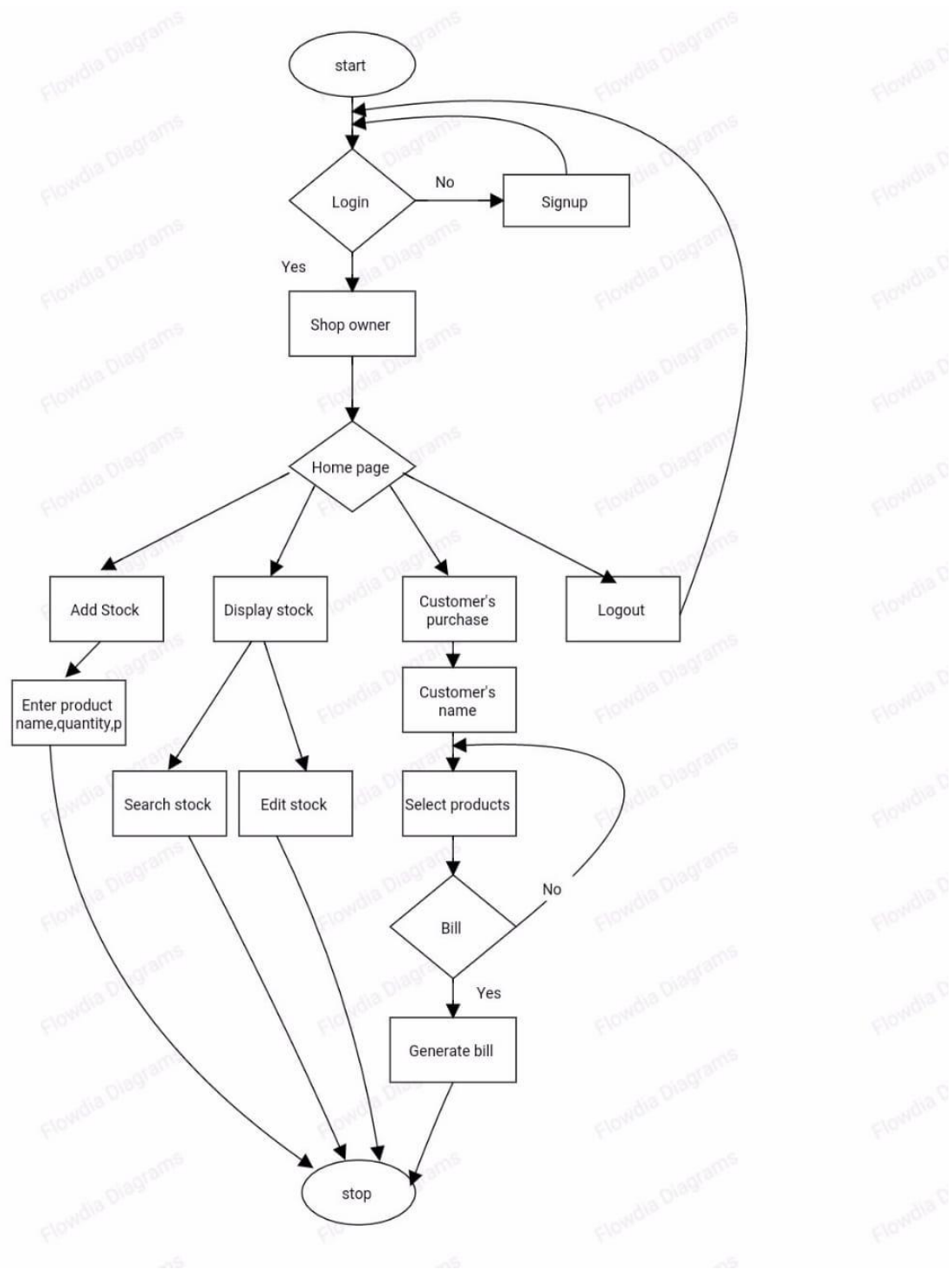
#### 2)Developer

- Processor : >i3
- RAM : > 4 GB
- Hard disk : >50 GB
- Browser : Chrome, Firefox , Internet explorer

## 3.3 Content Diagrams of Project



### 3.4 Flowchart



The above flowchart indicates the working of the application.

When the user starts the application, it asks the user for login. If the user has an account then he can login directly. If he is a new user, he has to register in the signup page and verify the email.

The home page is then displayed and he has 4 options in the navigation drawer- Add stock, Display, Purchase, Logout.

In the add stock he can add the stock to the database.

In the display option he can view and edit the entire stock.

In the purchase option he can select the stock and generate bill.

In the logout option he can logout from the application.

## ***Chapter 4***

### **DESIGN**

#### **4.1 Introduction**

Software design is a process of problem solving and planning for software solution. After the purpose and specification of software are determined, software developers will design or employ designers to develop a plan for solution. It includes low-level component and algorithm implementation issues as well as the architectural view.

