



Shri Vile Parle Kelavani Mandal's
**NARSEE MONJEE COLLEGE
OF COMMERCE AND ECONOMICS**
(Autonomous) NAAC Accredited 'A' Grade



ADD TO MY CART
(Mobile Application)

A Project Report

Submitted in partial fulfilment of the
Requirements for the award of the Degree of

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

By

Anishma Kathpal
45207180016

Under the esteemed guidance of

Mr. Gufran Qureshi
Assistant Professor

DEPARTMENT OF INFORMATION TECHNOLOGY
NARSEE MONJEE COLLEGE OF COMMERCE AND ECONOMICS
(Autonomous)

Abstract

“Add to My Cart” is a mobile application designed to emphasize the digital payment system. Due to the current situation in the market, this application is developed for contactless digital payment, reduce the long queues as well as the staff required at the supermarkets and shopping centres.

The project aims to create an application that avails users to add products to the cart by scanning the barcode of the respective product and adding it to the cart. Barcode contains information about a product like the price & weight of the product, date of manufacturing and expiry, name of the manufacturer, etc. Barcode is allocated by an international institution set up for this purpose. Every product has a unique barcode all over the world.

The achievement of this project is to eliminate the use of paper required to print receipts of the products purchased and hence encourages the use of digital payment.



TABLE OF CONTENTS

Contents

Chapter 1: Introduction	9
1.1 Background:	9
1.2 Objectives:.....	10
1.3 Purpose, Scope & Applicability:.....	11
1.3.1 Purpose:	11
1.3.2 Scope:.....	11
1.3.3 Applicability:.....	11
1.4 Achievements:.....	12
1.5 Organization of Report:	13
Chapter 2: Survey of Technologies	15
2.1.1 Android Studio VS Apache Cordova	15
2.1.2 SQLite VS MongoDB	18
2.2 Description of Chosen Technology	19
Chapter 3: Requirements and Analysis	21
3.1 Problem Definition	21
3.2 Requirements Specification	22
3.2.1 Existing System.....	22
3.2.2 Proposed System.....	23
3.2.3 System Requirements	24
3.3 Planning and Scheduling	26
3.4 Software and Hardware Requirements.....	28
3.4.1 Hardware Requirements	28
3.4.2 Software Requirements	28
3.5 Preliminary Product Description	28
3.6 Conceptual Models	29
3.6.1 Entity-Relationship Diagram	29
Chapter 4: System Design	30
4.1 Basic Module.....	30
4.2 Data Design.....	31
4.2.1 Schema Design	31
4.3 Procedural Diagram	33
4.3.1 Logical Diagram	33
4.3.2 Use Case Diagram	34
4.3.3 Sequence Diagram	35
4.4 User Interface Design	38
4.4.1 Home Page	38

4.4.2 Barcode Scanner	39
4.4.3 Product Details	40
4.4.4 Cart	41
4.4.5 Payment	42
4.5 Security Issues	43
4.5.1 Credit Card Fraud	43
4.5.2 Malware	43
4.5.3 Man-in-the-middle Attacks	43
4.5.4 Denial of Service Attacks (DOS)	44
4.5.5 Domain name service (DNS) attack	44
4.6 Test Cases	45
Chapter 5: Implementation and Testing.....	47
5.1 Implementation Approaches	47
5.1.1 Class Diagram	50
5.1.2 Database Design	51
5.1.3 Phase Table	52
5.1.4 Gantt Chart	54
5.2 Coding Details & Code Efficiency	56
5.2.1 Code Efficiency	56
5.2.2 COCOMO Model	78
5.3 Testing Approach	80
5.3.1 Unit Testing	80
5.3.2 Integrated Testing	83
5.3.3 Beta Testing	86
5.4 Modifications & Improvements	87
5.5 Test Cases	89
Chapter 6: Results & Discussion	90
6.1 Test Reports	90
6.2 User Documentation	94
Chapter 7: Conclusions	100
7.1 Conclusion	100
7.1.1. Significance of the system	100
7.2 Limitations of the System	101
7.3 Future Scope	101

Chapter 1: Introduction

1.1 Background:

“Add to my cart” uses the technology of barcode scanning to add products to the cart. Barcodes eliminate the possibility of human error. Errors in a physical count can result in excess inventory, out-of-stock items, incorrect orders, and profit loss. Many grocers are now deploying app-enabled mobile device cameras to scan inventory. In an industry where minor improvements in inventory management can make a significant difference in the bottom line, mobile barcode scanning technology offers excellent value. Enabling workers to perform physical counts more efficiently can also reduce labour costs or allow the retailer to deploy those workers to perform high-value tasks.

Smartphones also enable grocers the ability to establish a bring-your-own-device (BYOD) program where employees can use their own mobile devices for inventory or scanning capabilities. Because purpose-built scanners are much costlier than smartphones and tablets, retailers can replace or supplement one of them with several mobile devices.

From online shopping to instant paying and beyond, the world of mobile is single-handedly transforming the way that we research and purchase goods. A barcode scanner gathers product information by scanning a striped code usually located on the back of a product. Using an iPhone or Android device, a scanner app reads the barcode image pulling information like product name, price, Stock Keeping Unit (SKU), etc. The information you gather from scanning a product depends on the available information in the local database of the service used.

1.2 Objectives:

The primary objective of the project is to encourage cashless payments. The application will avail features of digital payment options like credit cards, debit cards, e-wallets, UPI, coupon codes, and more.

The secondary objective of the project is to avoid long queues at the billing counter and encourage contactless shopping (keeping the current COVID-19 scenarios in the notice). The user will scan the barcode of the product, add it to their virtual cart, and, before proceeding to the exit the user must make the digital payment through the app or cash payment at the counter. To avoid frauds/robberies, the final check-out bill shall be scanned at the exit, which would contain a sensor to detect the products at the time of check out.

The few other objectives to design an application like add to my cart is to have digital records and billings so that it is easier to keep an account of the finances and inventories. It would also provide the user with details and additional information about the product. The user can keep a record of the products purchased and can create a Wishlist for the next time.

1.3 Purpose, Scope & Applicability:

1.3.1 Purpose:

The main purpose of the Smart Shopping System is to provide a provision to improve the shopping services for the users through a fast, timely and, convenient system. The Smart Shopping Application was designed keeping the COVID-19 situation in mind as due to the COVID-19 lockdown imposed, there was a boost in the e-commerce sector due to the effectiveness of no contact shopping and to implement the same features for offline shopping.

This system will encourage a digital payment system and ensure a proper account of the inventories for the staff through its e-billing feature. A smart shopping system will help the user to have accurate information and details about the products. It would also work as a future Wish-list for the user to enable them with and comfortable shopping experience.

1.3.2 Scope:

The scope of the application is of two main roles: Admin and User.

The admin will have control of the application which will allow them to

- Keep account of the inventories
- Modify the details and availability of the products
- Have the record of all the transactions
- Have the record of the users and their detail
- The User will be able to scan the barcode of the product for more details
- Add their product in the Cart or Wishlist
- Choose their mode of payment (Digital/Cash)

1.3.3 Applicability:

The smart shopping system shall reduce the time spent in long queues at the payment & billing counter. It shall also enable the user to pay from their preferred mode of payment (Digital Transaction or Cash Payment). The back end of this system resides the database that provides data regarding the availability of stock and gets auto deducted on purchases of the product.

1.4 Achievements:

The smart shopping system provides a duo (Online & Offline) shopping experience that allows users to scan the barcode of a product from any category in the supermarket. It will display the information and availability of the product. This application will help in maintaining digital copies of the bills and the transaction for easy accountings. On proceeding to the exit, a staff member would confirm the payment and the products purchased, to avoid fraud and to maintain accuracy. Through the process of user log-in and authentication, the payment details would be safe and secured. As the application is simple and compact, it enables all to use it with ease and comfort and have a time-saving shopping experience.

1.5 Organization of Report:

This chapter has focused on a brief description of the background and context of the project and its relation to work already done in the area. It has given a concise statement of the aims and objectives of the project, answers questions on why this project is being done, how the project could improve the system its significance and theoretical framework, what are the key issues being covered in the project, what are the main functions of the project, and how this project will serve the computer world and people. This chapter will also explain what knowledge the student has achieved after the completion of the work, what contributions has the project made to the chosen area. It will also state the goals achieved describing the degree to which the findings support the original objectives laid out by the project.

In the 2nd chapter, Survey of Technologies, I will highlight my collective awareness and understanding of available technologies related to the topic of the project. I will also give the detail of all the related technologies that are necessary to complete the project and are available in the chosen area. I will also present a comparative study of all those available technologies and explain why selected the one technology for the completion of the objectives of the project.

The 3rd chapter, Requirements and Analysis, will define the problem on which I am working and will provide details of the overall problem in the project. In this phase, I will also define the requirements of the system, independent of how these requirements will be accomplished, describe the things in the system and the actions that can be done on these things, and identify the operation and problems of the existing system. The way I have planned to complete the project will be shown with the help of the Gantt Chart. This will describe how I have scheduled my whole project completion. It will also give the details of all the software and hardware needed for the development and implementation of the project. Lastly, I will explain the problem domain and produce a model of the system, which describes operations that can be performed on the system, and the allowable sequences of those operations.

The 4th chapter of the project will describe the desired features and operations in detail, including screen layouts, business rules, process diagrams, and pseudocode. I will divide

the overall problem into more manageable parts and develop each part or module separately. When all modules are ready, I will integrate all the modules into one system. In this phase, I will briefly describe all the modules and the functionality of these modules. In the user interface, I will describe the external and internal components and the architecture of the user interface. I will also include real-time considerations and security issues related to the project and explain how I intend to avoid those security problems. The test cases will provide easy detection of errors and mistakes within a minimum period and with the least effort.

The 5th chapter, Implementation and Testing, will define the plan of implementation, and the standards that I would be using in the implementation. The code will include only the important codes (algorithms, applets code, forms code, etc.). It will also include the testing approaches, modifications, and improvements that I have done to my project over time.

The 6th chapter, Results and Discussion, will explain the test results and reports based on the test cases, which should show that the project can face any problematic situation and that it works fine in different conditions. The results will be discussed with different scenarios which will in turn help to determine the success rate of the project.

The 7th chapter, Conclusions, brings together the limitations encountered during the testing of the project and how they can be eliminated in the future. It also enlists the criticisms accepted during the demonstrations of the project. It will also include the future scope of the project describing two things: firstly, new areas of the investigation prompted by developments in this project, and secondly, parts of the current work that were not completed due to time constraints and/or problems encountered.

Chapter 2: Survey of Technologies

2.1.1 Android Studio VS Apache Cordova

(Table 2.1.1: Comparison of Technologies: Android Studio VS Apache Cordova)

About	
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 or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE (Integrated Development Environment) for native Android application development.	Apache Cordova (formerly PhoneGap) is a mobile application development framework created by Nitobi. Adobe Systems purchased Nitobi in 2011, rebranded it as PhoneGap, and later released an open-source version of the software called Apache Cordova. Apache Cordova enables software programmers to build hybrid web applications for mobile devices using CSS3, HTML5, and JavaScript, instead of relying on platform-specific APIs like those in Android, iOS, or Windows Phone. It enables the wrapping up of CSS, HTML, and JavaScript code depending upon the platform of the device. It extends the features of HTML and JavaScript to work with the device. The resulting applications are hybrid, meaning that they are neither truly native mobile application (because all layout rendering is done via Web views instead of the platform's native UI framework) nor purely Web-based (because they are not just Web apps, but are packaged as apps for distribution and have access to native device APIs). Mixing native and hybrid code snippets have been possible since version 1.9.
Features	
It is useful in the following scenarios: 1. For Android development when you use Java as your front-end language. 2. For Android development and Kotlin development when you use Kotlin as an alternative to Java as your front-end language. 3. For Android development when you use Flutter and Dart as in your front-end development. 4. For Java development.	It is suitable for making portable applications, with the same code for several platforms. You can access the native features of the device or use an open-source plug-in from the repository to create a local database and access the internal storage of the device. It is wonderful for the construction of a native application, using standard web code. It is not recommended for enterprise applications.

Pros	
<ul style="list-style-type: none"> • Supports Gradle instead of Maven. • Debug mode is excellent. • Android Studio is not a unique product for Android development. IntelliJ IDEA is good for it too. • It is open-source software. • No need to pay, free of charge product. • It supports a big community of other android developers. • Provide Java to Kotlin code auto-translation. 	<ul style="list-style-type: none"> • Cordova makes it extremely easy to develop apps for multiple platforms. The setup is quite simple when it comes to creating a project, adding platforms, building, and deploying apps. If you have a little mobile app development experience, all you need to know is HTML, CSS, JavaScript and only a handful of Cordova commands to get started with your hybrid app. • Cordova provides a simple solution to access all the device features through native plugins. You have a host of third-party and Cordova plugins are available to use device features like a filesystem, camera, health kit, location services, etc. You can also write your very own plugins and use them for your Cordova-based apps. • Cordova is free to use! The only cost you will bear is the individual mobile platform developer program enrolment cost to deploy your apps to those platforms.
Cons	
<ul style="list-style-type: none"> • Unlike Eclipse, Android Studio has no concept of a workspace. Each window houses only one project. It is not easy to jump between projects. • Android Studio is not lightweight. It consumes lots of memory and takes lots of time to perform certain tasks. • I frequently see ignorable messages telling me that Android Studio has encountered an error (an error in the IDE, not an error in my code). I have never bothered to find the source of these messages because the messages go away quickly, and they do not keep me from running my code. 	<ul style="list-style-type: none"> • Needs to be fully compatible with mobile machines. • Support for a wide variety of platforms. • Needs better backwards compatibility.
Returns on Investment	
<ul style="list-style-type: none"> • Due to Android Studio's day by day improvement, our company is making Android applications in increasingly effective and efficient ways. • 24x7 support from Google and JetBrains is making our work running 24x7 smoother and making our clients happier and happier day by day. 	<ul style="list-style-type: none"> • It has a positive impact in general. Cordova is really a great solution for web developers who want to bring their incredible ideas to devices, but they just do not have a lot of time to put into iOS and Android learning curves. • Our biggest benefit was that the management of images for multiple devices. • Developing with Cordova has drastically reduced the cost of cross-platform deployment.

- Due to the awesome animation and transition tricks, we are providing our clients more than they expected.

System Requirements (Windows)

- | | |
|--|--|
| <ul style="list-style-type: none">• Microsoft® Windows® 7/8/10 (64-bit)• 4 GB RAM minimum, 8 GB RAM recommended• 2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)• 1280 x 800 minimum screen resolution | <ul style="list-style-type: none">• A Windows 8.1, 32 or 64-bit machine (<i>Home, Pro, or Enterprise</i> editions) with minimum 4 GB of RAM along with Visual Studio 2015 or Visual Studio 2013. An evaluation version of Windows 8.1 Enterprise is available from the Microsoft Developer Network.• For the Windows Phone emulators, Windows 8.1 (x64) Professional edition or higher, and a processor that supports Client Hyper-V and Second Level Address Translation (SLAT). |
|--|--|

2.1.2 SQLite VS MongoDB

(Table 2.1.2: Comparison of Technologies: SQLite VS MongoDB)

SQLite	MongoDB
Developed	
Developed by D. Richard Hipp in August 2000.	Developed by MongoDB, Inc in 2009.
Usage	
It is widely used in-process RDBMS.	It is one of the most popular document stores available both as a fully managed cloud service and for deployment on self-managed infrastructure.
Primary Database Model	
The primary database model for SQLite is Relational DBMS.	The primary database model for MongoDB is a Document store.
Secondary Database Model	
It has no Secondary database models.	It has a Document store as Secondary database models.
Server	
SQLite does not require a server to run. Hence, it is serverless.	Server operating systems for MongoDB are Linux, OS X, Solaris, and Windows.
Query Language	
It supports SQL query language only.	It supports JSON query language along with SQL.

2.2 Description of Chosen Technology

1. Android Studio (Kotlin Language)

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 a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development.

Android Studio provides a unified environment where you can build apps for Android phones, tablets, Android Wear, Android TV, and Android Auto. Structured code modules allow you to divide your project into units of functionality that you can independently build, test, and debug.

Kotlin is a general-purpose, free, open-source, statically typed “pragmatic” programming language initially designed for the JVM (Java Virtual Machine) and Android that combines object-oriented and functional programming features. It is focused on interoperability, safety, clarity, and tooling support.

2. SQLite

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures. These features make SQLite a popular choice as an Application File Format.

It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others.

Features of SQLite:

- SQLite lets you store data in a structured manner.
- SQLite has higher performance.
- SQLite databases can also be queried, and the data retrieval is much more robust.
- The `android.database` and `android.database.sqlite` packages offer a higher-performance alternative where source compatibility is not an issue.
- Android databases created in Android are visible only to the application that created them

- There is no file parsing and generating code to write and debug.
- Content can be accessed and updated using powerful SQL queries, reducing the complexity of the application code.
- Extending the file format for new capabilities in later releases is as simple as adding new tables or new columns to existing tables.
- Diverse content which might otherwise be stored as a "pile of files" can be encapsulated into a single disk file.

Chapter 3: Requirements and Analysis

3.1 Problem Definition

Traditional shopping is a method of buying a product by going to a store. It is a way like we go to our favourite store in a nearby mall. In traditional shopping, we can choose a product physically and check out what the product is like, how it looks, and the features it has. Therefore, some consumers still prefer the traditional type of shopping over online shopping because, for one, it allows customers to check out an item.

Traditional shopping can be very time-consuming if you have not made the decision of what to buy.

With everything that has been going on with the Coronavirus, the World Health Organization is encouraging you to use as many digital payment options as possible. We know that money changes hands frequently and can pick up all sorts of bacteria and viruses.

If you use cash, you miss a lot of good credit card rewards. Whether it is free to travel, cashback, extended warranties, or free rental car insurance, there are a lot of benefits that come with credit cards that you will never get if you opt not to use them.

One of the most important things in budgeting is to track your expenses, so you can see where your money is going and to have a record of your purchases. Sure, you can collect a pile of receipts. But it is easy to misplace them. It is a mess to categorize them. AND after a while, the ink on thermal receipts fades, leaving a blank paper of confusion. Instead, when tracking your expenses in a budget, electronic records reign supreme.

3.2 Requirements Specification

3.2.1 Existing System

The present scenario for shopping is to visit the shops and market manually and then from the available product list one needs to choose the item he or she wants and then pay for the same item mainly in cash mode is done, as not every society is well educated and aware to use net banking or card modes or wallets etc.

