

WILSON COLLEGE

(Affiliated to University of Mumbai)

MUMBAI-MAHARASHTRA-400007

DEPARTMENT OF INFORMATION TECHNOLOGY



Certificate of Completion

This is to certify that **Mr Rickashe Kenny Williams** has worked and duly completed his Project Work for the degree of **Bachelor of Information Technology** under the Faculty of **Science** in the subject **Software Quality Assurance** and his project is entitled “**A Smart Supermarket Application (Tranquil)**” under my supervision.

I further certify that the entire work has been done by the learner under my guidance and that no part of it has been submitted previously for any Degree or Diploma of any University.

It is his own work and facts reported by his personal findings and investigations.



Name and Signature of
Guiding Teacher

Date of Submission

A SMART SUPER MARKET APPLICATION (TRANQUIL)

A Project Report

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

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

By

Rickashe Kenny Williams
1920ITTY – 68

Under the esteemed guidance of

Dr Pradhnya WanKhade
Assistant Professor



DEPARTMENT OF INFORMATION TECHNOLOGY

WILSON COLLEGE

(Affiliated to University of Mumbai)

MUMBAI, 400007

MAHARASHTRA

2021-2022

PROFORMA FOR THE APPROVAL PROJECT PROPOSAL

PNR No: **2019016400225294**

Roll No: **1920ITTY-68**

1. Name of the Student: **Rickashe Kenny Williams**

2. Title of the Project: **A Smart Supermarket Application (Tranquil)**

3. Name of the Guide: **Pradhnya Wankhade**

4. Teaching experience of the Guide: **15 Years**

5. Is this your first Submission? Yes ☐ No ☐

Signature of the Student

Date:

Signature of the Guide

Date:

Signature of the Coordinator

Date:

DECLARATION

I hereby declare that the project entitled, “A Smart Supermarket Application” done at Wilson College, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project done in partial fulfillment of the requirement for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

Rickashe Kenny Williams

ABSTRACT

This project A Smart Super Market Application (Tranquil) is Mobile Application Designed for Supermarkets. It is developed on the platform named “Android Studio”.

The project is about making life much easier for those who shop at crowded supermarkets. To avoid standing in long ques at each counter and also making payment mode much easier.

Modern handheld devices such as smartphones are become increasingly powerful in the recent years. Dramatic break throughout in processing power along with the number of extra features included in these devices have opened the door to a wide range of commercial possibilities.

Nowadays most of the cell phones include camera, processors and internet access. However, even with all these added abilities, there are few applications that allow passing of the environment information and location-Based services.

The prime objective of this project is to create a full-fledged android application which can scan the barcode of the product just by using your smart phone and making easier to pay for the item.

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to my teacher for their exemplary guidance, monitoring and constant encouragement throughout the course of this project.

The blessings, help and guidance given by them, from time to time, shall carry me long way in the journey of life in which I am about to embark.

I also take this opportunity to express a deep sense of gratitude to Dr. Pradhnya Wankhade for her coordinational support, valuable information and guidance, which helped me in completing this task through various stages.

A large debt of gratitude is owed to my project guide Prof. Srilatha Ratnam who has not only endured, but also encouraged, assisted and inspired me for taking up the project on “A Smart Supermarket Application (Tranquil)”.

I want to acknowledge and thank her for giving me the opportunity to do this under her guidance and also sharing her immense knowledge. Her continuous guidance, time, valuable suggestions, inputs and helpful criticisms have helped me to accomplish such a challenging task.

Lastly, I thank almighty. My parents, family and friends for their constant encouragement, with which I could carry on this project through thick and thin.