#### **4.2 UML Diagram**

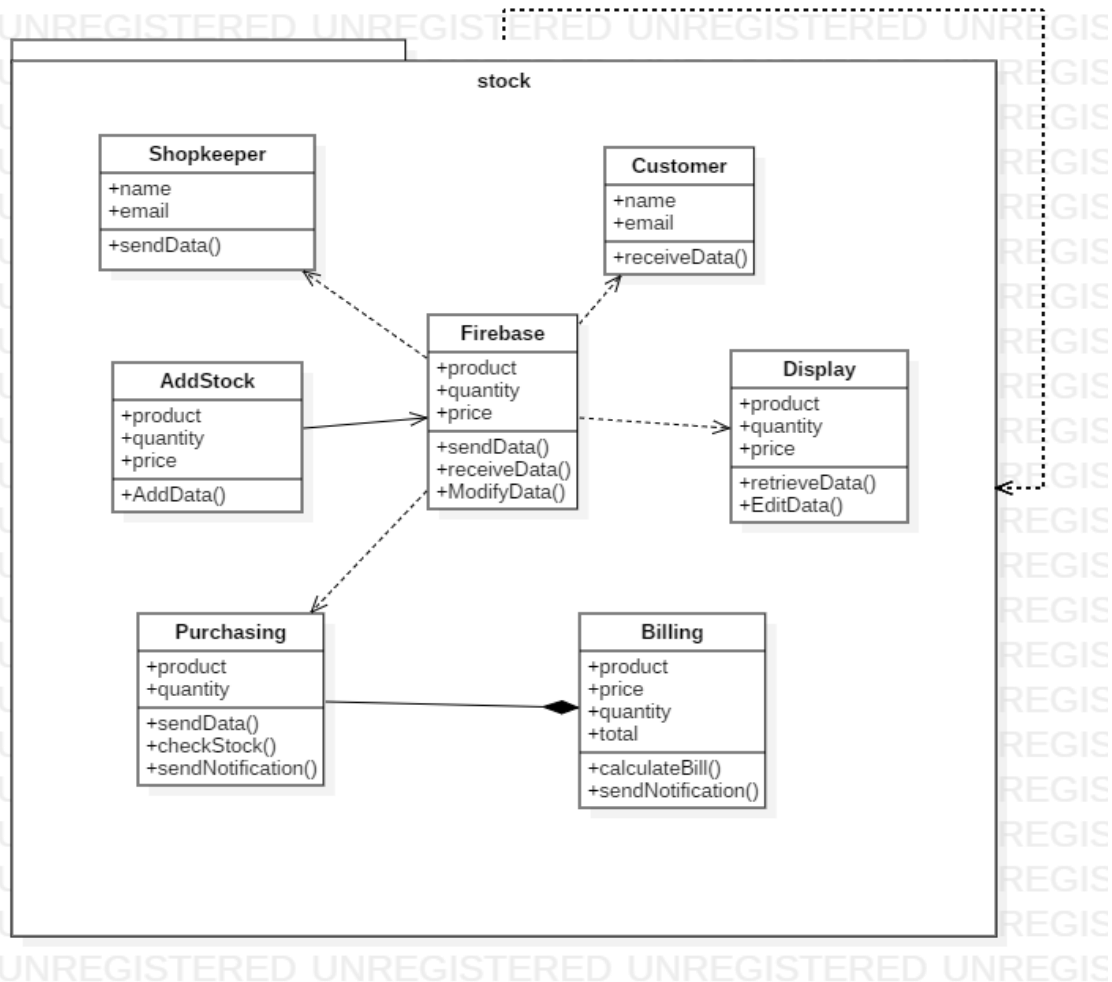
##### **4.2.1 Class Diagram**

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

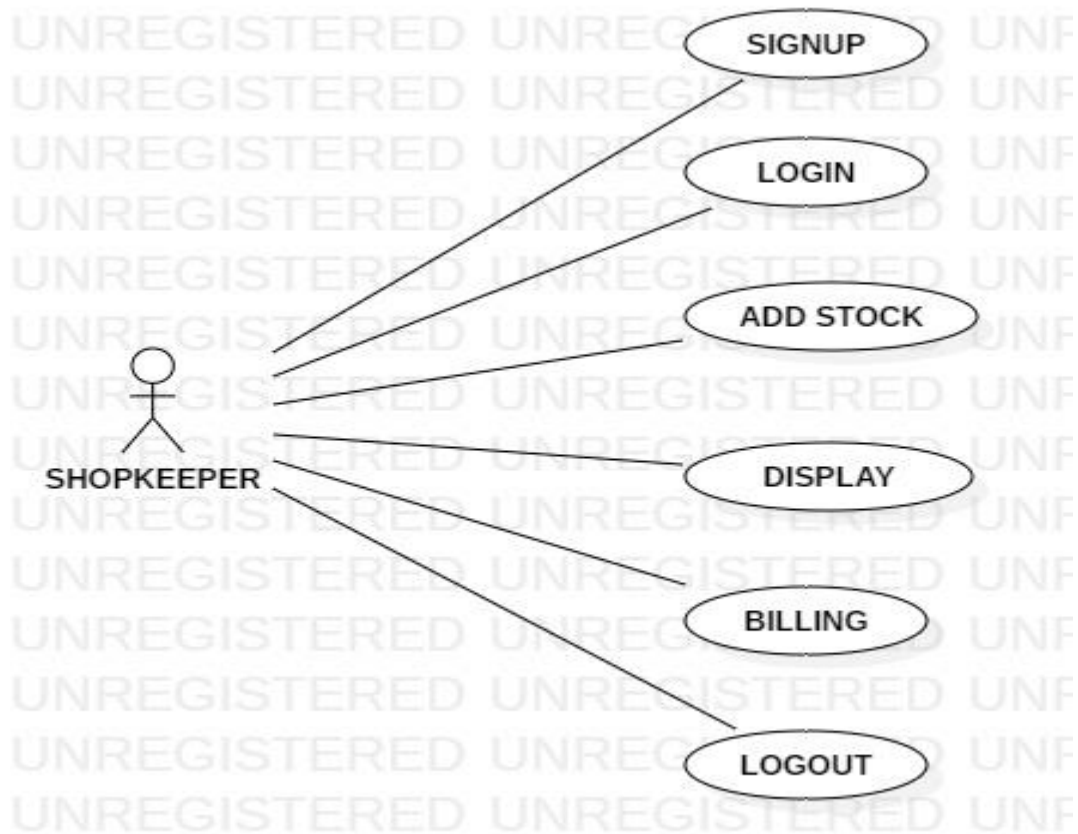
Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

The below class diagram indicates the attributes, functions and relationships between different classes.