This involves going to the shops and be in a long queue to purchase goods which will consume a lot of time and strength before the customer will search for the goods needed. These results in some problems like:

- Slow in processing customer's information
- Data inconsistency/ Redundancy
- Inaccurate during customers transaction

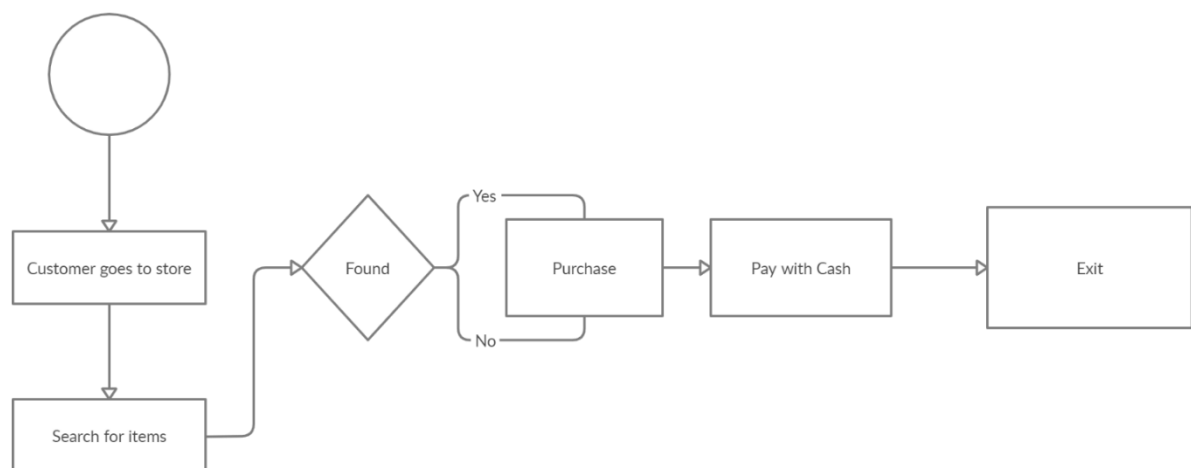


Figure 3.2.1 How Existing System used to work

3.2.2 Proposed System

The proposed new system will make use of files and records in tables prepared using a database to store information about the everyday transaction. Having analysed the existing system, there is a need for an alternative system; the proposed system will eliminate the problems experienced in the existing system. The proposed new system is designed to enhance the following:

- Convenience
- Consistency of Data
- Reliability
- Increases Productivity
- Easy update and Maintenance Operations
- Variety
- Speed Optimization and reduce paperwork

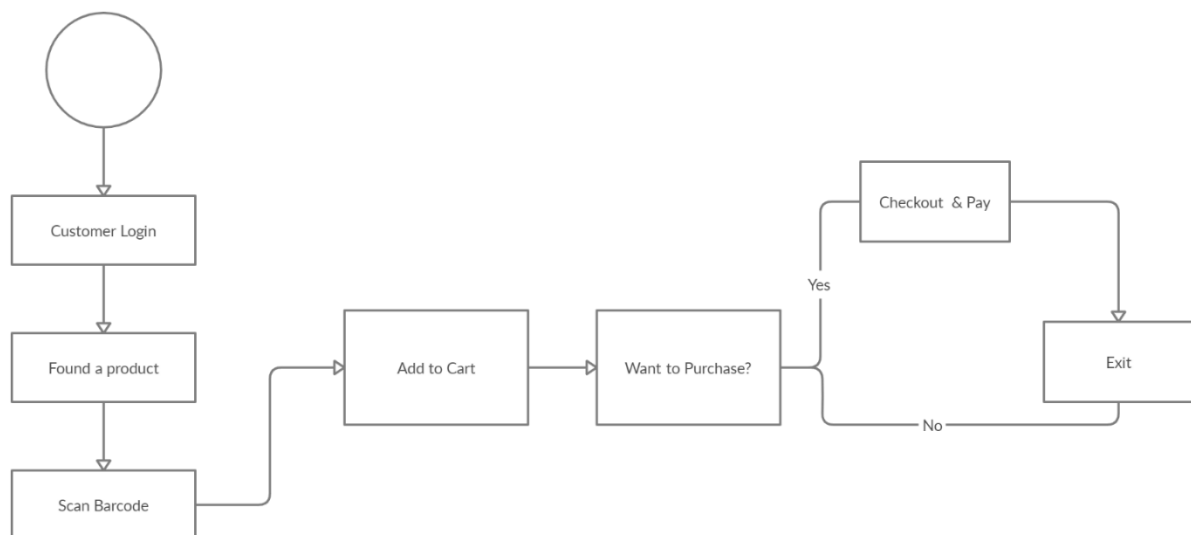


Figure 3.2.2 How Proposed System would work

3.2.3 System Requirements

3.2.3.1 Functional Requirements

The System must provide the following functionalities:

User:

- The users shall be able to view the categories on the application's home page.
- The users shall be able to view items in distinct categories.
- The users shall be able to add items to the cart.
- The users shall be able to view more information about an item before adding it to the cart.
- The users shall be able to view the shopping cart.
- The users shall be able to browse through the available items.
- The users shall be able to view the items added to the cart.
- The users shall be able to check out with the current items in the cart
- The users shall be able to continue shopping.
- The users shall be able to delete items from the cart.
- The users shall be able to check out items only when there are items in the shopping cart.
- The users shall log in or register using the user authentication form.
- The users shall not log in or register if the information is incomplete or invalid.
- The users shall place an order by completing the information in the order form.
- The users shall not be able to place an order if the information in the order form is invalid or incomplete.

Admin:

- The administrator shall be able to view all the users' information that completes the order form and the checkout process.
- The administrator shall be able to modify/update an item's price and description.
- The administrator shall be able to view the entire history of the checked-out items.
- The administrator shall be able to view the entire history for the users who successfully complete the checkout process.
- Keeping records of registration of customers.
- Keeping the records of products.
- Storing the items selected by the customer in the Checklist.
- Full control of the admin panel will be in the admin's hand.

3.2.3.2 Non-Functional Requirements

Following non-functional requirements will be there:

- Secure access of confidential data (customer's details)
- 24X7 availability
- The system must provide data integrity checks to ensure data remains consistent and updated.
- The system should provide a reliable environment to both customers and owners. All orders should be reaching the admin without any errors.

3.3 Planning and Scheduling

A Project Plan is defined as a management summary that describes the essentials of a project in terms of its objectives, justification and how the objectives are to be achieved. It describes how all the major activities under each project management function are to be accomplished, including that of overall project control.

Planning of a system involves the modularization of the project in a definite number of stages and creating a sequence of activities to be performed. The entire project can be viewed as several independent modules. These modules follow a chronological sequence. They are:

1. Preliminary studies and investigation.
2. Data gathering or Systems study.
3. System analysis.
4. Synopsis.
5. Development of Software and Testing.
6. Implementation and Evaluation.

Project scheduling is one of the critical management tasks as it dictates the time frames in which the project will be completed, in terms of resource requirements and the sequence of tasks to be completed. Project scheduling is defined as the process of determining when project activities will take place depending upon defined durations and precedent activities. Schedule constraints specify when an activity should start or end, based on duration, predecessors, external predecessor relationships, resource availability, target dates or other time constraints. Project scheduling is a complex and iterative task.

Scheduling refers to the time needed to conduct systems investigations, formulate the logical design, code the software, and prepare files, develop test data, test the software, and install it. In other words, each activity is associated with the project development requires an amount of time that must be correctly estimated and incorporated into the project schedule.

A Gantt chart is a type of bar chart that illustrates a project schedule. This chart lists the tasks to be performed on the vertical axis, and time intervals on the horizontal axis. The Gantt Chart shows planned and actual progress for several tasks displayed against a horizontal time scale. It is an effective and easy-to-read method of indicating the expected and actual status for each set of tasks compared to planned progress for each activity of the set. Gantt Charts provide a clear picture of the current state of the project.

The figure below shows the expected and actual duration of the first half of the project timeframe.



Figure 3.3.1 Gantt Chart for the first half of the project

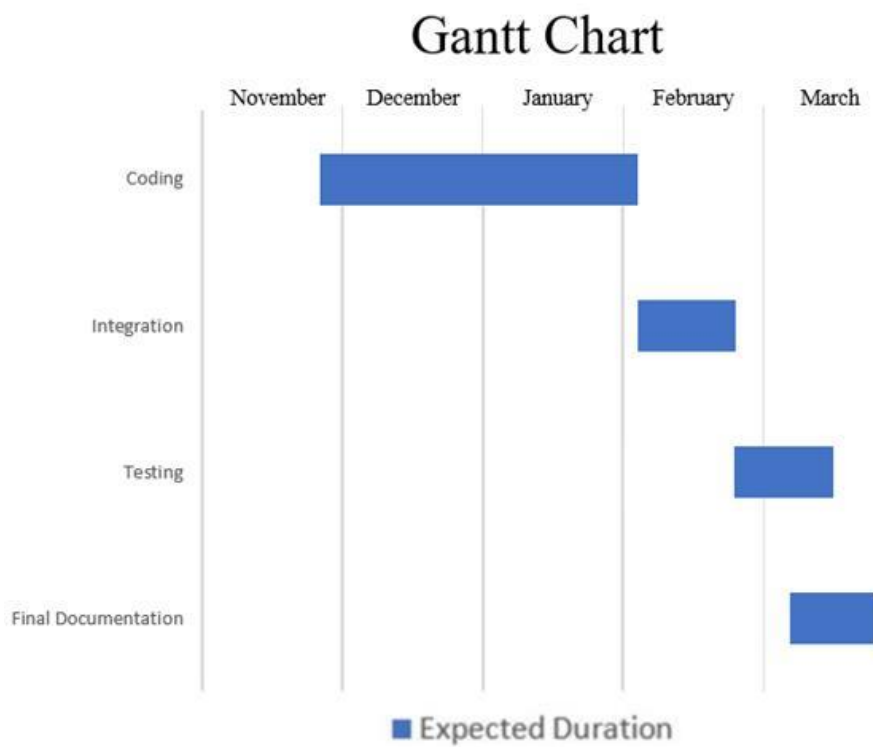


Figure 3.3.2 Gantt chart for the second half of the project

3.4 Software and Hardware Requirements

3.4.1 Hardware Requirements

The software application would be compatible on

- Processor: - Dual Core or above
- RAM: - 512MB or above.
- Hard disk: -80 GB or above

3.4.2 Software Requirements

Following are the software requirements needed for the project:

- Operating system: Windows XP, Window 7 or above
- IDE: Android Studio
- Front end: Kotlin
- Back end: SQLite

3.5 Preliminary Product Description

- **Administrator**

Here, the administrator keeps a record of the user database and manages the database of the inventories.

- **Registration Module**

In this module, the user must register in the application and provide the required credentials.

- **User Module**

In the user module, the user has access to the features like scanning a barcode of the product, making a checklist, adding products to their cart, and paying through the preferred mode.

3.6 Conceptual Models

3.6.1 Entity-Relationship Diagram

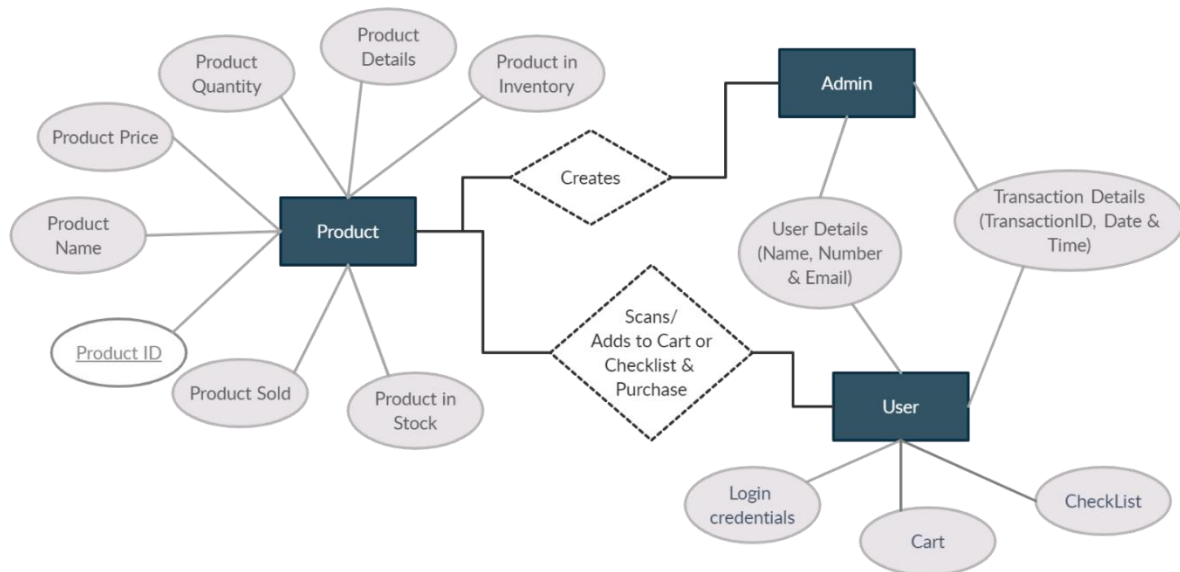


Figure 3.6.1: Entity Relationship Diagram

Chapter 4: System Design

4.1 Basic Module

1. Administrator

In this administrator module the store manager or admin having the authorization to access this module. The store manager or the admin module will have the authority to modify, add or delete the inventories. In this module, the admin will have access to all the transaction records. The administrator will have an account of all the registered users.

2. Registration Module

This is a module through which the user will be registered. The user can register through their Email, Mobile Number, Google Account, or, Facebook with authentication via a unique code or OTP (One-Time Password). To complete the registration, the user must fill in their additional details – name, email address & mobile number.

3. User Module

After registering as a user, the user will be redirected to the user module wherein the user will be able to scan the product, add the product to their cart, add the product to their checklist for later purchase and make the payment for the purchased products.

4.2 Data Design

A database design is a collection of stored data organized in such a way that the data requirements are satisfied by the database. The general goal is to make information access easy, quick, inexpensive, and flexible for the user. There are also some specific goals like controlled redundancy from failure, privacy, security, and performance. A collection of relative records makes up a table. To design and store data to the needed forms database tables are prepared.

4.2.1 Schema Design

4.2.1.1 Admin Table – Inventories

Name	Data Type	Description	Integrity Constraints
productId	int	Unique Code assigned to every product. Detected by scanning the bar code	Primary Key
productName	String	Name of the Product	Not Null
price	double	Cost of the Product	Not Null
quantity(int,string)	Object	Quantity of the product in terms of weight, volume, etc.	Not Null
productDetails	String	Additional Information about the product	Can be Null
inStock	int	Quantity of the Product present in stock (inventory – sold)	Not Null
sold	int	Quantity of the Product sold	Not Null
inventory	int	Total Quantity of Product	Not Null

Table 4.2.1.1 Admin Table – Inventories

4.2.1.2 Admin & User Table - Transaction

Name	Data Type	Description	Integrity Constraints
userId	int	Unique Code assigned to every user	Primary Key
transactionId	long	Unique Code assigned to every transaction	Primary Key
userName	String	Name of the User	Not Null
mobileNo	long	Contact Number of the User	Not Null
emailId	String	Email Address of the User	Not Null
itemTotal	int	Total Amount of the Purchase	Not Null
discount	int	Discount Received on Product Purchase	Can be Null
totalAmt	int	Total Amount Paid by the User on Purchase after Discount	Not Null
time	String	Time of the Purchase	Not Null
date	String	Date of the Purchase	Not Null

Table 4.2.1.2 Admin & User Table – Transaction

4.2.1.3 User Table

Name	Data Type	Description	Integrity Constraints
cart	boolean	Whether the product is in the cart or not	Not Null
checklist	boolean	Whether the product is in the checklist or not	Not Null