TABLE OF CONTENTS

Chapter 1: Introduction Page 1

- 1.1 Background
- 1.2 Objective
- 1.3 Purpose, Scope and Applicability
 - 1.3.1 Purpose
 - 1.3.2 Scope
 - 1.3.3 Applicability
- 1.4 Achievements

Chapter 2: Survey of technologies Page 4

Chapter 3: Requirements and Analysis Page 7

- 3.1 Problem Definition
- 3.2 Requirement and Specification
- 3.3 Planning and Scheduling
- 3.4 Hardware and Software Requirements
 - 3.4.1 Hardware Requirements
 - 3.4.1 Software Requirements
- 3.5 Product Preliminary Description
- 3.6 Conceptual Models
 - Game Flow Chart

Chapter 4: System Design Page 13

- 4.1 Basic Modules
- 4.2 Procedural Design
- 4.3 User Interface
- 4.4 Security Issues
- 4.5 Test Cases Design

Chapter 5: Implementation and TestingPage 20

- 5.1 Implementation Approach
- 5.2 Coding Details
- 5.3 Testing Approach
 - 5.3.1 Unit Testing
 - 5.3.2 Integrated Testing

Chapter 6: Results and Discussion Page 42

- 6.1 User Documentation

Chapter 7: Conclusion and Future plans Page 47

Chapter 1

Introduction

1.1 Background:

The Smart Supermarket Application (Tranquil) will be a very helpful application in the current scenario as we are dealing with the covid-19 pandemic and social distancing is a major concern in the current time.

As we are moving further with Technology, we are getting more and more introduced to the online environment. The online environment is good opportunity wherein people use it to the extent.

In today's world technology helps to build and grow the commerce and business sector and generate maximum possible outputs. The time taken by different sectors to generate business is now minimized with an advancement in information technology.

Modern Technology provides electronic security, storage and efficient communication. To run the business in the right way and generate expected outcomes, computer, software and the internet helps a lot.

When we come to the point of technology, the basic modern technology device that everyone uses is the Mobile phone. Smartphone is the basic advanced technology that is widely used in today's world.

So, we are making the use of these modern Smartphones devices to provide the safest and the easiest way for the customers to pay for their product they wish to purchase.

This project helps you scan the products barcode using your smartphone device and then allows you to pay for the product using online payment method.

This project is Android based application specially designed and developed for the customers in superstore.

1.2 Objective:

This Project was Basically Designed and Developed on the basics of the ongoing Covid-19 Situation. Where Social distancing is a major concern.

The main objective of this project is to develop an application for the customers who physically shop in Supermarkets. To avoid Standing in long Ques at the cash counter and also making payment made much easier.

This application consists of Scan and Pay method using your android mobile phone. Using this application, the customer can scan the product barcode from their mobile device and pay using any online payment method.

Using this application, the customers can also avoid standing in long ques at the superstore. This application will provide a faster checkout and safe payment method.

This application will have a user-friendly interface which will make is easier and convenient for all age groups.

1.3 Purpose, Scope and Applicability:

1.3.1 Purpose:

- The goal of this project is making life much simpler and easier in this ongoing covid-19 situation.
- This android application helps customers to avoid waiting in long ques in crowded supermarkets.
- This application also allows the customers to scan the barcode by using android mobile phone.

1.3.2 Scope:

- As the population of India is around 1.32 billion in present and is growing very fast. It is possible that the necessity of food is also going to increase. these leads to various life problems.
- One of these problems is overcrowding of superstores. To avoid this problem, we have come up this android mobile application “Tranquil”.

- Thus, in this field of growing technology we should take the full benefits of the growing technical industry. Thus, we have developed the technology which will help the people and also make life easy.

1.3.3 Applicability:

- This android application is designed and build for customers at supermarkets.
- The purpose of this project is to make use of technology and manpower to build a full-fledged android application.
- The features of this android application are:
 - Easy to access
 - User friendly interface
 - Reliable and coordinative
 - Simple registration
 - Login process
 - Registered info-based data analysis
 - Easy to handle

1.4 Achievements:

After completing this project. I gained a lot of practical knowledge about Android mobile programming, Java, Firebase. The latest version of android studio has a lots off new feature which makes it easier for programmers. The topic of inheritance played a very important role in this project.