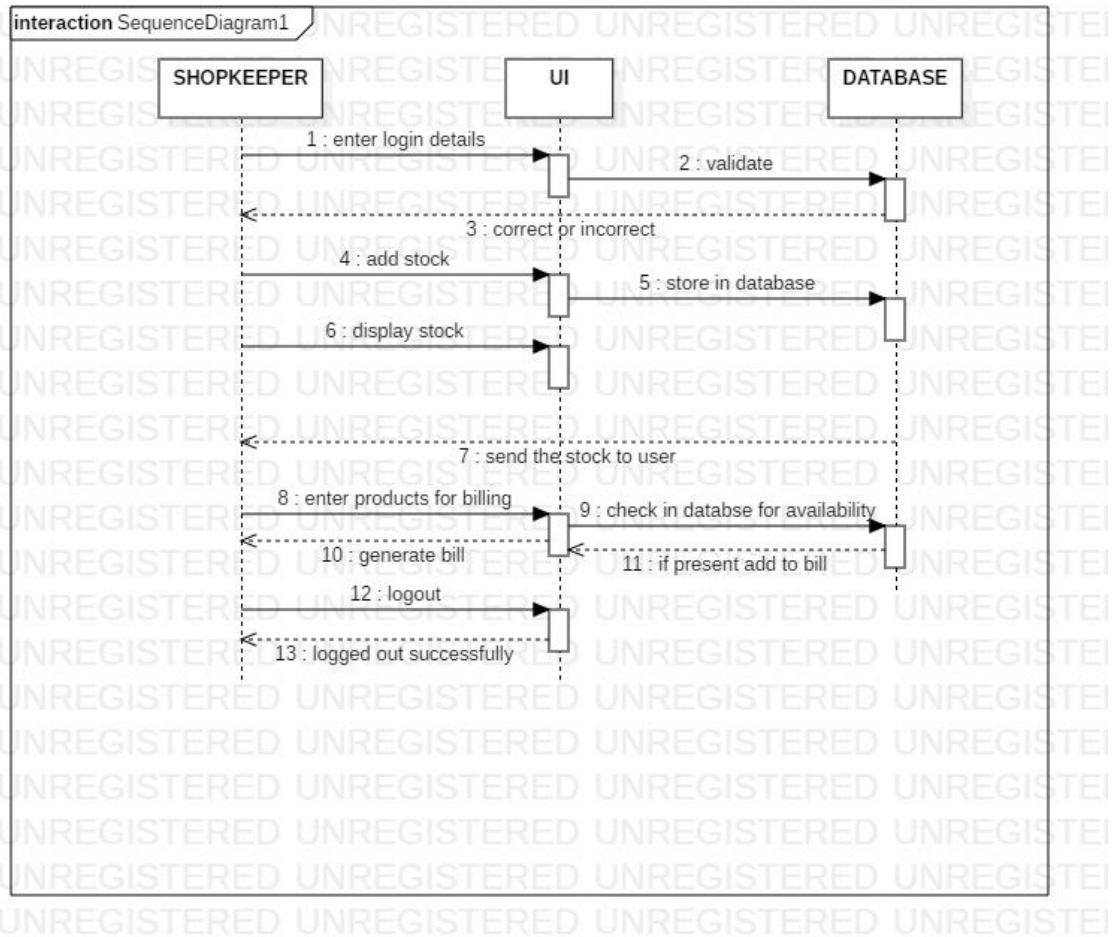
#### 4.2.2 Use Case Diagram

A use case diagram is a dynamic or behavior diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. In this context, a "system" is something being developed or operated, such as a web site. The "actors" are people or entities operating under defined roles within the system.



#### **4.2.3 Sequence Diagram**

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

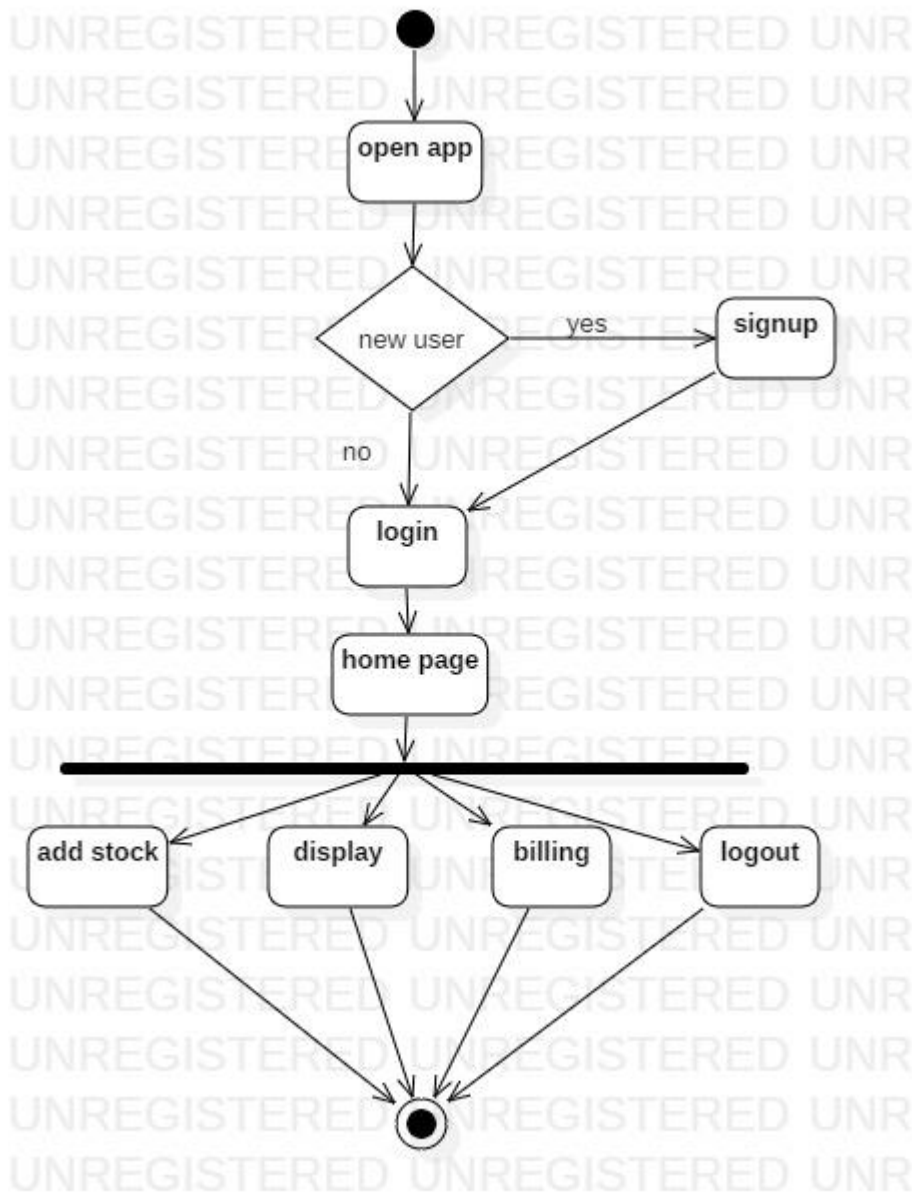


#### 4.2.4 Activity Diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.





### 4.3 Module design and organization

There are 2 modules:

1)User module:-

The user has to register in the application and then login into the application using his credentials.