Table 4.2.1.3 User Table

4.3 Procedural Diagram

4.3.1 Logical Diagram

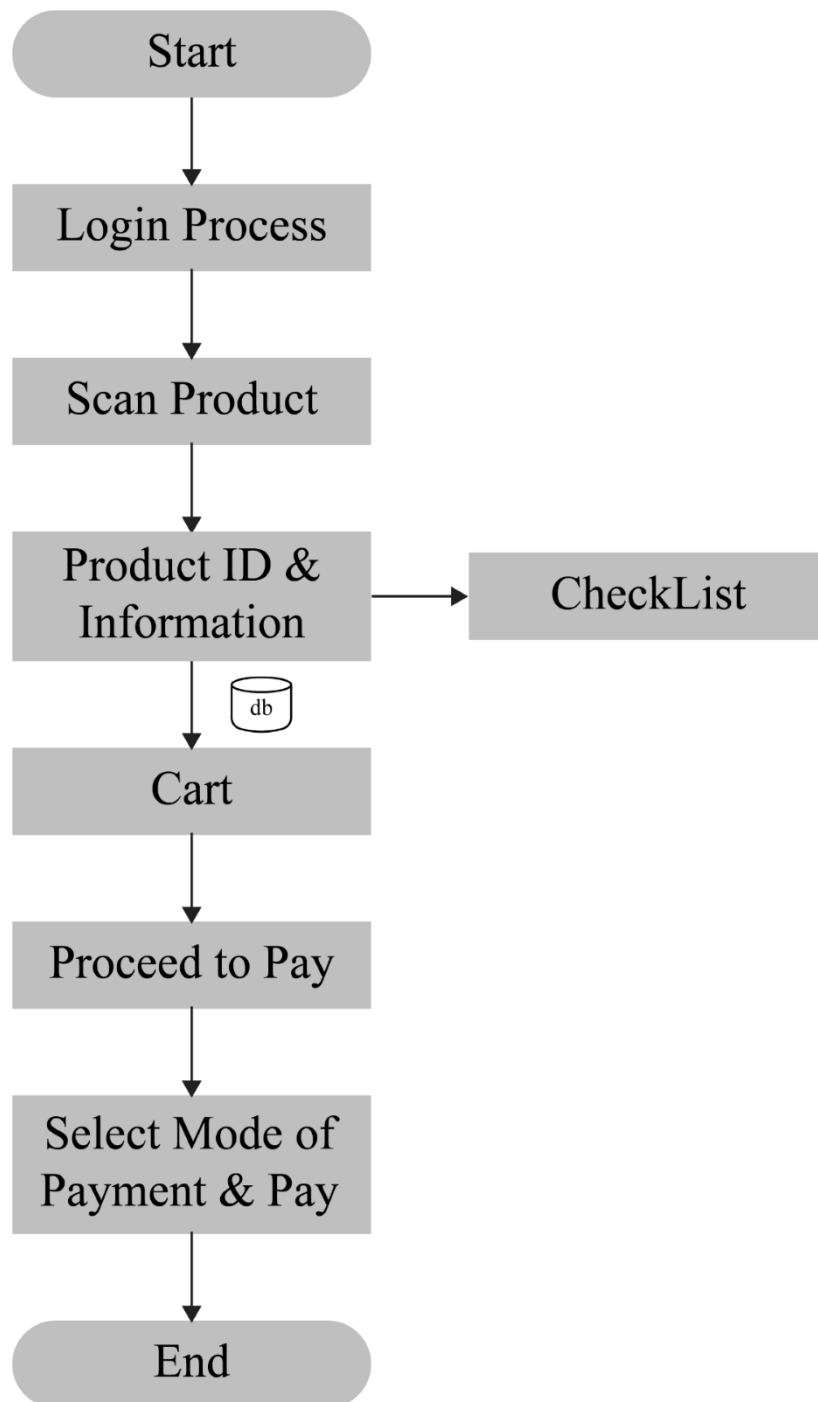


Figure 4.3.1 Logical Diagram

4.3.2 Use Case Diagram

4.3.2.1 Admin Use Case Diagram

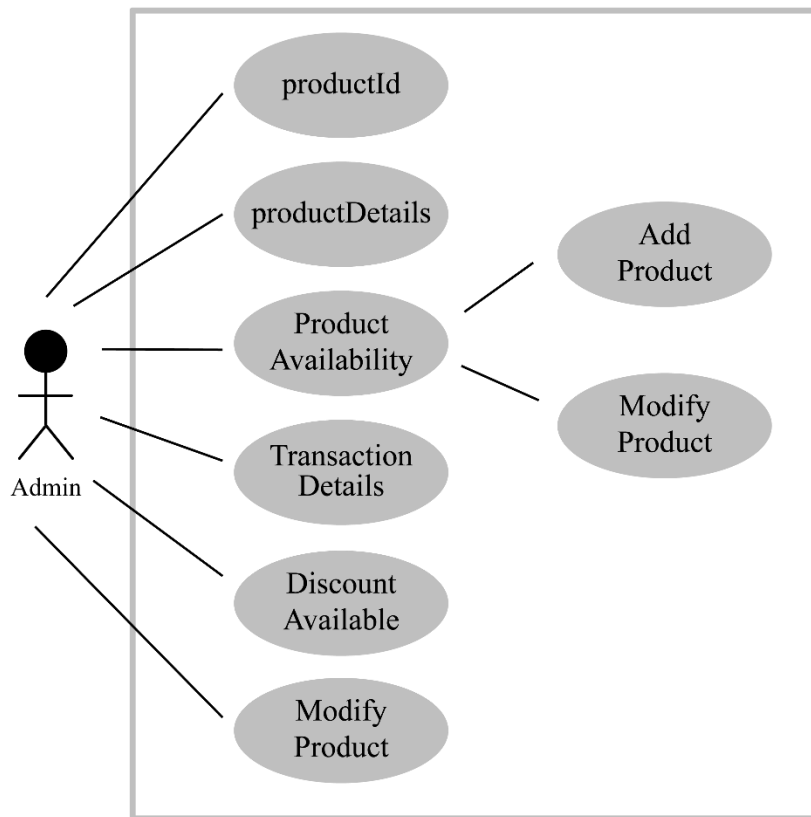


Figure 4.3.2.1 Admin Use Case Diagram

4.3.2.2 User Use Case Diagram

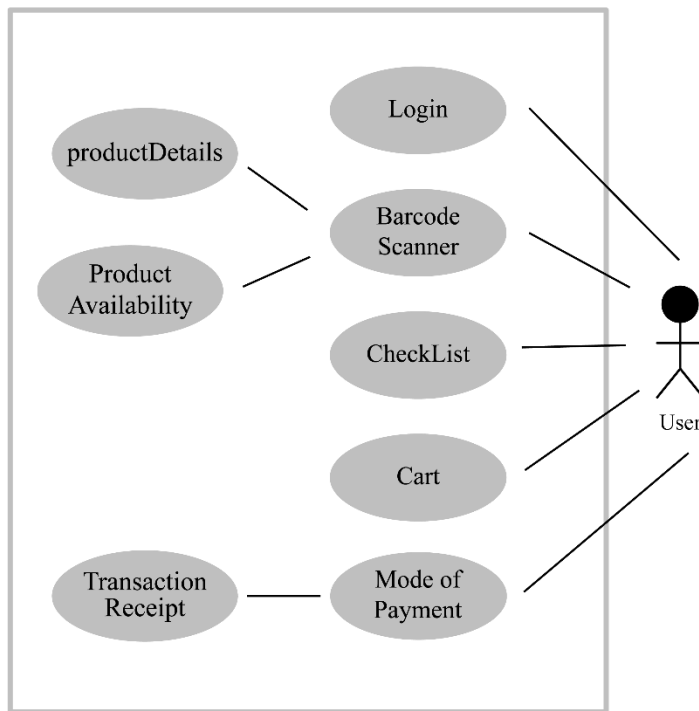


Figure 4.3.2.2 User Use Case Diagram

4.3.3 Sequence Diagram

4.3.3.1 Admin Sequence Diagram

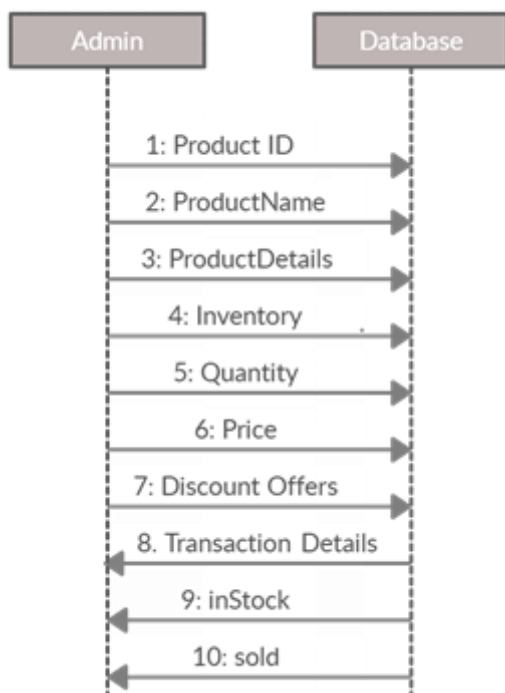


Figure 4.3.3.1 Admin Sequence Diagram

4.3.3.2 User Sequence Diagram

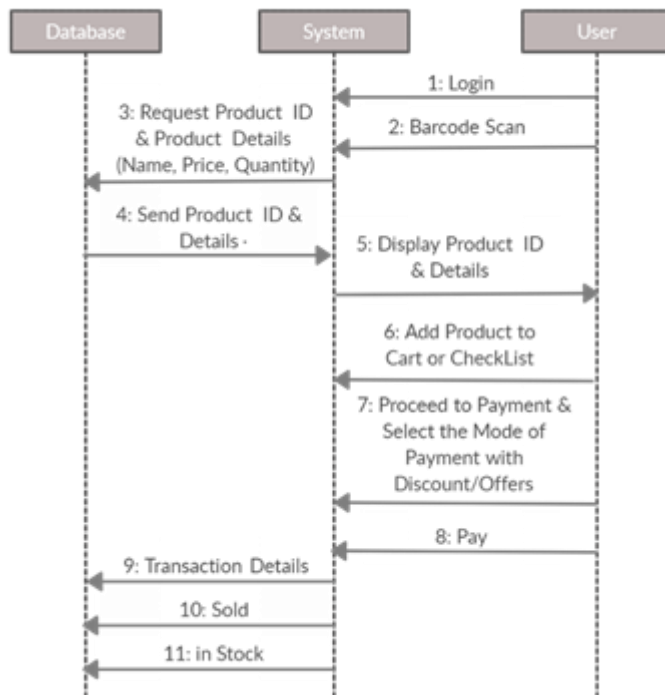


Figure 4.3.3.2 User Sequence Diagram

4.3.4 Algorithm Design

The below algorithm is implemented in the Smart Shopping System.

Step 1: Start

Step 2: The Admin creates a database of all the products available at the store.

Step 3: The user must log in or sign-up to access the smart shopping system. The user must complete the authentication process to proceed further.

Step 4: Once, the user has logged in, the user can scan the product found at the store.

Step 5: On Scanning the barcode, the user would be redirected to the product detail & information page. All the product's barcode scanned by the user would be displayed on the homepage of the application.

Step 6: The user can add the product to the cart to purchase or in the Checklist for future Purchases.

Step 7: Once the user has put all the desired products in the cart, the user must continue with the payment.

Step 8: On the payment pages, the user must choose their mode of payment (digital or cash) and can also check for discounts/offers available on a certain mode of payment or products.

Step 9: On selecting the digital mode of payment, the user would be redirected to the card payment, net banking, UPI, E-wallet, or any mode of the payment page, respectively.

Step 10: In case the user opts for cash payment; the user must show the bill and pay the amount at the exit counter.

Step 11: The admin would be updated with the transaction details, sold products, and the products in stock.

4.4 User Interface Design

4.4.1 Home Page

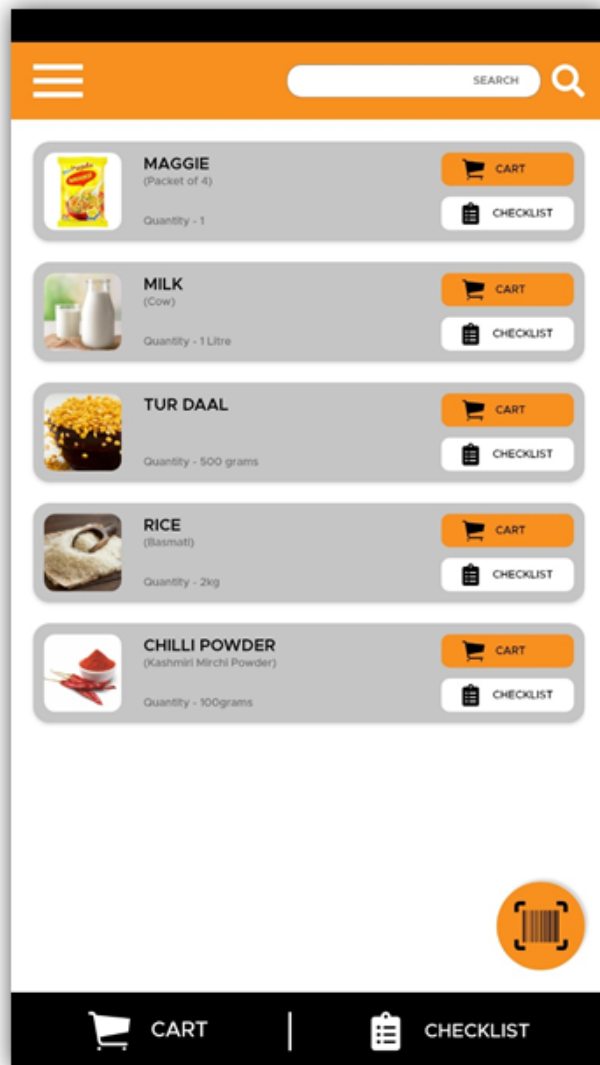


Figure 4.4.1 Home Page

The homepage of the application consists of:

- Option to scan the barcode of the found product
- All the scanned products by the user
- Search – To search the scanned products (useful when there are many products)
- Menu – consisting of user details, payment details, past transactions, notification for offers and more
- Cart – To view which desired items are to be purchased
- Checklist – To save the product for future purchase

4.4.2 Barcode Scanner



Figure 4.4.2 Barcode Scanner

The barcode scanner page allows the user to scan the product barcode using their mobile phone camera. The application would ask the user for permission to use the camera.

4.4.3 Product Details

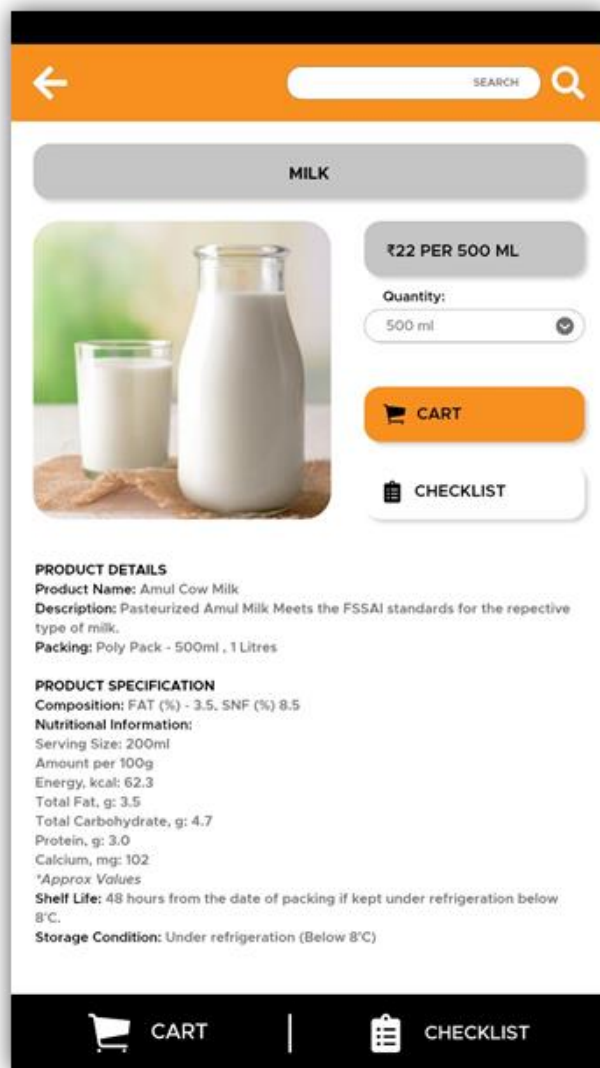


Figure 4.4.3 Product Details

The product detail page displays the information about the scanned product. It allows the user to add the product to the cart or in the Checklist as per the user's requirements. The information of the product is extracted from the database created by the admin.

4.4.4 Cart

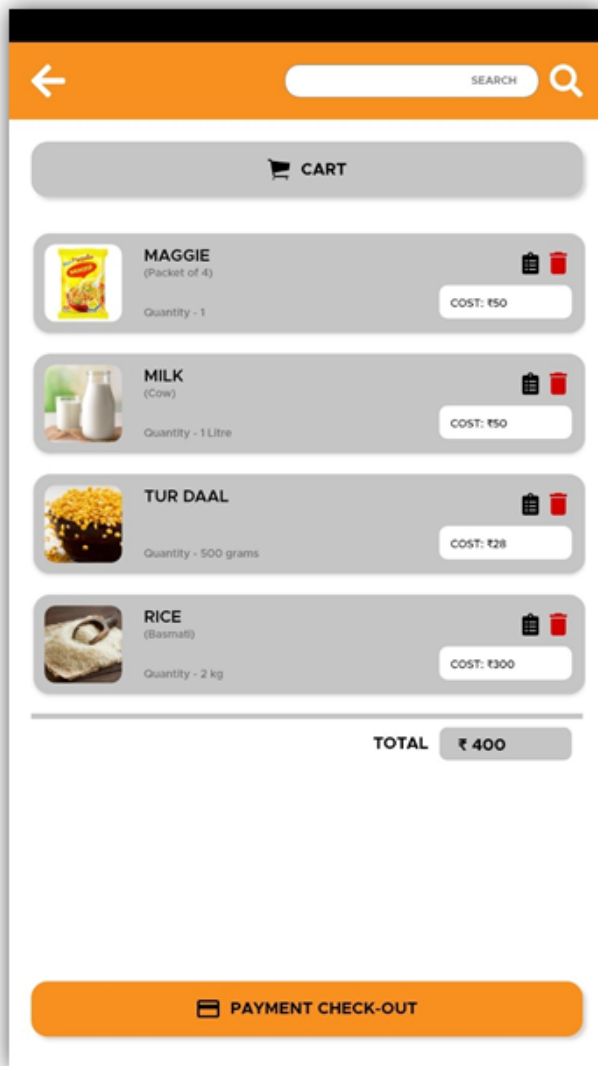


Figure 4.4.4 Cart

The cart page would display the products the user wishes to purchase along with the quantity and its cost. It also displays the total item cost that is to be paid for the purchase. Once the purchases are confirmed by the user, the user can continue to check out/payment.

4.4.5 Payment

The screenshot displays a mobile application's payment screen. At the top, there is an orange header bar with a back arrow on the left, a search bar in the center, and a magnifying glass icon on the right. Below the header, a prominent orange button labeled 'PAYMENT CHECK-OUT' is visible. Underneath this, a list of payment methods is presented as grey buttons: 'NETBANKING', 'UPI', 'E-WALLET', 'CREDIT/ DEBIT CARD', and 'CASH'. A light grey button labeled 'APPLY COUPON CODE/ OFFERS' is positioned below the payment methods. A horizontal line separates this section from a summary table. The table contains three rows: 'ITEM TOTAL' with a value of '₹ 400', 'DISCOUNT (-)' with a value of '₹ 50', and 'GRAND TOTAL' with a value of '₹ 350'. Another horizontal line follows the table. At the bottom, there are two large buttons: a bright green 'PROCEED TO PAY' button and a grey 'EXIT' button.

ITEM TOTAL	₹ 400
DISCOUNT (-)	₹ 50
GRAND TOTAL	₹ 350

Figure 4.4.5 Payment

The payment page allows the user to choose their mode of payment along with the options to select offers available on the product or any payment modes. It displays the total amount of the purchase after the discount. On clicking continue to pay, the user would be redirected to the final payment (bank/e-wallet application).

4.5 Security Issues

Security, as part of the software development process, is an ongoing process involving people and practices and ensures application confidentiality, integrity, and availability. Secure software is the result of security-aware software development processes where security is built-in and thus software is developed with security in mind.

Security is most effective if planned and managed throughout every stage of the software development life cycle (SDLC), especially in critical applications or those that process sensitive information.

As technology advances, application environments become more complex and application development security becomes more challenging. Applications, systems, and networks are constantly under various security attacks such as malicious code or denial of service. Some of the challenges from the application development security point of view include Viruses, Trojan horses, Logic bombs, Worms, Agents, and Applets.

4.5.1 Credit Card Fraud

Credit card fraud is the most common security threat faced while making a digital payment. It occurs when a hacker gains unauthorized access to customers' personal and payment information. To access this data, the hacker may penetrate the database of an e-commerce site using malicious software programs. At times, a hacker's intention when stealing customers' data is to sell it on black markets.

4.5.2 Malware

In information technology, malware simply refers to malicious software programs. Attackers usually inject web pages or files with these malicious programs to help them in gaining access to online retail stores. Through means such as SQL injection, they can easily insert the malware into a website's database allowing it to compromise the data stored in the database.

4.5.3 Man-in-the-middle Attacks

Failed Transactions might cause loss for the user as there might be situations where the transactions are not reflected in the admin's database as a man-in-the-middle hacked into the system. As hackers are becoming smarter with technology, they are devising ways of listening to the communications made by users of an e-commerce website. Through an approach known

as a man-in-the-middle attack, these hackers maliciously trick users into connecting to a public wireless network. They gain access to people's devices once they are on public wireless networks. Hackers get to see people's browsing history, credit card numbers, passwords, and usernames if the websites they are visiting lack strong encryptions.

4.5.4 Denial of Service Attacks (DOS)

This is a threat when the user tries to add the product to his cart and if the hacker overloads the application server, then that may lead the user to pay an extra amount. This type of attack is known as a denial-of-service attack.

4.5.5 Domain name service (DNS) attack

“Attacks against the Domain name service could route traffic to an attacker instead of to the legitimate smart shopping service.”

4.6 Test Cases

Testing is the process of executing a program with the intent of finding any errors.

Testing is vital to the success of the system. Without proper testing, hidden errors will surface after some time of use and perhaps irreversible damage has been done to valuable data. A series of tests like responsiveness, its value, stress, and security are performed before the system is ready for user acceptance testing. System testing follows the logical conclusion that all parts of the system are tested and found to be working properly under all kinds of situations, and then the system is achieving its goal of processing the data perfectly, according to user rules and requirements.