2)Profile module:-

Here the user can update his profile picture and change his password.

## ***Chapter 5***

# **IMPLEMENTATION AND RESULTS**

## **5.1 Introduction**

Implementation is the carrying out, execution, or practice of a plan, a method, or any design for doing something. As such, implementation is the action that must follow any preliminary thinking in order for something to actually happen. In an information technology context, implementation encompasses all the processes involved in getting new software or hardware operating properly in its environment, including installation, configuration, running, testing and making necessary changes.

## **5.2 Explanation of Key Functions**

### **5.2.1 Android Studio**

**Android Studio** is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. The current stable version is 3.3, which was released in January 2019.

### **Features**

The following features are provided in the current stable version:

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- ProGuard integration and app-signing capabilities
- Template-based wizards to create common Android designs and components
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations
- Support for building Android Wear apps
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Android Studio supports all the same programming languages of IntelliJ (and CLion) e.g. Java, C++, and more with extensions, such as Go; and Android Studio 3.0 or later supports Kotlin<sup>1</sup> and "Java 7 language features and a subset of Java 8 language features that vary by platform version." External projects backport some Java 9 features. While IntelliJ that Android Studio is built on supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android.

### **5.2.2 Firebase**

Firebase is a real time database provided by google to store the data for any mobile or web application.

Below are the services of the firebase:

## **Firebase Cloud Messaging**

Formerly known as Google Cloud Messaging (GCM), Firebase Cloud Messaging (FCM) is a cross-platform solution for messages and notifications for Android, iOS, and web applications, which as of 2016 can be used at no cost.

## **Firebase Auth**

Firebase Auth is a service that can authenticate users using only client-side code. It supports social login providers Facebook, GitHub, Twitter and Google (and Google Play Games). Additionally, it includes a user management system whereby developers can enable user authentication with email and password login stored with Firebase.

## **Realtime database**

Firebase provides a real-time database and backend as a service. The service provides application developers an API that allows application data to be synchronized across clients and stored on Firebase's cloud. The company provides client libraries that enable integration with Android, iOS, JavaScript, Java, Objective-C, Swift and Node.js applications. The database is also accessible through a REST API and bindings for several JavaScript frameworks such as AngularJS, React, Ember.js and Backbone.js. The REST API uses the Server-Sent Events protocol, which is an API for creating HTTP connections for receiving push notifications from a server. Developers using the real-time database can secure their data by using the company's server-side-enforced security rules. Cloud Firestore which is Firebase's next generation of the Realtime Database was released for beta use.

## **Firebase Storage**

Firebase Storage provides secure file uploads and downloads for Firebase apps, regardless of network quality. The developer can use it to store images, audio, video, or other user-generated content. Firebase Storage is backed by Google Cloud Storage.

## **Firebase Hosting**

Firebase Hosting is a static and dynamic web hosting service that launched on May 13, 2014. It supports hosting static files such as CSS, HTML, JavaScript and other files, as well as support through Cloud Functions. The service delivers files over a content delivery network (CDN) through HTTP Secure (HTTPS) and Secure Sockets Layer encryption (SSL). Firebase partners with Fastly, a CDN, to provide the CDN backing Firebase Hosting. The company states that Firebase Hosting grew out of customer requests; developers were using Firebase for its real-time database but needed a place to host their content.

## **ML Kit**

ML Kit is a mobile machine learning system for developers launched on May 8, 2018 in beta during the Google I/O 2018. ML Kit API's feature a variety of features including text recognition, detecting faces, scanning barcodes, labelling images and recognising landmarks. It is currently available for iOS or Android developers. You may also import your own TensorFlow Lite models, if the given API's aren't enough. The API's can be used on-device or on cloud.

## **Stability**

### **Crashlytics**

Crash Reporting creates detailed reports of the errors in the app. Errors are grouped into clusters of similar stack traces and triaged by the severity of impact on app users. In addition to automatic reports, the developer can log custom events to help capture the steps leading up to a crash. Before acquiring Crashlytics, Firebase was using its own Firebase Crash Reporting.

### **Performance**

Firebase Performance provides insights into an app's performance and the latencies the app's users experience.

## Firebase Test Lab for Android And iOS

Firebase Test Lab for Android and iOS provides cloud-based infrastructure for testing Android and iOS apps. With one operation, developers can initiate testing of their apps across a wide variety of devices and device configurations. Test results—including logs, videos, and screenshots—are made available in the project in the Firebase console. Even if a developer hasn't written any test code for their app, Test Lab can exercise the app automatically, looking for crashes. Test Lab for iOS is currently in beta.

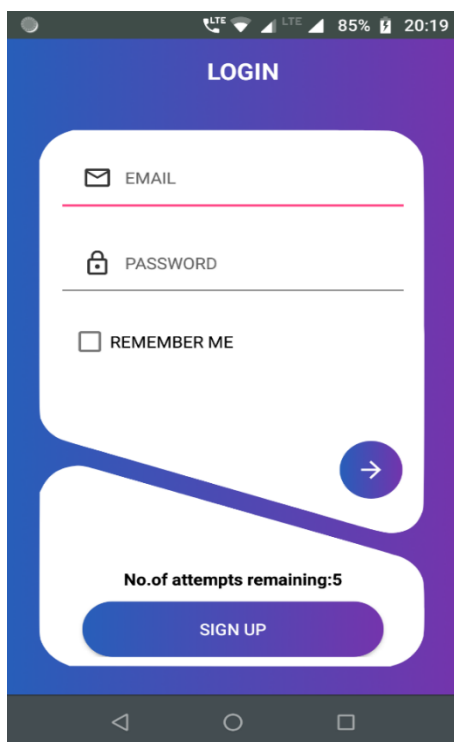
## 5.3 Method of Implementation

### 5.3.1 Forms

There are 2 forms in this application.

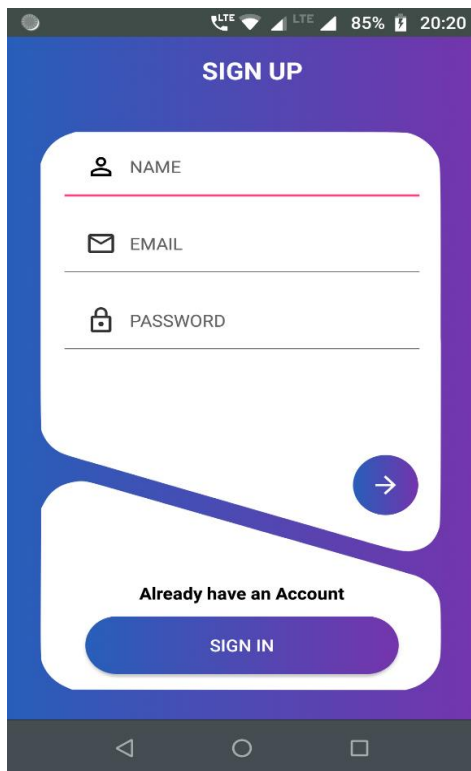
- Login form
- Signup form

#### LOGIN FORM

The screenshot shows a mobile application interface for a login form. The background is a gradient of blue and purple. At the top, the word "LOGIN" is displayed in white capital letters. Below it, there are two input fields: "EMAIL" with an envelope icon and "PASSWORD" with a lock icon. A "REMEMBER ME" checkbox is located below the password field. A blue circular button with a white right-pointing arrow is positioned to the right of the input fields. At the bottom, a text label "No.of attempts remaining:5" is shown above a blue "SIGN UP" button. The status bar at the very top indicates LTE signal, 85% battery, and the time 20:19.

**Fig:5.3.1.1** The login page requires email id and password. The login button disables after 5 incorrect logins.

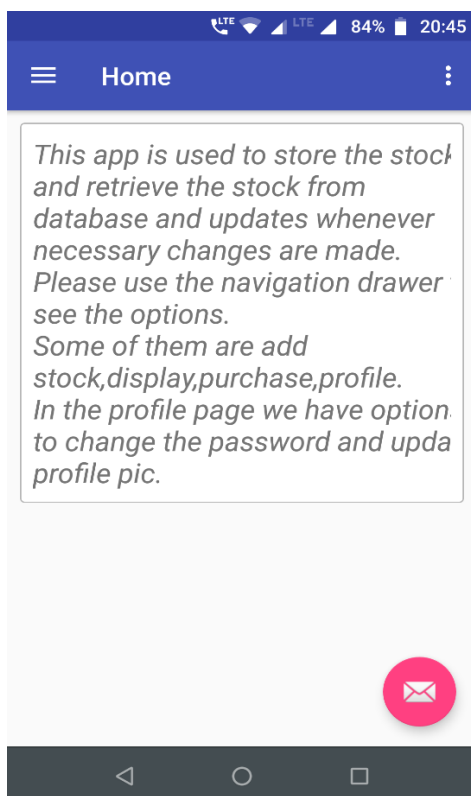
## SIGNUP FORM



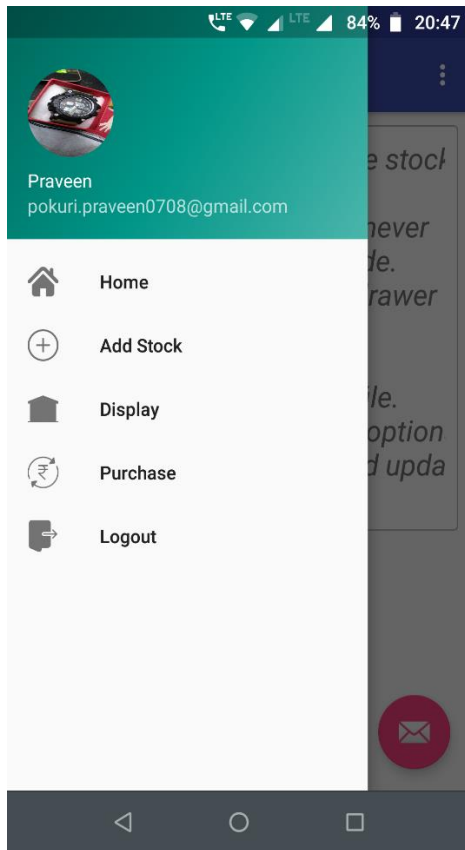
The screenshot shows a mobile application interface for signing up. At the top, there's a status bar with LTE, signal strength, 85% battery, and 20:20 time. Below that, a blue header bar contains the text "SIGN UP". The main content area is white with rounded corners. It features three input fields: "NAME" with a person icon, "EMAIL" with an envelope icon, and "PASSWORD" with a lock icon. A blue circular button with a white right-pointing arrow is positioned to the right of the password field. Below these fields, there's a section titled "Already have an Account" with a blue button labeled "SIGN IN". The bottom of the screen shows a dark grey navigation bar with standard Android icons.