Name	Description	Inputs	Expected Output
Login Test Case	Test for blank fields incorrect credentials and correct credentials.	No Input	Prompt Error Message “Fields Cannot be Left Blank. Please enter valid credentials.”
		Incorrect Credentials	Prompt Error Message “Incorrect Username or Password. Please enter valid credentials.”
		Correct Credentials	The user is logged in into the system successfully.
Logout Test Case	To check whether the admin and the user can log out	Login first into the system and then click on the logout button.	The user is logged out of the system successfully.

	of the system successfully.		
Scanning Barcode Test Case	To check whether the user can successfully scan the barcode of the product.	Scan the barcode of a product and read its description.	The user has successfully scanned the barcode.
Adding to Checklist Test Case	To check whether the user can list the products.	Add the name of the product to the Checklist.	The user has successfully added an item to the Checklist.
Adding to Cart Test Case	To check whether the user can add the products to the cart.	Add the product to the cart.	The user has successfully added an item to the cart.
Payment Test Case	To check whether the user can pay through their preferred mode of payment.	Checking out in the app and paying for the items in the cart.	The user has successfully paid for his cart through their selected payment mode.

Table 4.6 Test Cases

Chapter 5: Implementation and Testing

5.1 Implementation Approaches

The meaning of Agile is swift or versatile. "Agile process model" refers to a software development approach based on iterative development.

The Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In Agile, the tasks are divided into time boxes (small time frames) to deliver specific features for a release.

The iterative approach is taken and the working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.

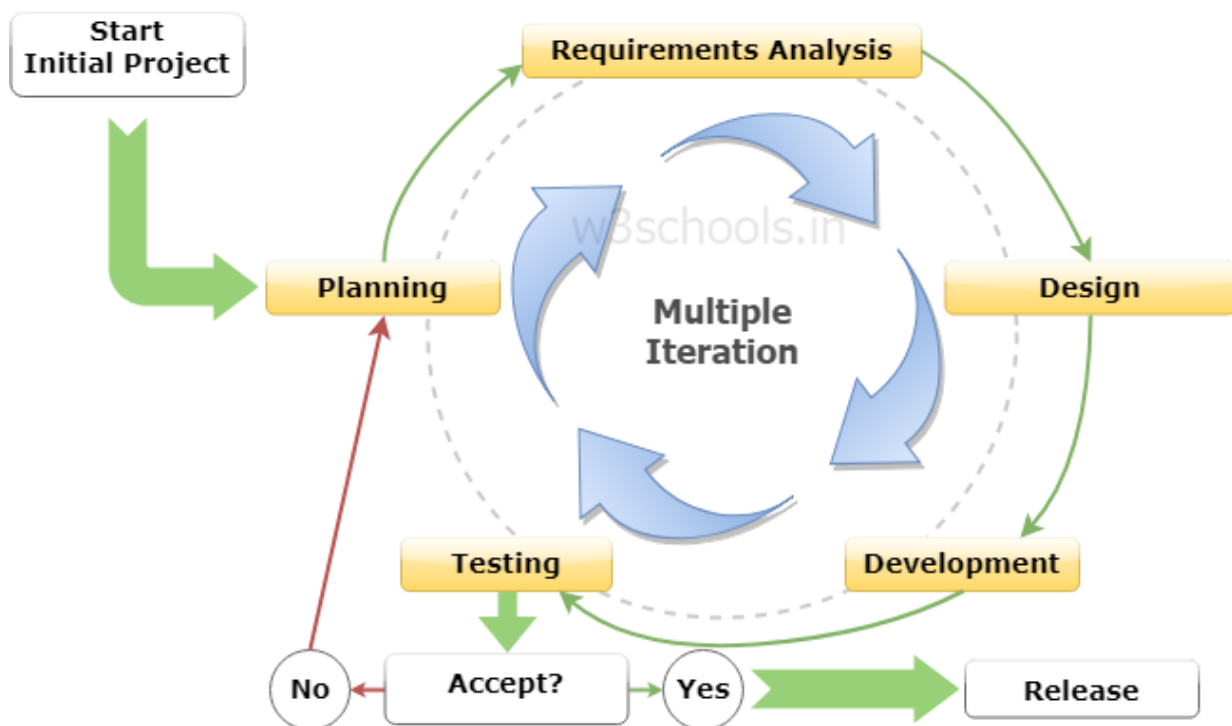


Figure 5.1 SDLC Agile Software Development Model

Following are the phases in the Agile model are as follows:

1. Requirements gathering
2. Design the requirements
3. Construction/ iteration
4. Testing/ Quality assurance
5. Deployment
6. Feedback

1. Requirements gathering:

In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

2. Design the requirements:

When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.

3. Construction/ iteration:

When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

4. Testing:

In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

5. Deployment:

In this phase, the team issues a product for the user's work environment.

6. Feedback:

After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

The agile model focuses on breaking down and making changes and then testing and applying the model. By applying a similar approach as we were presenting our project in the iterative forms, it helped us make our application in small iterations.

So, while making our project we decided what were the things we would complete in phase I, similarly completing our major work by phase II and then finalizing by phase III.

In phase I, we completed our UI design, Logo, and all the parts related to the interface and how the appearance of the application will be and started coding for the same. We got feedback and made the necessary changes.

Phase II consisted of working on the backend i.e., the database, and continuing with the frontend building.

In the last phase, we worked on the Cart and Checklist feature, thus wrapping up the project with final changes according to the feedback.

ITERATIONS

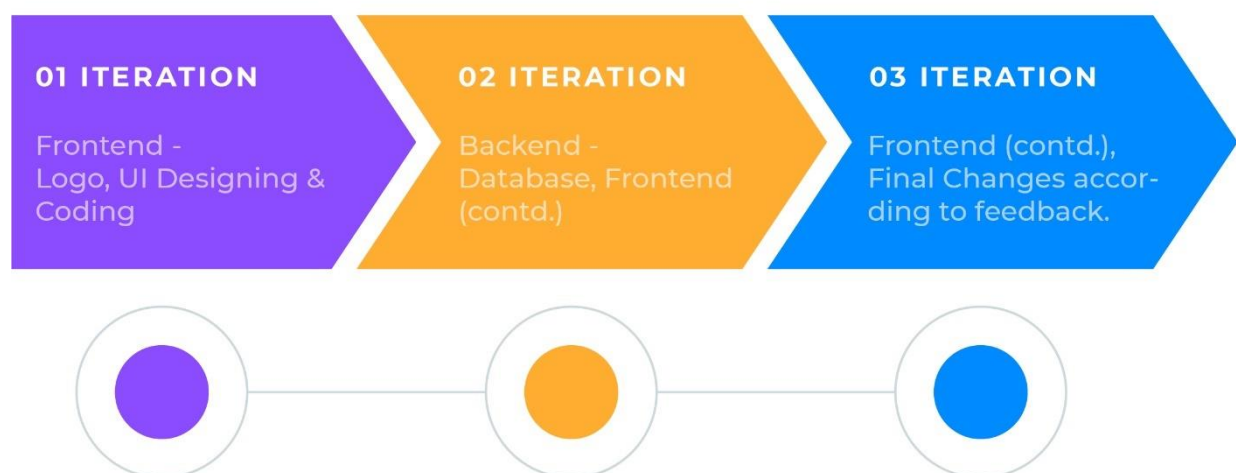


Figure 5.1.1 Implementation of Agile in Add to My Cart

5.1.1 Class Diagram

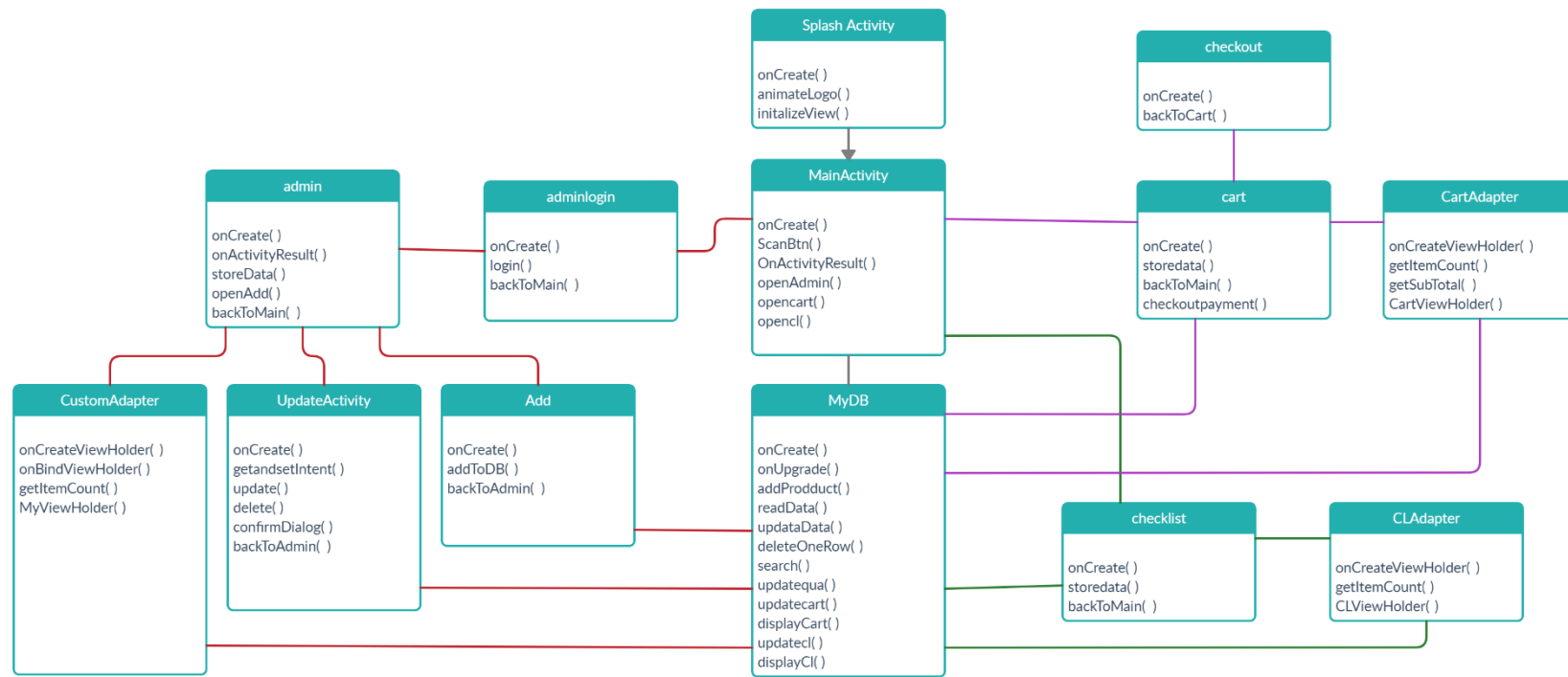


Figure 5.1.1 Class Diagram of Add to My Cart

5.1.2 Database Design

Database Inspector

OPPO CPH1969 > com.example.addtomycart

Databases

product_info

Refresh table Live updates

	_id	product_name	product_price	product_weight	product_category	product_description	cart	checklist	quantity
1	8902442211261	Youth Diary	70	160 Pages	Stationary	A5 Notebook Diary	0	0	5
2	8902442231047	Youth Diary	70	192 Pages	Stationary	B6 Notebook Diary	0	0	10

Results are read-only

50

Figure 5.1.2 Database Design of Add to My Cart

5.1.3 Phase Table

TASK	ASSIGNED TO	PROGRESS	START	END	DAYS
Project Approval & Planning			10-Aug-20	16-Nov-20	98
Introduction	Name		10-Aug-20	24-Aug-20	14
		100%	12-Aug-20	26-Aug-20	14
Survey of Technology			25-Aug-20	8-Sep-20	14
		100%	27-Aug-20	8-Sep-20	12
Requirement Analysis			9-Sep-20	14-Oct-20	35
		100%	9-Sep-20	17-Oct-20	38
System Design			15-Oct-20	15-Nov-20	31
		100%	18-Oct-20	16-Nov-20	29

19/4/21

TASK	ASSIGNED TO	PROGRESS	START	END	DAYS
Iteration 1			11-Jan-21	1-Feb-21	21
A. Planning			11-Jan-21	14-Jan-21	3
		100%	11-Jan-21	14-Jan-21	3
B. Implementation			15-Jan-21	20-Jan-21	5
		100%	15-Jan-21	21-Jan-21	6
C. Testing			21-Jan-21	25-Jan-21	4
		100%	22-Jan-21	29-Jan-21	7
Feedback			30-Jan-21	1-Feb-21	2

TASK	ASSIGNED TO	PROGRESS	START	END	DAYS
Iteration 2			1-Feb-21	7-Mar-21	34
A. Planning			1-Feb-21	6-Feb-21	5
		100%	1-Feb-21	8-Feb-21	7
B. Implementation			7-Feb-21	17-Feb-21	10
		100%	9-Feb-21	24-Feb-21	15
C. Testing			18-Feb-21	25-Feb-21	7
		100%	25-Feb-21	2-Mar-21	5
Feedback			3-Mar-21	7-Mar-21	4

TASK	ASSIGNED TO	PROGRESS	START	END	DAYS
Iteration 3			8-Mar-21	25-Mar-21	17
A. Planning			8-Mar-21	13-Mar-21	5
		100%	date	date	
B. Implementation			14-Mar-21	24-Mar-21	10
		100%	date	date	
C. Testing			16-Mar-21	25-Mar-21	9
		100%			
Completion			25-Mar-21	25-Mar-21	

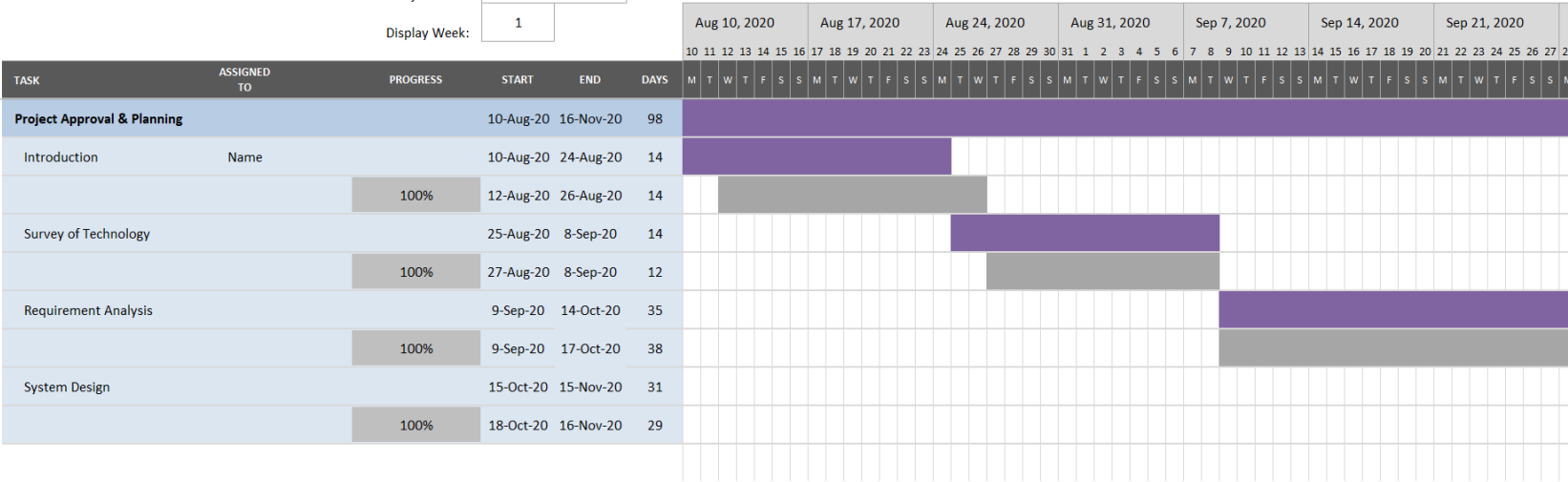
5.1.4 Gantt Chart

ADD TO MY CART

Company Name
Project Lead

Project Start: Mon, 8-10-2020

Display Week: 1

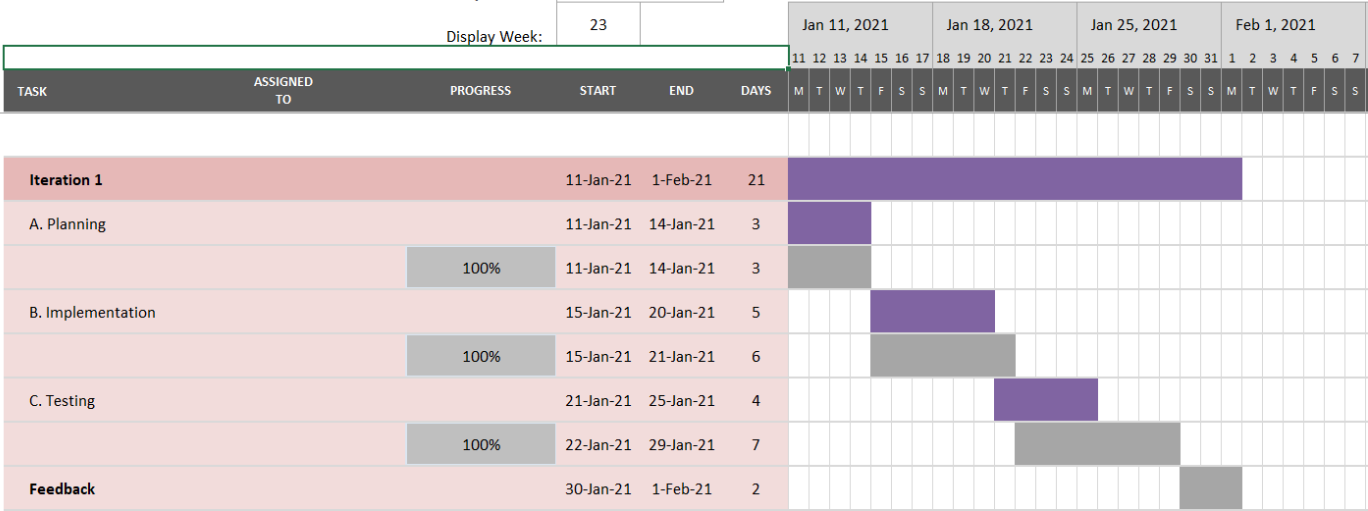


ADD TO MY CART

Company Name
Project Lead

Project Start: Mon, 8-10-2020

Display Week: 23



ADD TO MY CART

Company Name
Project Lead

Project Start: Mon, 8-10-2020
Display Week: 26

					Display Week:		26				Feb 1, 2021							Feb 8, 2021							Feb 15, 2021							Feb 22, 2021							Mar 1, 2021							Mar 8, 2021						
											1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6	7	8	9	10	11	12	13	14
TASK	ASSIGNED TO	PROGRESS	START	END	DAYS	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S					
Iteration 2			1-Feb-21	7-Mar-21	34																																															
A. Planning			1-Feb-21	6-Feb-21	5																																															
		100%	1-Feb-21	8-Feb-21	7																																															
B. Implementation			7-Feb-21	17-Feb-21	10																																															
		100%	9-Feb-21	24-Feb-21	15																																															
C. Testing			18-Feb-21	25-Feb-21	7																																															
		100%	25-Feb-21	2-Mar-21	5																																															
Feedback			3-Mar-21	7-Mar-21	4																																															

ADD TO MY CART

Company Name
Project Lead

Project Start: Mon, 8-10-2020
Display Week: 31

Project Start:						Mar 8, 2021							Mar 15, 2021							Mar 22, 2021							Mar 29, 2021						
Display Week:						8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3	
TASK	ASSIGNED TO	PROGRESS	START	END	DAYS	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S
Iteration 3			8-Mar-21	25-Mar-21	17																												
A. Planning			8-Mar-21	13-Mar-21	5																												
		100%	date	date																													
B. Implementation			14-Mar-21	24-Mar-21	10																												
		100%	date	date																													
C. Testing			16-Mar-21	25-Mar-21	9																												
		100%																															
Completion			25-Mar-21	25-Mar-21																													