**Fig:5.3.1.2** The signup form requires name, email, password (minimum 8 characters).

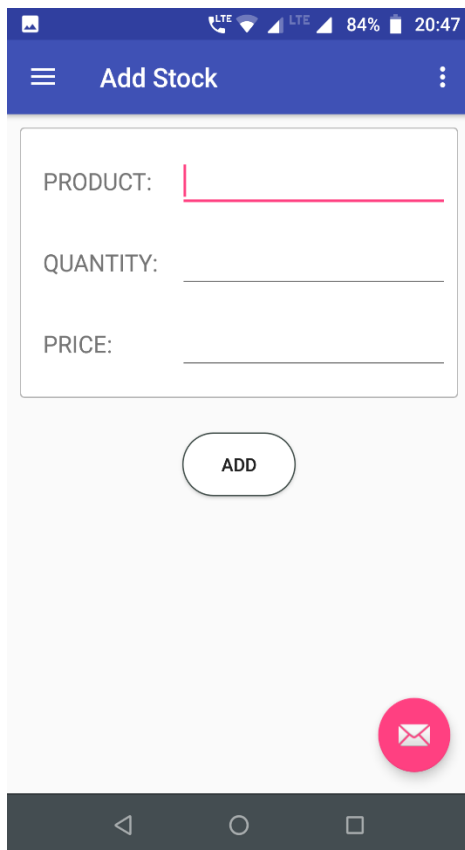
## 5.3.2 Output Screens



**Fig:5.3.2.1** The homepage includes all the features of the application.

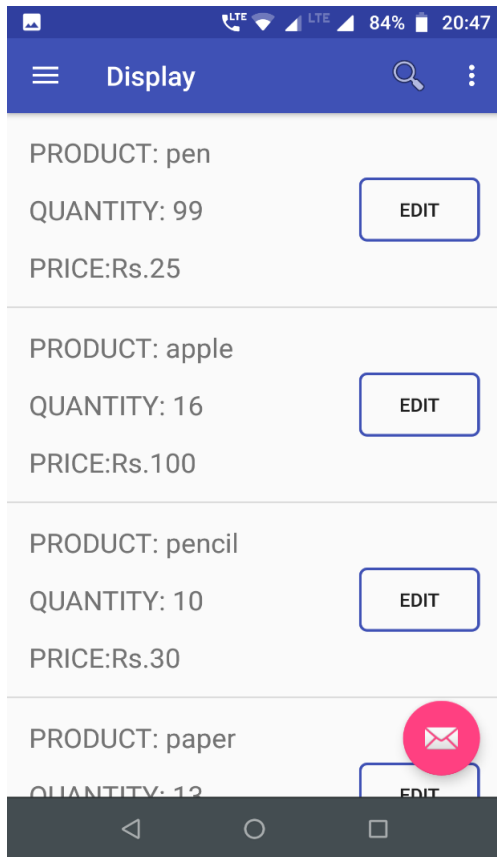


**Fig:5.3.2.2** The navigation drawer consists of 5 options to navigate into different screens of the user's choice.

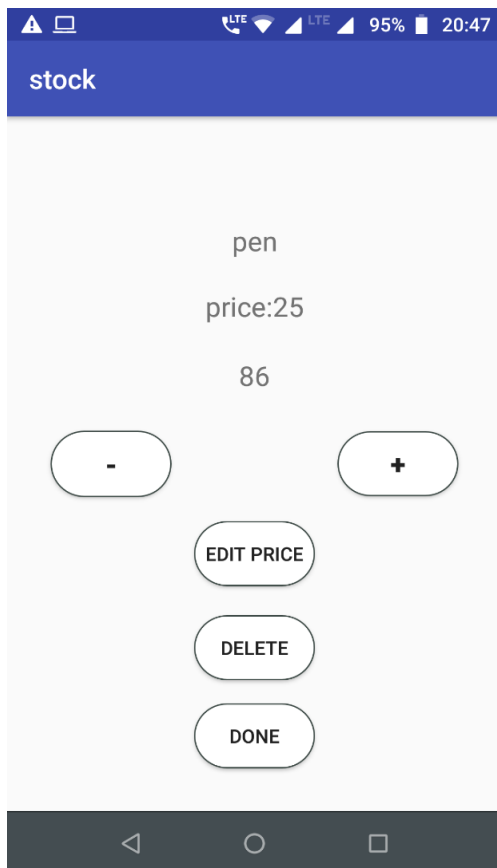


**Fig:5.3.2.3** To add the stock the user has to enter the product name, quantity, price and click on the add button. The user cannot leave any field empty.

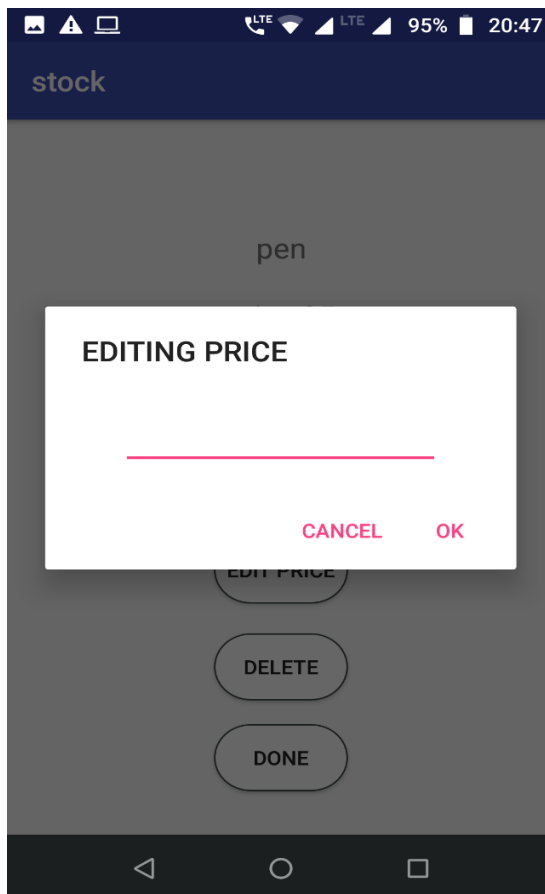




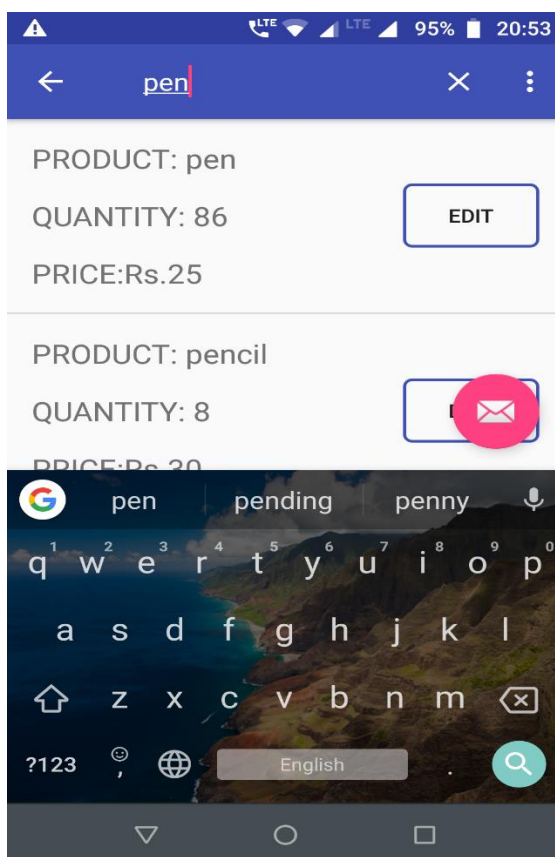
**Fig:5.3.2.4** In this page the user can view all the stocks and can edit any stock manually by clicking edit button. There is also a search option on the top right corner.