19/4/21

5.2 Coding Details & Code Efficiency

5.2.1 Code Efficiency

- MyDB.java:

```
package com.example.addtomycart;

import android.annotation.SuppressLint;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.widget.Toast;

import androidx.annotation.Nullable;

import java.util.ArrayList;

public class MyDB extends SQLiteOpenHelper {

    private final Context context;
    private static final String DB_NAME = "ATMC.db";
    private static final int DB_VERSION = 1;

    public static final String TABLE_NAME = "product_info";
    public static final String COLUMN_ID = "_id";
    public static final String COLUMN_NAME = "product_name";
    public static final String COLUMN_PRICE = "product_price";
    public static final String COLUMN_WEIGHT = "product_weight";
    public static final String COLUMN_CAT = "product_category";
    public static final String COLUMN_DES = "product_description";
    public static final String COLUMN_CART = "cart";
    public static final String COLUMN_CL = "checklist";
    public static final String COLUMN_QUA = "quantity";

    MyDB(@Nullable Context context) {
        super(context, DB_NAME, null, DB_VERSION);
        this.context = context;
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String query = "CREATE TABLE " + TABLE_NAME +
            " (" + COLUMN_ID + " TEXT PRIMARY KEY, " +
            COLUMN_NAME + " TEXT, " +
            COLUMN_PRICE + " TEXT, " +
            COLUMN_WEIGHT + " TEXT, " +
            COLUMN_CAT + " TEXT, " +
            COLUMN_DES + " TEXT, " +
            COLUMN_CART + " INTEGER, " +
            COLUMN_CL + " INTEGER, " +
            COLUMN_QUA + " INTEGER);";
        db.execSQL(query);
    }
}
```

```

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
    }

    void addProduct(String pid, String pname, String pprice, String pweight, String
cat, String pdes) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues cv = new ContentValues();
        cv.put(COLUMN_ID, pid);
        cv.put(COLUMN_NAME, pname);
        cv.put(COLUMN_PRICE, pprice);
        cv.put(COLUMN_WEIGHT, pweight);
        cv.put(COLUMN_CAT, cat);
        cv.put(COLUMN_DES, pdes);
        cv.put(COLUMN_CART, 0);
        cv.put(COLUMN_CL, 0);
        cv.put(COLUMN_QUA, 1);
        long result = db.insert(TABLE_NAME, null, cv);
        if (result == -1)
            Toast.makeText(context, "Failed", Toast.LENGTH_SHORT).show();
        else
            Toast.makeText(context, "Added Successfully",
Toast.LENGTH_SHORT).show();
    }

    public Cursor readData() {
        String query = "SELECT * FROM " + TABLE_NAME;
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = null;
        if (db != null) {
            cursor = db.rawQuery(query, null);
        }
        return cursor;
    }

    @SuppressWarnings("Recycle")
    public ArrayList<String> search(String id) throws SQLException {
        Cursor cursor;
        SQLiteDatabase db = this.getReadableDatabase();
        cursor = db.query(TABLE_NAME, new String[]{COLUMN_ID, COLUMN_NAME, COL-
UMN_PRICE, COLUMN_WEIGHT, COLUMN_CAT, COLUMN_DES}, COLUMN_ID + "=?",
            new String[]{id}, null, null, null);
        ArrayList<String> data = new ArrayList<>();
        if (cursor.moveToFirst()) {
            do {
                String id1 = cursor.getString(cursor.getColumnIndex(COLUMN_ID));
                String name = cursor.getString(cursor.getColumnIndex(COLUMN_NAME));
                String cost = cursor.getString(cursor.getColumnIndex(COL-
UMN_PRICE));

                String w = cursor.getString(cursor.getColumnIndex(COLUMN_WEIGHT));
                String cat = cursor.getString(cursor.getColumnIndex(COLUMN_CAT));
                String des = cursor.getString(cursor.getColumnIndex(COLUMN_DES));
                data.add(id1);
                data.add(name);
                data.add(cost);
                data.add(w);
            } while (cursor.moveToNext());
        }
    }

```

```

        data.add(cat);
        data.add(des);
    } while (cursor.moveToNext());
}
cursor.close();
return data;
}

void updateData(String id, String name, String price, String weight, String
cat, String des) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues cv = new ContentValues();
    cv.put(COLUMN_ID, id);
    cv.put(COLUMN_NAME, name);
    cv.put(COLUMN_PRICE, price);
    cv.put(COLUMN_WEIGHT, weight);
    cv.put(COLUMN_CAT, cat);
    cv.put(COLUMN_DES, des);
    long result = db.update(TABLE_NAME, cv, "_id=?", new String[]{id});
    if (result == -1) {
        Toast.makeText(context, "Failed", Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(context, "Updated Successfully!",
Toast.LENGTH_SHORT).show();
    }
}

void updatequa(int val,String id){
    SQLiteDatabase db=this.getWritableDatabase();
    ContentValues cv2=new ContentValues();
    cv2.put(COLUMN_QUA,val);
    db.update(TABLE_NAME, cv2,"_id=?", new String[]{id});
    db.close();
}

void updatecart(int i,String id) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues cv1 = new ContentValues();
    cv1.put(COLUMN_CART, i);
    db.update(TABLE_NAME, cv1,"_id=?", new String[]{id});
    db.close();
}

public void updatecl(int i, String id1) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues cv1 = new ContentValues();
    cv1.put(COLUMN_CL, i);
    db.update(TABLE_NAME, cv1,"_id=?", new String[]{id1});
    db.close();
}

public Cursor displayCart() {
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = null;
    if (db != null) {
        cursor = db.query(TABLE_NAME, new String[]{COLUMN_ID,COLUMN_NAME, COL-
UMN_PRICE,COLUMN_CART,COLUMN_QUA}, COLUMN_CART + "=?",
        new String[]{"1"}, null, null, null, null);
    }
}

```

```

        return cursor;
    }

    public Cursor displayCl() {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = null;
        if (db != null) {
            cursor = db.query(TABLE_NAME, new String[]{COLUMN_ID, COLUMN_NAME, COLUMN_PRICE, COLUMN_CL}, COLUMN_CL + "=?",
                new String[]{String.valueOf(1)}, null, null, null, null);
        }
        return cursor;
    }

    void deleteOneRow(String id) {
        SQLiteDatabase db = this.getWritableDatabase();
        long result = db.delete(TABLE_NAME, "_id=?", new String[]{id});
        if (result == -1) {
            Toast.makeText(context, "Failed to Delete.",
                Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(context, "Successfully Deleted.",
                Toast.LENGTH_SHORT).show();
        }
    }
}

```

- MainActivity.java:

```

package com.example.addtomycart;

import android.Manifest;
import android.annotation.SuppressLint;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.SwitchCompat;
import androidx.core.app.ActivityCompat;

import com.google.zxing.integration.android.IntentIntegrator;
import com.google.zxing.integration.android.IntentResult;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    private TextView textView, textView1, name, weight, cost, cat;
    Button addcart, addchecklist;
    String name1, cost1;
}

```



```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    textView = findViewById(R.id.textView);
    textView1 = findViewById(R.id.textView1);
    name = findViewById(R.id.name);
    cost = findViewById(R.id.cost);
    weight = findViewById(R.id.weight);
    cat = findViewById(R.id.cat);
    addcart = findViewById(R.id.cartadd);
    addcheckboxlist = findViewById(R.id.checkboxlistadd);
    ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA}, PackageManager.PERMISSION_GRANTED);
}

public void openAdmin(View v) {
    Intent intent = new Intent(this, adminlogin.class);
    startActivity(intent);
}

public void ScanBtn(View view) {
    IntentIntegrator intentIntegrator = new IntentIntegrator(this);
    SwitchCompat torch = findViewById(R.id.switch1);
    torch.setTextOn("On");
    torch.setTextOff("Off");
    intentIntegrator.setTorchEnabled(torch.isChecked());
    intentIntegrator.setOrientationLocked(false);
    intentIntegrator.initiateScan();
}

@SuppressLint("SetTextI18n")
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    IntentResult intentResult = IntentIntegrator.parseActivityResult(requestCode, resultCode, data);
    String id = intentResult.getContents();
    ArrayList<String> data1;
    MyDB db = new MyDB(this);
    data1 = db.search(id);
    if (!data1.isEmpty()) {
        textView.setText(data1.get(0));
        name.setText(data1.get(1));
        cost.setText("Rs. " + data1.get(2));
        weight.setText(data1.get(3));
        cat.setText(data1.get(4));
        textView1.setText(data1.get(5));
        addcart.setVisibility(View.VISIBLE);
        addcheckboxlist.setVisibility(View.VISIBLE);
        name1 = data1.get(1);
        cost1 = data1.get(2);
        String id1 = data1.get(0);

        Button button = findViewById(R.id.cartadd);
        button.setOnClickListener(v -> {
            MyDB myDB = new MyDB(MainActivity.this);
            myDB.updatecart(1, id1);
            Toast.makeText(MainActivity.this, "Successfully Added in Cart",

```

```

Toast.LENGTH_SHORT).show();
    });

    Button button1 = findViewById(R.id.checkListadd);
    button1.setOnClickListener(v -> {
        MyDB myDB = new MyDB(MainActivity.this);
        myDB.updatecl(1, id1);
        Toast.makeText(MainActivity.this, "Successfully Added in Check-
list", Toast.LENGTH_SHORT).show();
    });

    } else {
        textView.setText(id + " - Not in Database ");
        Toast.makeText(this, "Product not in Database",
Toast.LENGTH_SHORT).show();
    }
    super.onActivityResult(requestCode, resultCode, data);
}

public void opencart(View v) {
    Intent intent = new Intent(MainActivity.this, cart.class);
    startActivity(intent);
}

public void opencl(View v) {
    Intent intent2 = new Intent(MainActivity.this, checklist.class);
    startActivity(intent2);
}
}

```

- cart.java:

```

package com.example.addtomycart;

import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;

public class cart extends AppCompatActivity {

    TextView tot;
    RecyclerView recyclerView;
    CartAdapter adapter;
    ArrayList<String> cname, ccost, cid, cqua;
    ArrayList<Integer> ctotat;
    int total;

    public void backToMain(View v) {

```

```

        Intent intent = new Intent(cart.this, MainActivity.class);
        startActivity(intent);
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cart);
        cname = new ArrayList<>();
        ccost = new ArrayList<>();
        cqua = new ArrayList<>();
        cid = new ArrayList<>();
        cttotal=new ArrayList<>();
        storedata();
        tot = findViewById(R.id.amt);
        recyclerView = findViewById(R.id.cartLv);
        adapter = new CartAdapter(cart.this, this, cid,cname, ccost, cqua,cto-
tal,tot);
        recyclerView.setAdapter(adapter);
        recyclerView.setLayoutManager(new LinearLayoutManager(cart.this));
    }

    void storedata() {
        int amt,qua;
        MyDB db = new MyDB(cart.this);
        Cursor cursor = db.displayCart();
        if (cursor.getCount() != 0) {
            while (cursor.moveToNext()) {
                cid.add(cursor.getString(0));
                cname.add(cursor.getString(1));
                ccost.add(cursor.getString(2));
                amt=Integer.parseInt(cursor.getString(2));
                cqua.add(cursor.getString(4));
                qua=Integer.parseInt(cursor.getString(4));
                cttotal.add(amt*qua);
            }
            Toast.makeText(this, "Total Data in Cart: " + cursor.getCount(),
Toast.LENGTH_SHORT).show();
        } else
            Toast.makeText(this, "No Data in Cart.", Toast.LENGTH_SHORT).show();
    }

    public void checkoutpayent(View v) {
        Intent intent1 = new Intent(cart.this, checkout.class);
        intent1.putExtra("subtotal",adapter.getSubTotal());
        startActivity(intent1);
    }
}

```

- CartAdapter.java:

```
package com.example.addtomycart;

import android.app.Activity;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;

public class CartAdapter extends RecyclerView.Adapter<CartAdapter.CartViewHolder> {

    Activity activity;
    Context context;
    ArrayList<String> name, cost, id, qua;
    ArrayList<Integer> total;
    TextView sum;
    int subtotal;

    public CartAdapter(Activity activity1, Context context1, ArrayList<String> cid,
        ArrayList<String> cname, ArrayList<String> ccost, ArrayList<String> cqua, Ar-
        rayList<Integer> ctotat, TextView tot) {
        activity = activity1;
        context = context1;
        id = cid;
        name = cname;
        cost = ccost;
        qua = cqua;
        total = ctotat;
        sum = tot;
    }

    public CartAdapter() {

    }

    @NonNull
    @Override
    public CartViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
        viewType) {
        View itemview = LayoutInflater.from(context).inflate(R.layout.list_item,
        parent, false);
        return new CartViewHolder(itemview);
    }

    @Override
    public void onBindViewHolder(@NonNull CartViewHolder holder, int position) {
        MyDB db = new MyDB(context);
        holder.name0.setText(String.valueOf(name.get(position)));
        holder.price0.setText(String.valueOf(total.get(position)));
    }
}
```

```

        db.updatequa(Integer.parseInt(qua.get(position)), id.get(position));
        holder.qua0.setText(qua.get(position));
        holder.inc.setOnClickListener(v -> {
            int value = Integer.parseInt(qua.get(position));
            value++;
            db.updatequa(value, id.get(position));
            qua.set(position, String.valueOf(value));
            total.set(position, Integer.parseInt(cost.get(position)) * value);
            holder.price0.setText(String.valueOf(total.get(position)));
            notifyDataSetChanged();
            holder.qua0.setText(qua.get(position));
            notifyDataSetChanged();
        });
        holder.dec.setOnClickListener(v -> {
            int value = Integer.parseInt(qua.get(position));
            value--;
            if (value != 0) {
                db.updatequa(value, id.get(position));
                qua.set(position, String.valueOf(value));
                Toast.makeText(context, "Quantity Updated.",
                    Toast.LENGTH_SHORT).show();
            } else {
                value = 1;
                Toast.makeText(context, "Quantity cannot be 0.",
                    Toast.LENGTH_SHORT).show();
            }
            total.set(position, Integer.parseInt(cost.get(position)) * value);
            holder.price0.setText(String.valueOf(total.get(position)));
            notifyDataSetChanged();
            holder.qua0.setText(qua.get(position));
            notifyDataSetChanged();
        });
        holder.delete.setOnClickListener(v -> {
            db.updatecart(0, id.get(position));
            name.remove(position);
            cost.remove(position);
            qua.remove(position);
            id.remove(position);
            total.remove(position);
            notifyDataSetChanged();
            notifyItemChanged(position);
        });
        subtotal=getSubTotal();
        sum.setText(String.valueOf(subtotal));
    }

    @Override
    public int getItemCount() {
        return name.size();
    }

    public int getSubTotal() {
        int subTotal = 0;
        for (int i = 0; i < total.size(); i++) {
            subTotal = subTotal + total.get(i);
        }
        return subTotal;
    }

    static class CartViewHolder extends RecyclerView.ViewHolder {

```

```

        TextView name0, price0, qua0;
        public LinearLayout mainLayout;
        Button inc, dec, delete;

        public CartViewHolder(@NonNull View itemView) {
            super(itemView);
            name0 = itemView.findViewById(R.id.pro_name);
            price0 = itemView.findViewById(R.id.pro_cost);
            qua0 = itemView.findViewById(R.id.pro_qua);
            inc = itemView.findViewById(R.id.add1);
            dec = itemView.findViewById(R.id.remove);
            delete = itemView.findViewById(R.id.delete1);
            mainLayout = itemView.findViewById(R.id.cartLayout);
        }
    }
}

```

- activity_main.xml:

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:scrollbars="vertical"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/checklistadd"
        android:layout_width="0dp"
        android:layout_height="48dp"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="32dp"
        android:layout_marginBottom="8dp"
        android:layout_alignParentRight="true"
        android:paddingLeft="0dp"
        android:paddingTop="0dp"
        android:paddingRight="0dp"
        android:paddingBottom="0dp"
        android:text="@string/add_to_checklist"
        android:textAlignment="center"
        android:textSize="12sp"
        android:textStyle="bold"
        android:visibility="invisible"
        app:cornerRadius="50dp"
        app:iconPadding="0dp"
        app:layout_constraintBottom_toTopOf="@+id/textView1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/cat"
        app:layout_constraintTop_toBottomOf="@+id/cartadd" />

    <TextView
        android:id="@+id/cat"
        android:layout_width="0dp"
        android:layout_height="25dp"
        android:layout_marginStart="32dp"

```

```
android:layout_marginTop="8dp"
android:layout_marginEnd="200dp"
android:background="@drawable/roundedbtn_greystroke"
android:elegantTextHeight="true"
android:paddingStart="5dp"
android:text="@string/product_category"
android:textAlignment="viewStart"
android:textColor="@color/black"
android:textSize="14sp"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/weight"
tools:ignore="RtlSymmetry" />
```

```
<TextView
    android:id="@+id/textView"
    android:layout_width="0dp"
    android:layout_height="25dp"
    android:layout_marginStart="32dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="32dp"
    android:background="@drawable/roundedbtn_greystroke"
    android:elegantTextHeight="true"
    android:paddingStart="5dp"
    android:text="@string/product_id"
    android:textAlignment="viewStart"
    android:textColor="@color/black"
    android:textIsSelectable="true"
    android:textSize="16sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/toolbar"
    tools:ignore="RtlSymmetry" />
```

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="32dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="32dp"
    android:layout_marginBottom="150dp"
    android:background="@drawable/roundedbtn_greystroke"
    android:elegantTextHeight="true"
    android:paddingStart="5dp"
    android:scrollbarAlwaysDrawVerticalTrack="true"
    android:scrollbars="vertical"
    android:text="@string/product_description"
    android:textAlignment="viewStart"
    android:textColor="@color/black"
    android:textSize="14sp"
    app:layout_constraintBottom_toTopOf="@+id/scanbtn"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/cat"
    tools:ignore="RtlSymmetry" />
```

```
<TextView
    android:id="@+id/name"
```

```
android:layout_width="0dp"
android:layout_height="wrap_content"
android:layout_marginStart="32dp"
android:layout_marginTop="8dp"
android:layout_marginEnd="32dp"
android:background="@drawable/roundedbtn_greystroke"
android:elegantTextHeight="true"
android:paddingStart="5dp"
android:text="@string/product_name"
android:textAlignment="viewStart"
android:textColor="@color/black"
android:textSize="16sp"
android:textStyle="bold"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/textView"
tools:ignore="RtlSymmetry" />
```

<TextView

```
android:id="@+id/cost"
android:layout_width="0dp"
android:layout_height="25dp"
android:layout_marginStart="32dp"
android:layout_marginTop="8dp"
android:layout_marginEnd="200dp"
android:background="@drawable/roundedbtn_greystroke"
android:elegantTextHeight="true"
android:paddingStart="5dp"
android:text="@string/product_price1"
android:textAlignment="viewStart"
android:textColor="@color/black"
android:textSize="14sp"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/divider"
tools:ignore="RtlSymmetry" />
```

<TextView

```
android:id="@+id/weight"
android:layout_width="0dp"
android:layout_height="25dp"
android:layout_marginStart="32dp"
android:layout_marginTop="8dp"
android:layout_marginEnd="200dp"
android:background="@drawable/roundedbtn_greystroke"
android:elegantTextHeight="true"
android:paddingStart="5dp"
android:text="@string/product_unit"
android:textAlignment="viewStart"
android:textColor="@color/black"
android:textSize="14sp"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/cost"
tools:ignore="RtlSymmetry" />
```

<ImageButton

```
android:id="@+id/scanbtn"
android:layout_width="105dp"
android:layout_height="108dp"
```



```
    android:layout_marginEnd="16dp"
    android:layout_marginBottom="16dp"
    android:backgroundTint="#00FFFFFF"
    android:contentDescription="@string/todo"
    android:onClick="ScanBtn"
    android:scaleType="fitXY"
    app:layout_constraintBottom_toTopOf="@+id/constraintLayout2"
    app:layout_constraintEnd_toEndOf="parent"
    app:srcCompat="@drawable/scan" />
```

```
<androidx.appcompat.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="0dp"
    android:layout_height="64dp"
    android:background="?attr/colorPrimary"
    android:minHeight="?attr/actionBarSize"
    android:theme="?attr/actionBarTheme"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

```
<androidx.appcompat.widget.SwitchCompat
    android:id="@+id/switch1"
    android:layout_width="120dp"
    android:layout_height="34dp"
    android:layout_marginEnd="16dp"
    android:checked="false"
    android:text="@string/torch"
    android:textAlignment="textStart"
    android:textAllCaps="false"
    android:textColor="@color/white"
    android:textStyle="bold"
    android:theme="@style/SCBSwitch"
    app:layout_constraintBottom_toBottomOf="@+id/toolbar"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toEndOf="@+id/toolbar"
    app:layout_constraintTop_toTopOf="@+id/toolbar"
    app:layout_constraintVertical_bias="0.533"
    tools:ignore="MissingConstraints" />
```

```
<View
    android:id="@+id/divider"
    android:layout_width="match_parent"
    android:layout_height="2dp"
    android:layout_marginTop="16dp"
    android:background="@color/yellow"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/name" />
```

```
<ImageView
    android:id="@+id/adminbtn"
    android:layout_width="40dp"
    android:layout_height="32dp"
    android:layout_marginStart="16dp"
    android:layout_marginTop="16dp"
    android:contentDescription="@string/todo"
    android:onClick="openAdmin"
```

```

        android:scaleType="fitXY"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/admin" />

<androidx.constraintlayout.widget.ConstraintLayout
    android:id="@+id/constraintLayout2"
    android:layout_width="0dp"
    android:layout_height="50dp"
    android:background="@color/black"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent">

    <Button
        android:id="@+id/button2"
        android:layout_width="195dp"
        android:layout_height="50dp"
        android:background="#00FFFFFF"
        android:onClick="openc1"
        android:text="@string/checklist"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/divider2"
        app:layout_constraintTop_toTopOf="parent" />

    <View
        android:id="@+id/divider2"
        android:layout_width="2dp"
        android:layout_height="0dp"
        android:layout_marginTop="5dp"
        android:layout_marginBottom="5dp"
        android:background="@color/white"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:background="#00FFFFFF"
        android:onClick="opencart"
        android:text="@string/cart"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/divider2"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>

<Button
    android:id="@+id/cartadd"
    android:layout_width="0dp"
    android:layout_height="48dp"
    android:layout_marginStart="16dp"
    android:layout_marginTop="4dp"
    android:layout_marginEnd="32dp"
    android:elegantTextHeight="true"

```

```

        android:paddingLeft="0dp"
        android:paddingTop="0dp"
        android:paddingRight="0dp"
        android:paddingBottom="0dp"
        android:text="@string/add_to_cart"
        android:textAlignment="center"
        android:textSize="12sp"
        android:textStyle="bold"
        android:visibility="invisible"
        app:cornerRadius="50dp"
        app:iconPadding="0dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toEndOf="@+id/cost"
        app:layout_constraintTop_toBottomOf="@+id/divider" />

```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

- activity_update.xml:

- ```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:an-
droid="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:background="#AE1C1C"
 tools:context=".UpdateActivity">

 <ScrollView
 android:id="@+id/scrollView2"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:background="#FFFFFF"
 tools:layout_editor_absoluteX="0dp"
 tools:layout_editor_absoluteY="-161dp">

 <androidx.constraintlayout.widget.ConstraintLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 tools:layout_editor_absoluteX="-120dp"
 tools:layout_editor_absoluteY="0dp">

 <EditText
 android:id="@+id/pdes1"
 android:layout_width="0dp"
 android:layout_height="100dp"
 android:layout_marginStart="32dp"
 android:layout_marginEnd="32dp"
 android:background="@drawable/roundedbtn_greystroke"
 android:gravity="left|top"
 android:importantForAutofill="no"
 android:inputType="textMultiLine"
 android:paddingStart="5dp"
 android:paddingTop="2dp"
 android:scrollbarAlwaysDrawVerticalTrack="true"
 android:textColor="@color/black"
 />
 </androidx.constraintlayout.widget.ConstraintLayout>
 </ScrollView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

```

 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/temp6"
 tools:ignore="LabelFor,RtlHardcoded,RtlSymmetry" />

<TextView
 android:id="@+id/temp6"
 android:layout_width="wrap_content"
 android:layout_height="25dp"
 android:layout_marginStart="32dp"
 android:layout_marginTop="16dp"
 android:text="@string/product_description"
 android:textStyle="bold"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/pcat1" />

<EditText
 android:id="@+id/pcat1"
 android:layout_width="0dp"
 android:layout_height="30dp"
 android:layout_marginStart="32dp"
 android:layout_marginEnd="32dp"
 android:background="@drawable/roundedbtn_greystroke"
 android:importantForAutofill="no"
 android:inputType="text"
 android:paddingStart="5dp"
 android:paddingTop="2dp"
 android:textColor="@color/black"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/temp5"
 tools:ignore="LabelFor,RtlSymmetry" />

<TextView
 android:id="@+id/temp5"
 android:layout_width="wrap_content"
 android:layout_height="25dp"
 android:layout_marginStart="32dp"
 android:layout_marginTop="16dp"
 android:text="@string/product_category"
 android:textStyle="bold"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/pweight1" />

<EditText
 android:id="@+id/pweight1"
 android:layout_width="0dp"
 android:layout_height="30dp"
 android:layout_marginStart="32dp"
 android:layout_marginEnd="32dp"
 android:background="@drawable/roundedbtn_greystroke"
 android:importantForAutofill="no"
 android:inputType="text"
 android:paddingStart="5dp"
 android:paddingTop="2dp"
 android:textColor="@color/black"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/temp4"
 tools:ignore="LabelFor,RtlSymmetry" />

```

```

<EditText
 android:id="@+id/pprice1"
 android:layout_width="0dp"
 android:layout_height="30dp"
 android:layout_marginStart="32dp"
 android:layout_marginEnd="32dp"
 android:background="@drawable/roundedbtn_greystroke"
 android:importantForAutofill="no"
 android:inputType="number"
 android:paddingStart="5dp"
 android:paddingTop="2dp"
 android:textColor="@color/black"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintHorizontal_bias="0.0"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/temp3"
 tools:ignore="LabelFor,RtlSymmetry" />

<EditText
 android:id="@+id/pname1"
 android:layout_width="0dp"
 android:layout_height="30dp"
 android:layout_marginStart="32dp"
 android:layout_marginEnd="32dp"
 android:background="@drawable/roundedbtn_greystroke"
 android:importantForAutofill="no"
 android:inputType="text"
 android:paddingStart="5dp"
 android:paddingTop="2dp"
 android:textColor="@color/black"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintHorizontal_bias="0.0"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/temp2"
 tools:ignore="LabelFor,RtlSymmetry" />

<TextView
 android:id="@+id/temp3"
 android:layout_width="wrap_content"
 android:layout_height="25dp"
 android:layout_marginStart="32dp"
 android:layout_marginTop="16dp"
 android:text="@string/product_price"
 android:textStyle="bold"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/pname1" />

<TextView
 android:id="@+id/temp4"
 android:layout_width="wrap_content"
 android:layout_height="25dp"
 android:layout_marginStart="32dp"
 android:layout_marginTop="16dp"
 android:text="@string/product_weight_pages_volume_with_units"
 android:textStyle="bold"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/pprice1" />

<androidx.appcompat.widget.Toolbar

```

```

 android:id="@+id/toolbar4"
 android:layout_width="match_parent"
 android:layout_height="64dp"
 android:background="?attr/colorPrimary"
 android:minHeight="?attr/actionBarSize"
 android:theme="?attr/actionBarTheme"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintHorizontal_bias="1.0"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent"
 tools:ignore="DuplicateIds" />

<ImageView
 android:id="@+id/back2"
 android:layout_width="34dp"
 android:layout_height="32dp"
 android:layout_marginStart="16dp"
 android:layout_marginTop="16dp"
 android:contentDescription="@string/todo"
 android:onClick="backToRecycle"
 android:scaleType="fitXY"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent"
 app:srcCompat="@drawable/back" />

<EditText
 android:id="@+id/pid1"
 android:layout_width="0dp"
 android:layout_height="30dp"
 android:layout_marginStart="32dp"
 android:layout_marginEnd="32dp"
 android:background="@drawable/roundedbtn_greystroke"
 android:importantForAutofill="no"
 android:inputType="number"
 android:paddingStart="5dp"
 android:paddingTop="2dp"
 android:textColor="@color/black"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintHorizontal_bias="0.0"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/temp1"
 tools:ignore="LabelFor,RtlSymmetry" />

<TextView
 android:id="@+id/temp2"
 android:layout_width="wrap_content"
 android:layout_height="25dp"
 android:layout_marginStart="32dp"
 android:layout_marginTop="16dp"
 android:text="@string/product_name"
 android:textStyle="bold"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/pid1" />

<TextView
 android:id="@+id/temp1"
 android:layout_width="100dp"
 android:layout_height="25dp"
 android:layout_marginStart="32dp"
 android:layout_marginTop="32dp"

```

```

 android:text="@string/product_id"
 android:textStyle="bold"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/toolbar4" />

<Button
 android:id="@+id/update"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_marginStart="80dp"
 android:layout_marginTop="32dp"
 android:layout_marginEnd="80dp"
 android:backgroundTint="@color/yellow"
 android:onClick="update"
 android:padding="0dp"
 android:paddingTop="0dp"
 android:text="@string/update"
 android:textColor="@color/white"
 android:textSize="18sp"
 android:textStyle="bold"
 app:cornerRadius="100dp"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/pdes1" />

<Button
 android:id="@+id/delete"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_marginStart="80dp"
 android:layout_marginTop="4dp"
 android:layout_marginEnd="80dp"
 android:backgroundTint="#CA3030"
 android:onClick="delete"
 android:text="@string/delete"
 android:textSize="18sp"
 android:textStyle="bold"
 app:cornerRadius="100dp"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/update" />

</androidx.constraintlayout.widget.ConstraintLayout>

</ScrollView>

<androidx.appcompat.widget.Toolbar
 android:id="@+id/toolbar3"
 android:layout_width="match_parent"
 android:layout_height="64dp"
 android:background="?attr/colorPrimary"
 android:minHeight="?attr/actionBarSize"
 android:theme="?attr/actionBarTheme"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintHorizontal_bias="1.0"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent" />

<ImageView
 android:id="@+id/backadmin"

```



```

 android:layout_width="34dp"
 android:layout_height="32dp"
 android:layout_marginStart="16dp"
 android:layout_marginTop="16dp"
 android:contentDescription="@string/todo"
 android:onClick="backToRecycle"
 android:scaleType="fitXY"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent"
 app:srcCompat="@drawable/back" />

<TextView
 android:id="@+id/textView3"
 android:layout_width="wrap_content"
 android:layout_height="32dp"
 android:layout_marginTop="16dp"
 android:layout_marginEnd="16dp"
 android:text="@string/update"
 android:textAllCaps="true"
 android:textColor="@color/white"
 android:textSize="24sp"
 android:textStyle="bold"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintTop_toTopOf="@+id/scrollView2" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

- row.xml:

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/adminlayout"
 android:layout_width="match_parent"
 android:layout_height="wrap_content">

 <androidx.cardview.widget.CardView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_margin="5dp"
 tools:ignore="MissingConstraints"
 tools:layout_editor_absoluteX="8dp">

 <androidx.constraintlayout.widget.ConstraintLayout
 android:id="@+id/constraintLayout"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_margin="2dp"
 android:padding="10dp">

 <TextView
 android:id="@+id/procat"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_marginStart="4dp"
 android:layout_marginTop="4dp"
 android:text="@string/textview"

```



```
 android:textColor="#686868"
 android:textSize="10sp"
 app:layout_constraintStart_toEndOf="@+id/textView6"
 app:layout_constraintTop_toBottomOf="@+id/view"
 tools:ignore="SmallSp" />
```

```
<TextView
 android:id="@+id/textView6"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_marginStart="4dp"
 android:layout_marginTop="4dp"
 android:text="@string/grams_category"
 android:textColor="#686868"
 android:textSize="10sp"
 app:layout_constraintBottom_toBottomOf="@+id/procat"
 app:layout_constraintStart_toEndOf="@+id/proweight"
 app:layout_constraintTop_toBottomOf="@+id/view"
 tools:ignore="SmallSp" />
```

```
<TextView
 android:id="@+id/proweight"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_marginTop="4dp"
 android:text="@string/textview"
 android:textColor="#686868"
 android:textSize="10sp"
 app:layout_constraintBottom_toBottomOf="@+id/textView6"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/view"
 tools:ignore="SmallSp" />
```

```
<TextView
 android:id="@+id/proID"
 android:layout_width="wrap_content"
 android:layout_height="19dp"
 android:background="@drawable/roundedbtn_greystroke"
 android:paddingStart="5dp"
 android:paddingEnd="5dp"
 android:text="@string/textview"
 android:textColor="@color/black"
 android:textSize="14sp"
 android:textStyle="bold"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent" />
```

```
<TextView
 android:id="@+id/proname"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_marginTop="4dp"
 android:text="@string/textview"
 android:textAllCaps="true"
 android:textColor="@color/black"
 android:textStyle="bold"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/proID" />
```

```
<TextView
```

```

 android:id="@+id/prodes"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_marginTop="4dp"
 android:text="@string/textview"
 android:textSize="10sp"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/proweight"
 tools:ignore="SmallSp" />

<TextView
 android:id="@+id/proprice"
 android:layout_width="wrap_content"
 android:layout_height="0dp"
 android:layout_marginTop="1dp"
 android:layout_marginBottom="4dp"
 android:background="@drawable/roundedbtn_greycircle"
 android:padding="4dp"
 android:textAlignment="center"
 android:textColor="@color/black"
 android:textSize="20sp"
 android:textStyle="bold"
 app:layout_constraintBottom_toTopOf="@+id/view"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/textView2" />

<View
 android:id="@+id/view"
 android:layout_width="wrap_content"
 android:layout_height="2dp"
 android:layout_marginTop="4dp"
 android:background="@color/yellow"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintHorizontal_bias="1.0"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/proname" />

<TextView
 android:id="@+id/textView2"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/price"
 android:textAlignment="center"
 android:textColor="@color/black"
 android:textSize="4sp"
 app:layout_constraintEnd_toEndOf="@+id/proprice"
 app:layout_constraintStart_toStartOf="@+id/proprice"
 app:layout_constraintTop_toTopOf="parent"
 tools:ignore="SmallSp" />

</androidx.constraintlayout.widget.ConstraintLayout>
</androidx.cardview.widget.CardView>
</LinearLayout>

```

### 5.2.2 COCOMO Model

Cocomo (Constructive Cost Model) is a regression model based on LOC, i.e., the **number of Lines of Code**. It is a procedural cost estimate model for software projects and is often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time, and quality.

The key parameters which define the quality of any software products, which are also an outcome of the Cocomo are primarily Effort & Schedule:

- **Effort:** Amount of labour that will be required to complete a task. It is measured in person-months units.
- **Schedule:** Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put in. It is measured in the units of time such as weeks, months.

Different models of Cocomo have been proposed to predict the cost estimation at different levels, based on the amount of accuracy and correctness required. All of these models can be applied to a variety of projects, whose characteristics determine the value of constant to be used in subsequent calculations. These characteristics of different system types are mentioned below.

Boehm's definition of organic, semidetached, and embedded systems:

- **Organic** – A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.
- **Semi-detached** – A software project is said to be a Semi-detached type if the vital characteristics such as team size, experience, knowledge of the various programming environment lie in between that of organic and Embedded. The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience and better guidance and creativity.  
E.g.: Compilers or different Embedded Systems can be considered of Semi-Detached type.
- **Embedded** – A software project requiring the highest level of complexity, creativity, and experience requirement fall under this category. Such software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.

All the above system types utilize different values of the constants used in Effort Calculations.

### Estimation of Effort:

#### Calculations –

- $E = a(KLOC)^b$  Person-Month
- $Time = c(Effort)^d$  Months
- $Person\ required = Effort/time\ Persons$

The above formula is used for the cost estimation of the basic COCOMO model, and also is used in the subsequent models. The constant values a, b, c, and d for the Basic Model for the different categories of the system:

| Software Projects | a   | b    | c   | d    |
|-------------------|-----|------|-----|------|
| Organic           | 2.4 | 1.05 | 2.5 | 0.38 |
| Semi Detached     | 3.0 | 1.12 | 2.5 | 0.35 |
| Embedded          | 3.6 | 1.20 | 2.5 | 0.32 |

The effort is measured in Person-Months and as evident from the formula is dependent on Kilo-Lines of code. The development time is measured in Months.