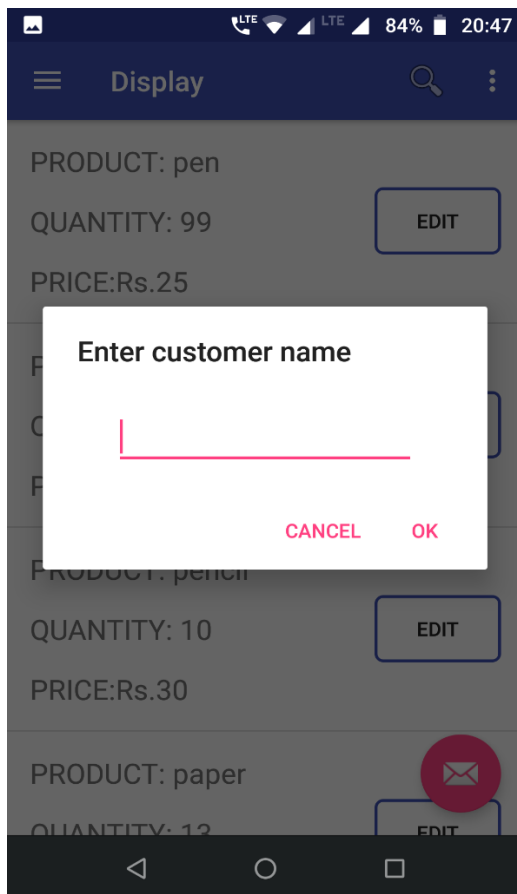
**Fig:5.3.2.5** When the user clicks on the edit button he can manually edit the stock by clicking on '-' or '+' buttons. User can also delete the stock and edit the price of the stock.



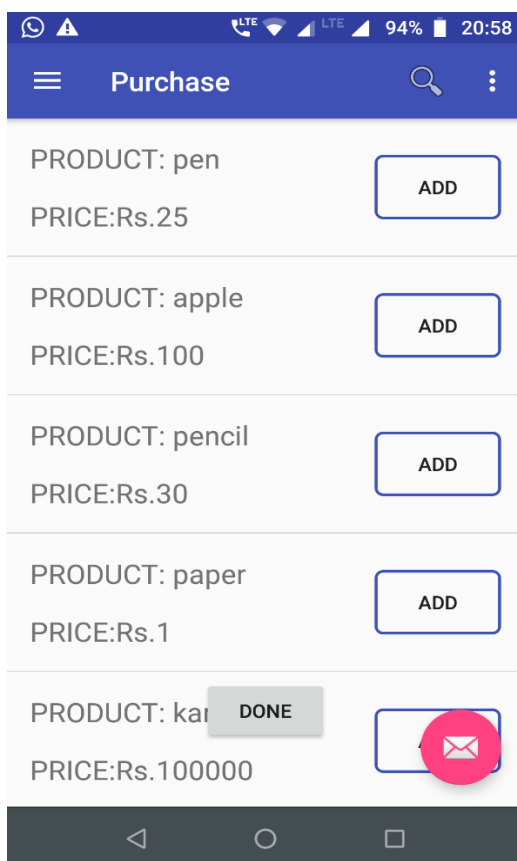
**Fig:5.3.2.6** When the user clicks on the edit price button a pop up box is opened and the user has to enter the new price and click on ok button.



**Fig:5.3.2.7** The search option is on the top right corner and only the related stocks are displayed on the screen.



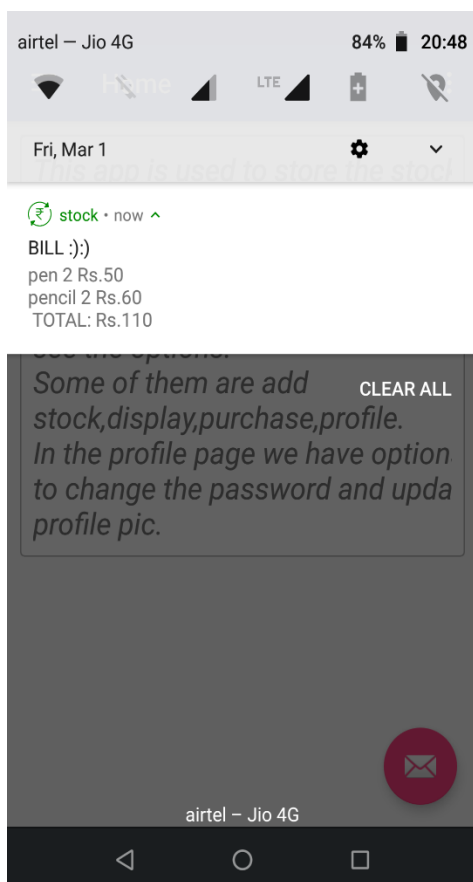
**Fig:5.3.2.8** When the user clicks on purchase in the navigation drawer he has to enter the customer name for billing.



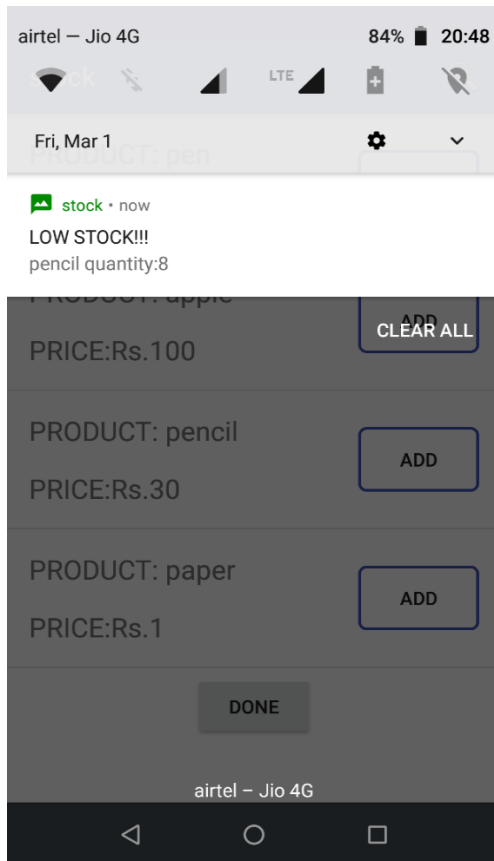
**Fig:5.3.2.9** The shopkeeper has to select the required products and click on done button.