The Project Code = 1043 LOC = 1.043 KLOC [1 KLOC = 1000 LOC]

| Mode          | Effort              | Time                |
|---------------|---------------------|---------------------|
| Organic       | $2.4(1.043)^{1.05}$ | $2.5(2.508)^{0.38}$ |
| Semi-Detached | $3.0(1.043)^{1.12}$ | $2.5(3.144)^{0.35}$ |
| Embedded      | $3.6(1.043)^{1.20}$ | $2.5(3.786)^{0.32}$ |

| Mode          | Effort | Time  |
|---------------|--------|-------|
| Organic       | 2.508  | 3.120 |
| Semi-Detached | 3.144  | 3.733 |
| Embedded      | 3.786  | 3.827 |

Average Number of Persons Required = 1

# 5.3 Testing Approach

## 5.3.1 Unit Testing

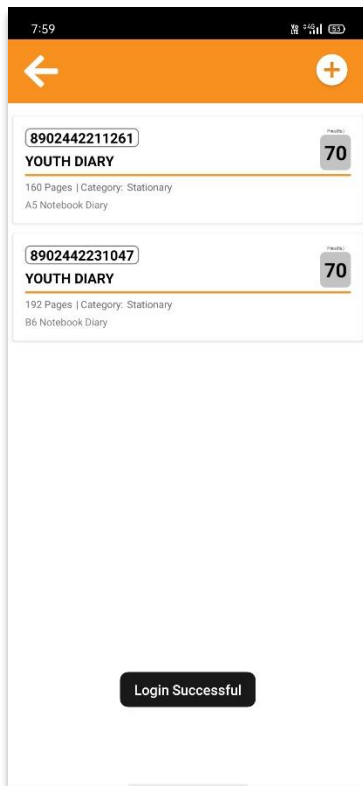
AdminLogin Module

| Test Scenario ID | Test Case Description                          | Preconditions                | Test Data                          | Postconditions             | Expected Result  | Status       |
|------------------|------------------------------------------------|------------------------------|------------------------------------|----------------------------|------------------|--------------|
| 1                | Login in with a valid email and valid password | Valid data in URL variable   | Username: admin<br>Password: admin | User will be Logged in     | Login Successful | Pass         |
|                  |                                                |                              |                                    |                            |                  |              |
| 2                | Login with an incorrect email or password      | Invalid data in URL variable | Username: admin<br>Password: 1234  | User will not be Logged in | Login Failed     | Unauthorized |
|                  |                                                |                              |                                    |                            |                  |              |

### TEST CASE 1

Username: admin

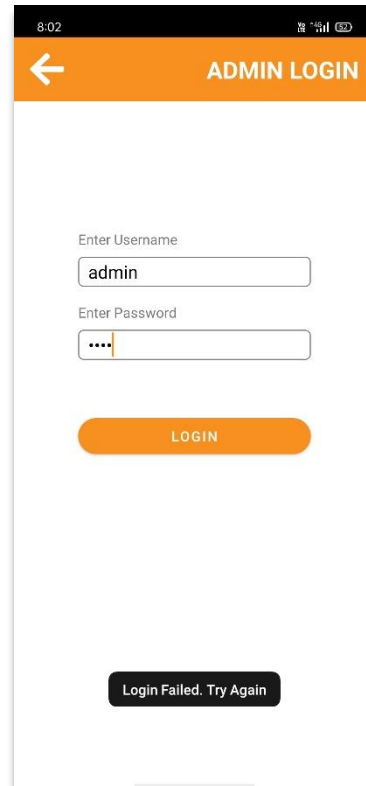
Password: admin



### TEST CASE 2

Username: admin

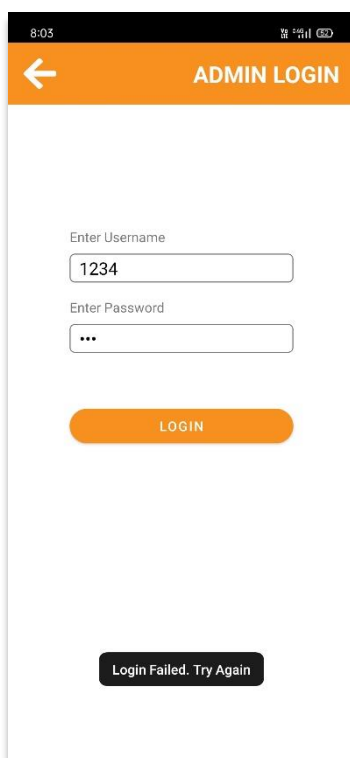
Password: 1234



### TEST CASE 3

Username: 1234

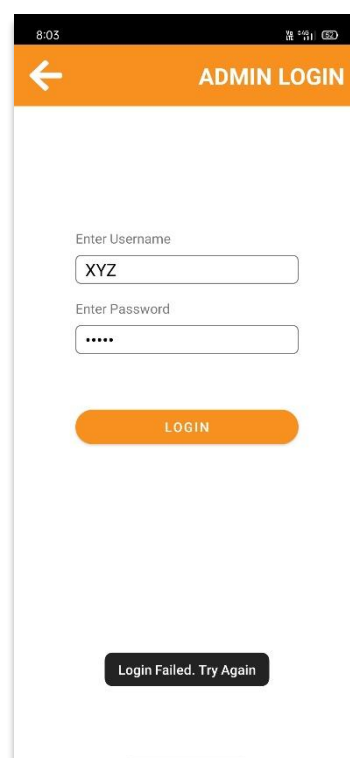
Password: xyz



### TEST CASE 4

Username: XYZ

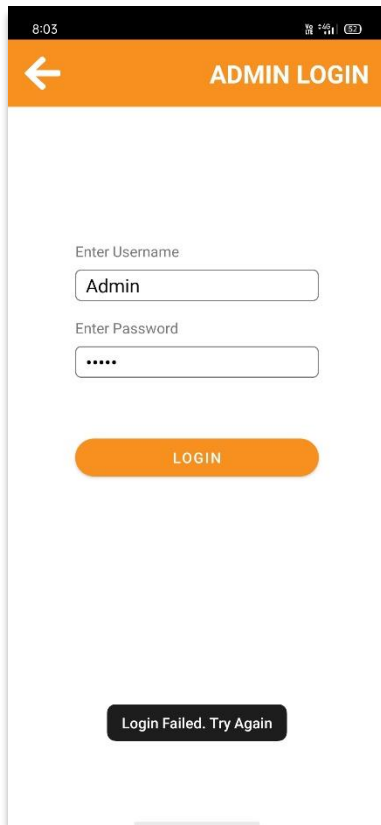
Password: admin



## TEST CASE 5

Username: Admin

Password: Admin



The screenshot shows a mobile application interface for 'ADMIN LOGIN'. At the top, there is a status bar with the time '8:03' and battery level '44%'. Below the status bar is an orange header with a white back arrow and the text 'ADMIN LOGIN'. The main content area is white and contains two input fields: 'Enter Username' with the value 'Admin' and 'Enter Password' with masked characters '\*\*\*\*\*'. Below these fields is an orange 'LOGIN' button. At the bottom, a dark grey message box states 'Login Failed. Try Again'.

8:03 44%

← ADMIN LOGIN

Enter Username  
Admin

Enter Password  
\*\*\*\*\*

LOGIN

Login Failed. Try Again

### 5.3.2 Integrated Testing

- RELATION BETWEEN **ADMIN** AND **ADD** MODULE

| Test Scenario ID | Test Case Description                                    | Preconditions                 | Test Data                                                                     | Postconditions               | Expected Result  | Status  |
|------------------|----------------------------------------------------------|-------------------------------|-------------------------------------------------------------------------------|------------------------------|------------------|---------|
| 1                | Update a product in the database with respective details | Valid data in required fields | Product ID: 8902442231047<br>Product Name: Youth Diary<br>Product Price: 70rs | The database will be updated | Database Updated | Success |
|                  |                                                          |                               |                                                                               |                              |                  |         |
| 2                | Modify Details of a product in the database              | Valid data in required fields | Product ID: 8902442231047<br>Product Name: Youth Diary<br>Product Price: 75rs | The database will be updated | Database Updated | Success |



- **TEST CASE 1**

[illegible]

ADD

Product ID

8902442231047

Product Name

Youth Diary

Product Price (in Rs.)

70

Product Weight/Pages/Volume (with Units)

192 Pages

Product Category

Stationary

Product Description

B6 Size Diary

ADD

Added Successfully

- TEST CASE 2

9:32 9:32

← UPDATE

Product ID

8902442231047

Product Name

Youth Diary

Product Price (in Rs.)

75

Product Weig.ht/Pages/Volume (with Units)

192 Pages

Product Category

Stationary

Product Description

B6 Size Diary

UPDATE

DELETE

9:32 9:32

← UPDATE

Product ID

8902442231047

Product Name

Youth Diary

Product Price (in Rs.)

75

Product Weig.ht/Pages/Volume (with Units)

192 Pages

Product Category

Stationary

Product Description

B6 Size Diary

UPDATE

Updated Successfully!

DELETE

### **5.3.3 Beta Testing**

Beta testing is the best chance to find bugs and usability issues before a product is fully released. While internal testing can uncover many problems, nothing can truly simulate real users trying to complete real tasks.

Additionally, beta testing is the first opportunity to test software in an actual production environment versus a lab or stage setting. This ensures the software can perform under real workloads and that speed, storage, and scalability all work as expected.

Agile development is flexible and high-speed, and therefore requires a beta testing approach that is also fast and flexible. Iterative testing is frequently conducted with a unique set of testers for each iteration, to ensure each participant can engage and provide feedback with open eyes, without the influence of their previous experiences. However, for some products, it makes more sense to re-engage the same set of testers continually, so that each user can receive a build after each sprint and provide updated feedback over time before the final release ships to paying customers.

Beta testers should expect a nearly feature-complete product that may still have some bugs, crashes, and incomplete documentation. The aim is to identify and fix critical/important issues, and suggest user experience improvements that are achievable before launching the product.

Beta testing seeks to improve the success of the upcoming product launch or release, by providing evidential recommendations for product improvement and a comprehensive perspective of customer experience. The feedback collected will not only improve the current release but can also help drive priorities for future releases and ensure the roadmap and planning is as responsive as possible to what's being learned from the market. Beta tester input comes in much larger quantities and often with more detail than typical product feedback, which arrives somewhat randomly and via various channels. Also, future product development is heavily influenced by beta testing outcomes.

Beta testing is a fantastic practice that allows you to test your product with real users before it reaches the market. Though the purpose of beta testing may vary depending on the product, the ultimate goal remains the same – create products that will have an excellent user experience.

## 5.4 Modifications & Improvements

In modern IT, change management has many different guises. Project managers view change management as the process used to obtain approval for changes to the scope, timeline, or budget of a project. Infrastructure professionals consider change management to be the process for approving, testing and installing a new piece of equipment, a cloud instance, or a new release of an application.

When your organization undertakes projects or initiatives to improve performance, seize opportunities or address key issues, they often require changes; changes to processes, job roles, organizational structures, and types and uses of technology.

While all changes are unique and all individuals are unique, decades of research shows there are actions we can take to influence people in their transitions. Change management provides a structured approach for supporting the individuals in your organization to move from their current states to their future states.

How to implement change management:

1. Define the change.
2. Select the change management team.
3. Identify management sponsorship and secure commitment.
4. Develop implementation plan including metrics.
5. Implement the change—in stages, if possible.
6. Collect and analyse data.
7. Quantify gaps and understand resistance.
8. Modify the plan as needed and loop back to the implementation step.

How can Agile change management methodology be put to use for managing change requirements?

- Use Agile Task Boards for Superior Project Tracking

Agile task boards help categorize different development tasks into various categories:

1. To-Do
2. In Progress
3. In Testing
4. Done

This categorization of development tasks gives team members a clear view of where they stand in terms of their progress, and which areas of the project need their attention. There are 3 ways an agile task board can help manage a change or changes in requirements:

- Have clear visibility about the current implementation status of the project and identify how these changes will impact tasks that are pending.
- Identify project requirements that are impacted by changing requirements.
- Keep all team members on the same page with the help of threaded comments. Any changing requirements before and during the sprint can be viewed on the task board with the help of these comments.

### **Modifications & Improvements made in Add to My cart**

- The preferred language is changed to **Java** from **Kotlin**.
- Also, there is no Login module for the **User**, only for the **Admin**.

## 5.5 Test Cases

| Test Scenario ID | Test Case Description                                    | Preconditions                 | Test Data                                                                     | Postconditions             | Expected Result  | Status       |
|------------------|----------------------------------------------------------|-------------------------------|-------------------------------------------------------------------------------|----------------------------|------------------|--------------|
| 1                | Login in with a valid email and valid password           | Valid data in URL variable    | Username: admin<br>Password: admin                                            | User will be Logged in     | Login Successful | Pass         |
|                  |                                                          |                               |                                                                               |                            |                  |              |
| 2                | Login with an incorrect email or password                | Invalid data in URL variable  | Username: admin<br>Password: 1234                                             | User will not be Logged in | Login Failed     | Unauthorized |
|                  |                                                          |                               |                                                                               |                            |                  |              |
| 3                | Update a product in the database with respective details | Valid data in required fields | Product ID: 8902442231047<br>Product Name: Youth Diary<br>Product Price: 70rs | Database will be updated   | Database Updated | Success      |
|                  |                                                          |                               |                                                                               |                            |                  |              |
| 4                | Modify Details of a product in the database              | Valid data in required fields | Product ID: 8902442231047<br>Product Name: Youth Diary<br>Product Price: 75rs | Database will be updated   | Database Updated | Success      |

# Chapter 6: Results & Discussion

## 6.1 Test Reports

- Admin Login

ADMIN LOGIN

Enter

admin

Enter Password

.....

LOGIN

←

+

8901324023916

UNRULED REGISTER

180 Pages | Category: Stationery

Apsara Unruled Blank Register

Size: 29.7 x 21cm

55

8904109420078

GILOY GHANVATI

60 Tablets | Category: Pharmacy

Patanjali Ayurveda

100

8901719114014

FABI JAM-IN

150g | Category: Biscuits

Best Before 6 Months from Packaging.

Nutritional Facts/Information (Amount per 10g approx)

- Energy 462kcal

- Protein 4.8g

- Carbohydrate 78g

of which sugars 37g

- Fat 14.6g

saturated fat 6.7g

trans fat 0g

Date of Manufacture - 13/09/2020

35

8901058873146

MAGGIE FUSIAN - BANGKOK SWEET CHILLI NOODLE

71g | Category: Noodles

Best Before 8 months from Manufacture.

Date of Manufacture - 23/07/2020

20

Login Successful

- Adding Product to Database

←

ADD

Product ID

8904042320015

Product Name

Bluebird Instant Pudding

Product Price (in Rs.)

50

Product Weig.ht/Pages/ Volume (with Units)

100g

Product Category

Desserts

Product Description

Manufacturing Date - August 2020

Best Before - Feb 2022

ADD

←

ADD

Product ID

8904042320015

Product Name

Bluebird Instant Pudding

Product Price (in Rs.)

50

Product Weig.ht/Pages/ Volume (with Units)

100g

Product Category

Desserts

Product Description

Manufacturing Date - August 2020

Best Before - Feb 2022

ADD

Added Successfully

←

+

8901324023916

UNRULED REGISTER

180 Pages | Category: Stationery

Apsara Unruled Blank Register

Size: 29.7 x 21cm

55

8904109420078

GILOY GHANVATI

60 Tablets | Category: Pharmacy

Patanjali Ayurveda

100

8901719114014

FABI JAM-IN

150g | Category: Biscuits

Best Before 6 Months from Packaging.

Nutritional Facts/Information (Amount per 10g approx)

- Energy 462kcal

- Protein 4.8g

- Carbohydrate 78g

of which sugars 37g

- Fat 14.6g

saturated fat 6.7g

trans fat 0g

Date of Manufacture - 13/09/2020

35

8901058873146

MAGGIE FUSIAN - BANGKOK SWEET CHILLI NOODLE

71g | Category: Noodles

Best Before 8 months from Manufacture.

Date of Manufacture - 23/07/2020

20

8904042320015

BLUEBIRD INSTANT PUDDING

100g | Category: Desserts

Manufacturing Date - August 2020

Best Before - Feb 2022

50

Total Data: 5

- Updating Product Information

UPDATE

Product ID

8904109420078

Product Name

Giloy Ghanvati

Product Price (in Rs.)

100

Product Weig.ht/ Pages/ Volume (with Units)

60 Tablets

Product Category

Pharmacy

Product Description

Patanjali Ayurveda

UPDATE

DELETE

UPDATE

Product ID

8904109420078

Product Name

Giloy Ghanvati

Product Price (in Rs.)

100

Product Weig.ht/ Pages/ Volume (with Units)

60 Tablets

Product Category

Pharmacy

Product Description

Patanjali Ayurvedic Medicine

UPDATE

DELETE

Updated Successfully!

- Deleting Product

UPDATE

Product ID

8904109420078

Product Name

Giloy Ghanvati

Product Price (in Rs.)

100

Product Weig.ht/ Pages/ Volume (with Units)

60 Tablets

Delete Giloy Ghanvati ?

Are you sure you want to delete Giloy Ghanvati ?

NO

YES

UPDATE

DELETE

+

8901324023916

UNRULED REGISTER

180 Pages | Category: Stationery

Apsara Unruled Blank Register

Size: 29.7 x 21cm

55

8901719114014

FABI JAM- IN

150g | Category: Biscuits

Best Before 6 Months from Packaging.

Nutritional Facts/Information (Amount per 10g approx)

- Energy 462kcal

- Protein 4.8g

- Carbohydrate 78g

- of which sugars 37g

- Fat 14.6g

- saturated fat 6.7g

- trans fat 0g

Date of Manufacture - 13/09/2020

35

8901058873146

MAGGIE FUSIAN - BANGKOK SWEET CHILLI NOODLE

71g | Category: Noodles

Best Before 8 months from Manufacture.

Date of Manufacture - 23/07/2020

20

8904042320015

BLUEBIRD INSTANT PUDDING

100g | Category: Desserts

Manufacturing Date - August 2020

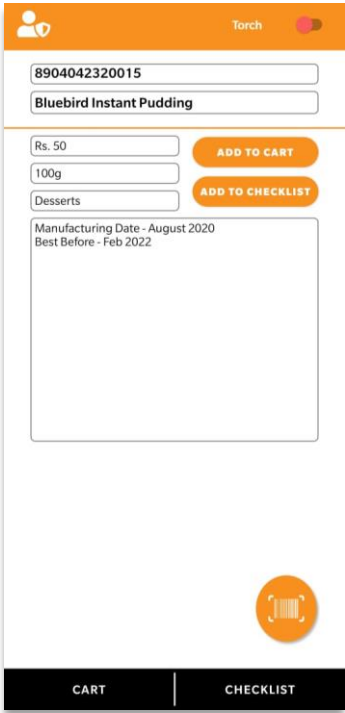
Best Before - Feb 2022

50

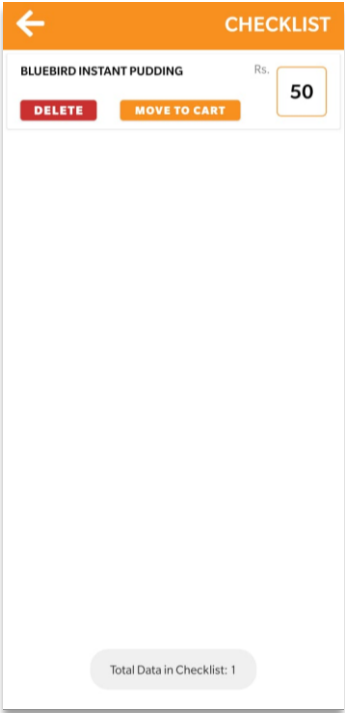
Successfully Deleted.



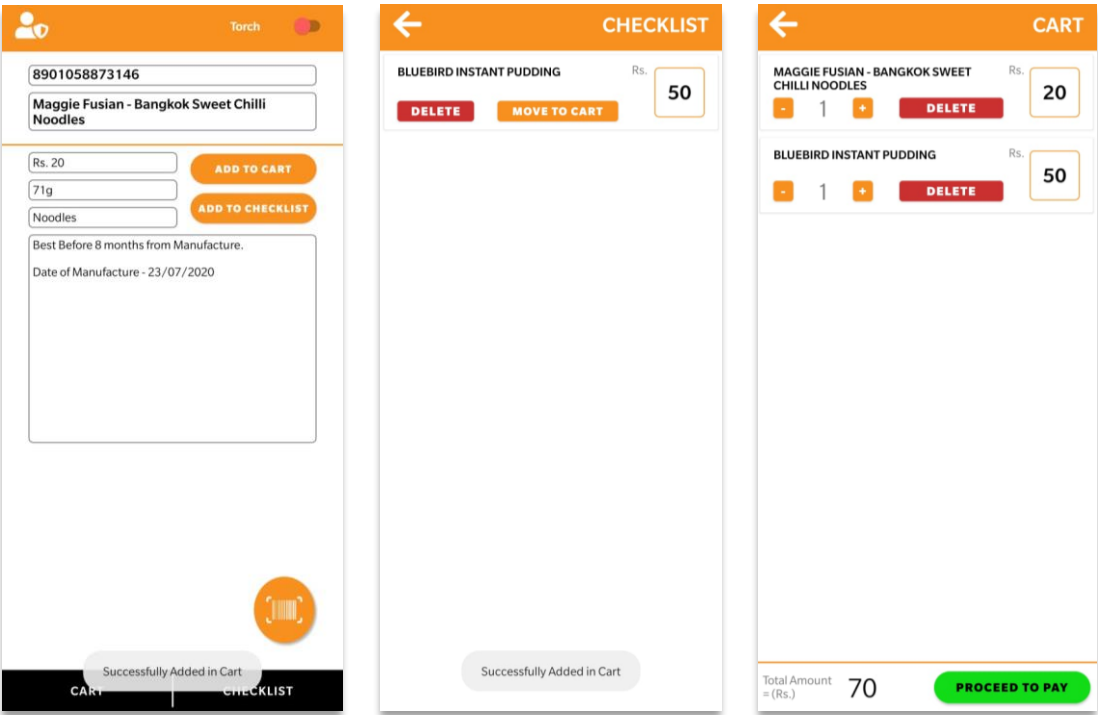
- Scanning a Product's Barcode and Retrieving respective data from the Database to display.



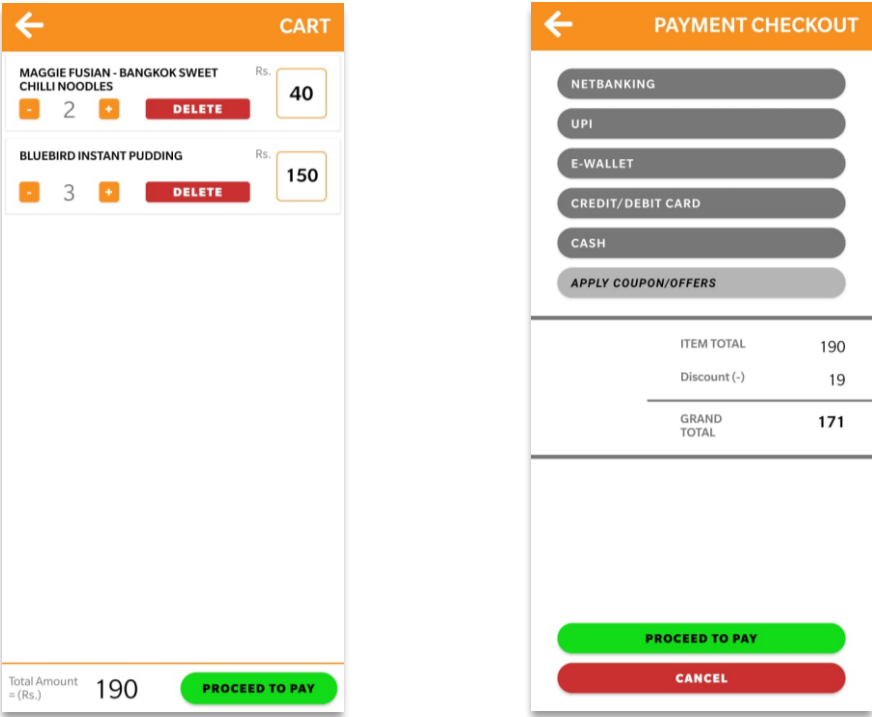
- Adding the product from the scanned result to Checklist.



- Moving that product to the Cart.



- Quantity Update & Proceeding to Payment (Dummy).



## 6.2 User Documentation

User Guide for Add to My Cart application:

Our application is a Smart Shopping System that will help you to have a favourable shopping experience. You can easily scan the barcode of the application and read the product description. Add it to your cart and pay for it as preferred (digital/cash). Additionally, the admin can add products to the database and modify the details or delete the entry as needed.

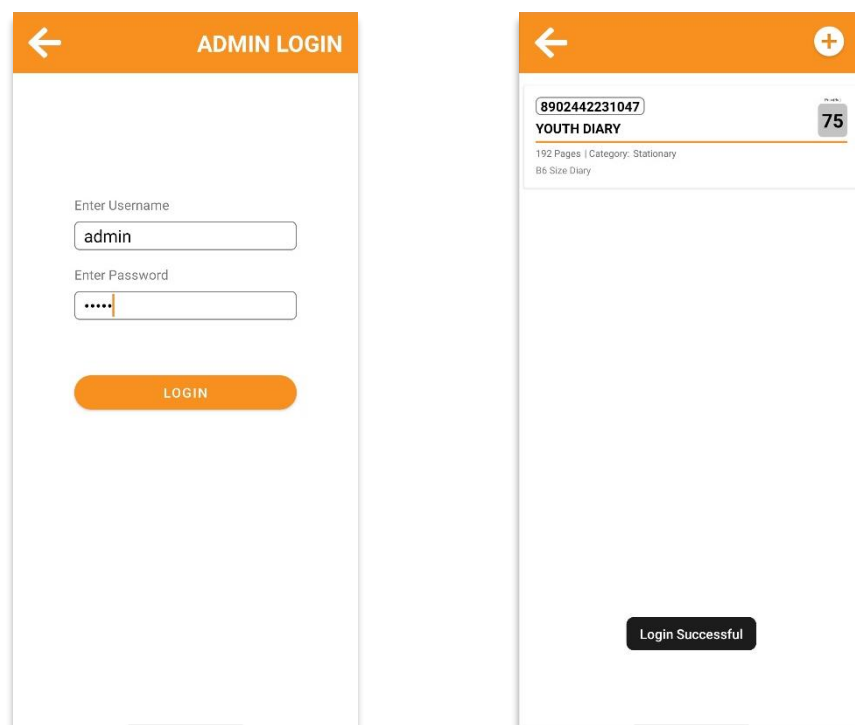
This user guide will provide thorough explanation about Add to My Cart and its features.

NOTE: As soon as you launch the application, it will ask for camera permission, please click on Allow.

### 1. User Guide for Admin.

- **Admin Login**

The Admin can Login by clicking on the top left icon. Then, a username and password have to be entered for authorization. For authorization, the username has to be “admin” and password has to be “admin”. By clicking on Login, the admin will be successfully authorized.



Admin Login is necessary to be able to add products to the database.

- **Adding products to the database with details**

Add the details as asked below, Product ID is the Barcode Number. Then click on Add.

ADD

Product ID

Product Name

Product Price (in Rs.)

Product Weig.ht/Pages/Volume (with Units)

Product Category

Product Description

ADD

82

**DETTOL INSTANT HAND SANITIZER**

50ml | Category: Sanitizer  
Alcohol Disinfectant Hand Sanitizer

- **Modifying details of the product already in the database**

Modify any type of detail as needed, and click on Update.

UPDATE

Product ID

Product Name

Product Price (in Rs.)

Product Weig.ht/Pages/Volume (with Units)

Product Category

Product Description

UPDATE  
DELETE

90

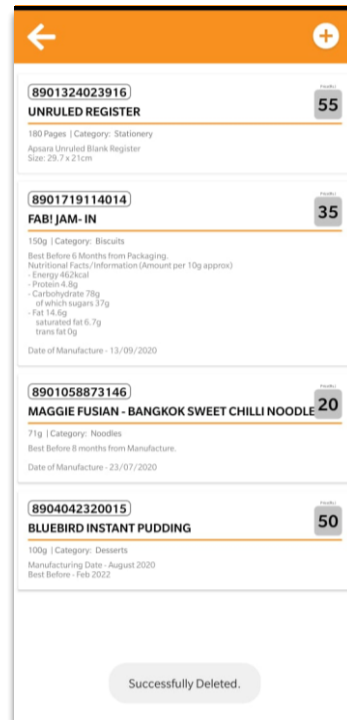
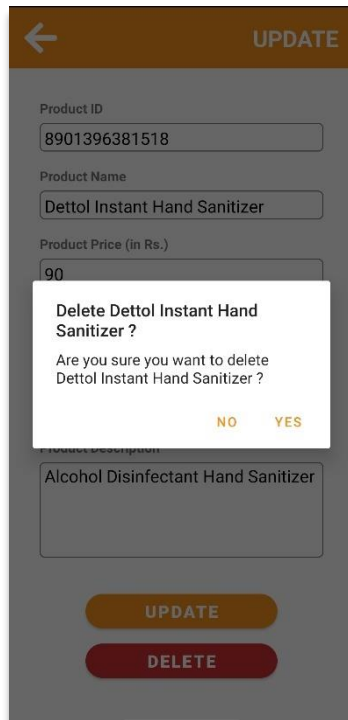
**DETTOL INSTANT HAND SANITIZER**

50ml | Category: Sanitizer  
Alcohol Disinfectant Hand Sanitizer

Total Data: 1

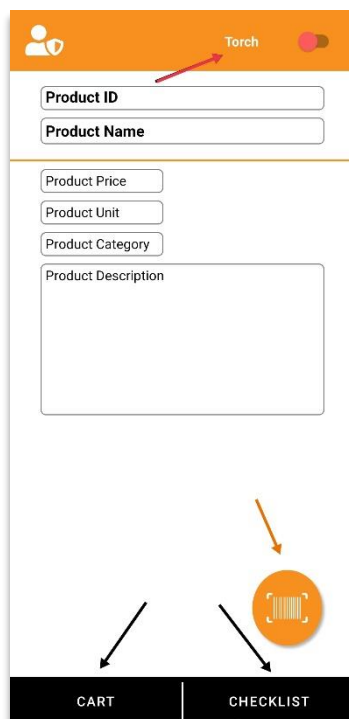
- **Deleting entry of a product from the database**

To delete an entry, click on Delete then select Yes.



## 2. User Guide for Customer

- **Introduction to Home Page**



- The red arrow indicates the toggle button to turn on/off the torch (flashlight).
- The orange arrow indicates the Scanner button.
- The black arrows indicate the Cart and Checklist both.

And the listed text indicates the data fields that will be presented after scanning the

- **Scanning Barcode of products**

By clicking on the Scan button, the camera will launch within the same application.

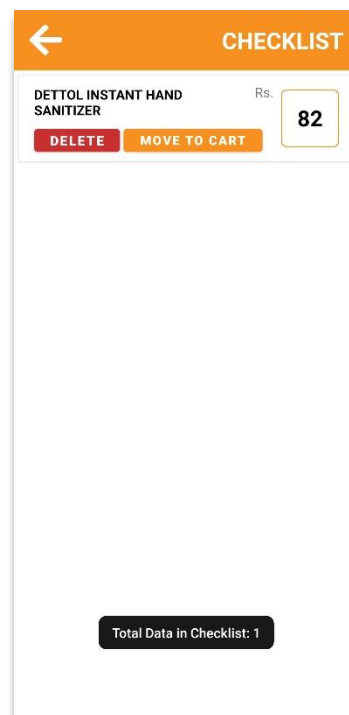
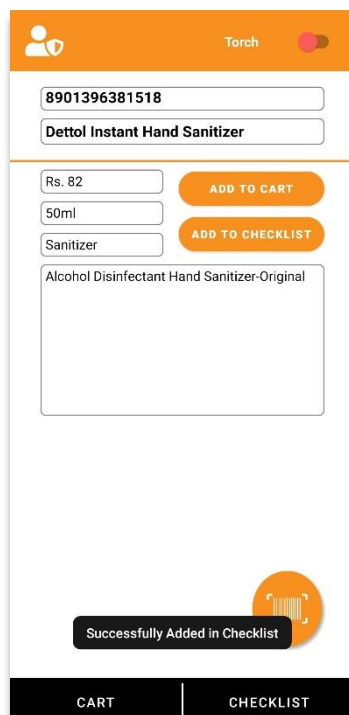
Hold your phone horizontally and scan the barcode of the required product. It will take 2-3 seconds, please don't exit the camera.

When you hear the beeping sound, then the barcode has been scanned.



- **Adding Products to Checklist**

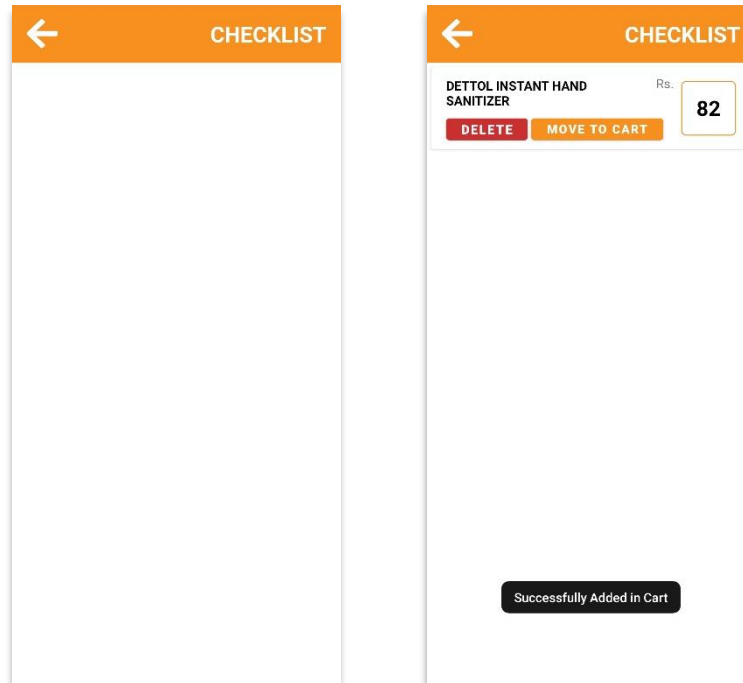
Click on Add to Checklist. Then click on Checklist in the right bottom to view.



- **Deleting Product from Checklist/Moving product to Cart**

To Delete: Click on Delete and the product will be removed from the Checklist.

To Add to Cart from Checklist: Click on Add to Cart, go back, click on Cart and the item will be added.

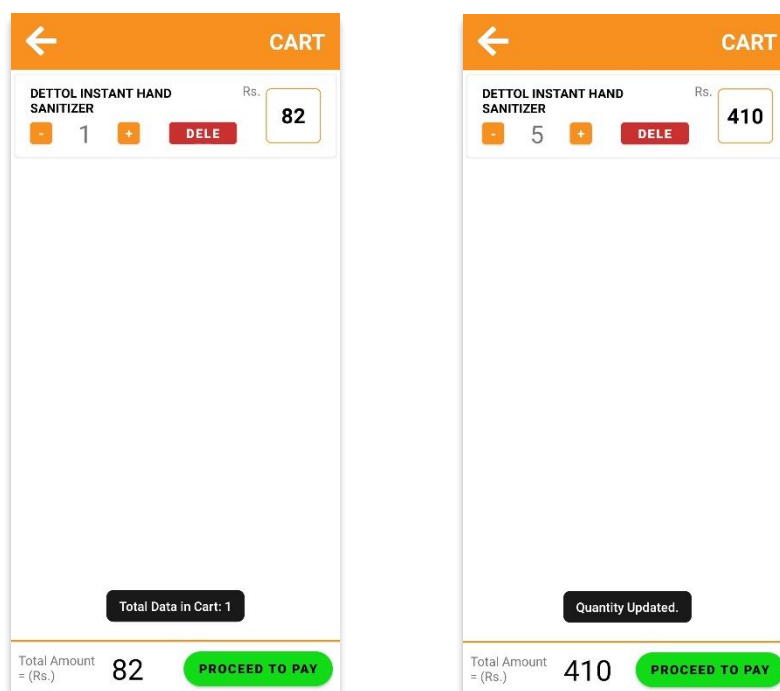


- **Adding to Cart/Changing the quantities**

Click on Add to Cart, then click on Cart in the left bottom and view the cart.

To change the quantity, click on “-” or “+” button.

To delete the item, simply click on Delete.



- **Payment**

Click on Proceed to Pay from the Cart, apply any coupon codes if you have, or choose the payment mode, enter the details, and click on Proceed to pay.

The image shows a mobile app interface for the 'PAYMENT CHECKOUT' screen. At the top, there is an orange header with a back arrow and the text 'PAYMENT CHECKOUT'. Below the header, there is a list of payment methods: NETBANKING, UPI, E-WALLET, CREDIT/DEBIT CARD, CASH, and APPLY COUPON/OFFERS. Each method is represented by a grey button. Below the payment methods, there is a summary table with the following data:

|              |     |
|--------------|-----|
| ITEM TOTAL   | 410 |
| Discount (-) | 41  |
| GRAND TOTAL  | 369 |

Below the summary table, there are two buttons: a green 'PROCEED TO PAY' button and a red 'CANCEL' button.



# Chapter 7: Conclusions

## 7.1 Conclusion

Add to My Cart is an application that focuses on making the shopping experience convenient. It is developed with the purpose of reducing the time a customer must stand in a queue for the billing of the products he/she wishes to purchase.

Adobe XD is used for designing the User Interface. Java is used for developing the front-end to provide the user an interactive interface. SQLite is used as the database engine. XML is used for defining the layout of the application.

In the traditional system, whenever the customer had to purchase products, they had to stand in long queues and wait for their turn for billing. To elaborate on the innovations of the application, the Admin can add products available in the store to the database. Which then, a customer can scan the barcode and view details of the products. Also, the admin can modify the details of the product or even delete the details altogether. In addition to the features available to the customer, he/she can add the required product to the Checklist and then move it to the Cart, or directly add it to the Cart. The customer can also pay using a credit/debit card, even if they are not carrying it but have the details of it.

The efficiency of the application is enhanced after every iteration and with the help of test cases.

### 7.1.1. Significance of the system

The advantages our application has:

- The user can scan the barcode and view the details directly on the application. They need not look for the details on the physical product as often the placement is not fixed on the products, and the font size is too small to read.
- The admin can modify the details of the product without having to delete the previous record and making a new one. He/she can modify the details by selecting a specific product in case there is a typing or input error or incomplete details.
- The user can add the product to the Checklist if they want to purchase it later.
- The user can change the quantity of the product he/she wishes to purchase.
- The user can pay via digital mode and go cashless, and they need not carry credit/debit cards as long as they have the details of the same.

## **7.2 Limitations of the System**

- The application does not have a feature that lets the user sign-up or login.
- Because of the above point, the admin cannot view the details of an order placed by a customer after payment.
- The Admin can log in only with the prevailing password that's programmed.
- The database does not update on Real-Time basis.

## **7.3 Future Scope**

The additional features that can be added to the application in the future to enhance the efficiency, when releases for commercial purpose, are as follows:

A Registration Module for User will be created, and the Login Module for Admin will be improved which will let them create their password.

The Real-Time Database feature will also be included.

A feature that generates a digital receipt of the order shall also be included for both the user and the admin. The digital receipt could be sent to the customer or stored in the profile section of the application once the Registration for User module is created.

Advanced Technology such as the Internet of Things (IoT) can be added to the system. A display device like a screen with a multipurpose single point load cell (weighing sensor) can be attached to the cart that displays the weight of the cart and increases/decreases the number of items as the weight changes.