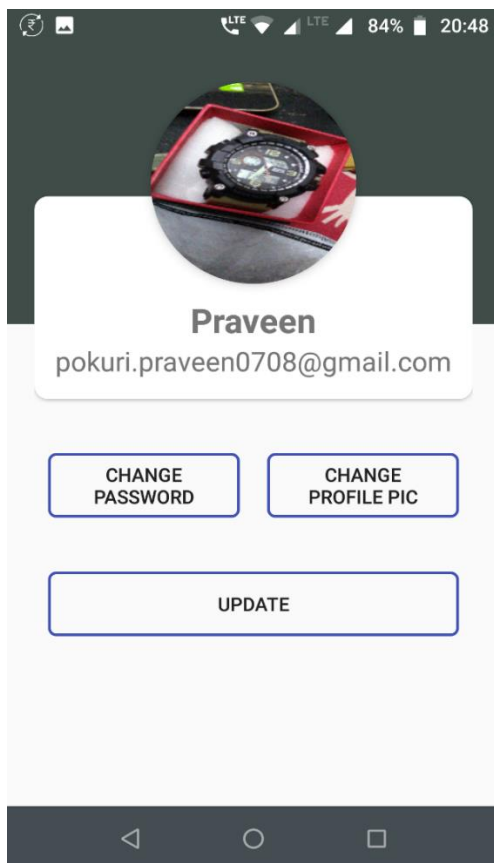
**Fig:5.3.2.10** After clicking the done button the total bill is displayed and we can click on the back button to add more products.



**Fig:5.3.2.11** The bill is displayed as a notification.



**Fig:5.3.2.12** When a product is low in stock a notification is displayed to warn the shopkeeper.



**Fig:5.3.2.13** In the profile page the name and email of the user is displayed. The user can change his password and profile picture.

## ***Chapter 6***

# **TESTING AND VALIDATION**

## **6.1 Introduction**

Testing is a process of executing a program with the aim of finding error. To make our software perform well it should be error free. If testing is done successfully it will remove all the errors from the software.

Software testing helps in finalizing the software application or product against business and user requirements. It is very important to have good test coverage in order to test the software application completely and make it sure that it's performing well and as per the specifications.

While determining the test coverage the test cases should be designed well with maximum possibilities of finding the errors or bugs. The test cases should be very effective. This objective can be measured by the number of defects reported per test cases. Higher the number of the defects reported the more effective are the test cases.

Once the delivery is made to the end users or the customers, they should be able to operate it without any complaints. In order to make this happen the tester should know as how the customers are going to use this product and accordingly, they should write down the test scenarios and design the test cases. This will help a lot in fulfilling all the customer's requirements.

### **6.1.1 Types of Testing**

#### **1) Unit testing:**

A Unit is a smallest testable portion of system or application which can be compiled, linked, loaded, and executed. This kind of testing helps to test each module separately.

The aim is to test each part of the software by separating it. It checks that components are fulfilling functionalities or not. This kind of testing is performed by developers.

#### **2) Integration testing:**

Integration means combining. For Example, In this testing phase, different software modules are combined and tested as a group to make sure that integrated system is ready for system testing.

Integrating testing checks the data flow from one module to other modules. This kind of testing is performed by testers.

#### **3) System testing:**

System testing is performed on a complete, integrated system. It allows checking system's compliance as per the requirements. It tests the overall interaction of components. It involves load, performance, reliability and security testing.

System testing is most often the final test to verify that the system meets the specification. It evaluates both functional and non-functional needs for the testing.

#### **4) Acceptance testing:**

Acceptance testing is a test conducted to find if the requirements of a specification or contract are met as per its delivery. Acceptance testing

is basically done by the user or customer. However, other stockholders can be involved in this process.

#### 5) **Alpha testing:**

This is a type of validation testing. It is a type of *acceptance testing* which is done before the product is released to customers. It is typically done by QA people.

#### 6) **Beta testing:**

The beta test is conducted at one or more customer sites by the end-user of the software. This version is released for the limited number of users for testing in real time environment.

## 6.2 Design of test cases and scenarios

S.No	INPUT	EXPECTED OUTPUT	OBSERVED OUTPUT	TEST CASE
1.	Enter email and wrong password	Invalid credentials	Login failed	Pass
2.	Register with invalid email	Verification not possible	Verification not possible	Pass
3.	Register with password length less than 8	Registration failed	Registration failed	Pass
4.	Leave some fields empty.	Display a message "incomplete fields"	Display a message "incomplete fields"	Pass
5.	Enter product, quantity and leave price empty	Display a message "incomplete fields"	Display a message "incomplete fields"	Pass
6.	Enter existing	Display a	Display a	Pass



	product	message “product exists”	message “product exists”	
7.	Search a product in database that doesn't exist	Display a message “not found”	Display a message “not found”	Pass
8.	Manually decrease the stock for notification	Display low stock notification	Display low stock notification	Pass
9.	Add products again after generating bill	Added	Added	Pass

## ***Chapter 7***

### **CONCLUSION & FUTURE WORK**

#### **7.1 Conclusion**

Stock management application has a user-friendly interface and we have tested it with different test cases for different scenarios. This application includes features like adding stock, displaying stock, searching a product in the stock, deleting a stock, updating the price of a stock, billing, displaying the bill as notification, displaying a low stock notification.

#### **7.2 Future work**

- Customers can view their previous purchase history to keep track of their monthly expenses.
- Customers can track their monthly bill and pay it to the shopkeeper at the end of the month.
- Admin can send a notification to all the users indicating the new features in the application.

## ***Chapter 8***

### **REFERENCES**

#### **8.1 References**

**Android studio documentation:**

<https://developer.android.com/docs/>

**Firebase documentation:**

<https://firebase.google.com/docs/>

**StackOverFlow:**

<https://stackoverflow.com/questions/37094631/get-the-pushed-id-for-specific-value-in-firebase-android>

<https://stackoverflow.com/questions/39702304/retrieve-stored-image-from-firebase-storage>