Working on Android Studio and Firebase for the last few months have been amazing. To design the UI in android studio now has the features of drag and drop, which makes designing in android studio more comfortable.

Chapter 2

Survey of Technologies

Java Language:

Most of the android application are developed using the java language. Android is an open source and Linux-based operating system for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies.

Java is a very popular programming language. It was originally developed by James Gosling at Sun Microsystems and released in may 1995 as a core component of Sun Microsystems Java platform. This language is now owned by Oracle Corporation.

This language was developed long after C and C++, Java incorporates many of the powerful features of those powerful language while addressing some of their drawbacks.

Java is a high-level, class-base, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is general-purpose programming language intended to let programmers write once, run anywhere, meaning that compiled Java code can run on all platforms that support Java without the need to recompile.

The syntax of Java is similar to C and C++, but has fewer low-level facilities than either of them. The Java runtime provides dynamic capabilities (such as reflection and runtime code modification) that are typically not available in traditional compiled languages.

Some of Java's important core features are:

- It's easy to learn and understand.
- It's designed to be platform independent, secure and robust.
- It's object-oriented.

Android relies heavily on these Java fundamentals. The Android SDK includes many standard java libraries (Data Structure, Maths Libraries, Graphics Libraries, Networking libraries and everything else you could want) as well as special Android libraries that will help develop awesome android application.

There are number of ways to create apps for android devices, but recommended method for most developers is to write native apps using Java and the android SDK. Java for Android apps are both similar and quite different from other types of Java applications.

If you have experience with Java (Or Similar language) they you'll probably feel pretty comfortable diving right into the code and learning how to use the android SDK to make your app.

Firestore Database:

Firestore is platform developed by Google for creating mobile and web applications. It was originally a independent company founded in 2011. In 2014, Google acquired the platform and it is now their flagship offering for app development.

Firestore provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment.

Firestore offers number of services including:

1. **Analytics:** Google analytics for firestore offers free, unlimited reporting on many as 500 separate events. Analytics presents data about user behaviour in IOS and Android apps, enabling better decision- making about improving performance and app marketing.
2. **Authentication:** Firestore authentication makes it easier for developers to build secure authentication systems and enhance the sign-in and onboarding experience for users. This feature offers a complete identity solution, supporting email and password accounts, phone auth, as well as Google, Facebook, GitHub, Twitter Login and More.
3. **Cloud messaging:** Firestore Cloud Messaging (FCM) is a cross-platform messaging tool that lets companies reliably receive and deliver messages on iOS, Android and the web at no cost.
4. **Realtime database:** The Firestore Realtime Database is a cloud-hosted NoSQL database that enables data to be stored and synced between users in real time. The data is synced across all clients in real time and is still available when an app goes offline.
5. **Crashlytics:** Firestore Crashlytics is a real-time crash reporter that helps developers track, prioritize and fix stability issues that reduce the quality of their apps. With crashlytics,

developers spend less time organizing and troubleshooting crashes and more time building features for their apps.

6. Performance: Firebase Performance Monitoring service gives developers insight into the performance characteristics of their iOS and Android apps to help them determine where and when the performance of their apps can be improved.

7. Test lab: Firebase Test Lab is a cloud-based app-testing infrastructure. With one operation, developers can test their iOS or Android apps across a variety of devices and device configurations. They can see the results, including videos, screenshots and logs, in the Firebase console.

Firebase use cases include:

- Create onboarding flows – developers can give users a quick, intuitive sign-in process using Firebase Authentication. They allow users to sign into their apps via their Google, Twitter, Facebook or GitHub accounts in less than five minutes. Developers can also track each step of their onboarding flows to enhance the user experience. Additionally, developers can use Google Analytics for Firebase to log events at each step of their onboarding flows, create funnels to determine where users are dropping off and use remote configuration to make changes to their apps to see how those changes affect conversions.
- Customize a “welcome back” screen – developers can use personalization to give every user the best experience by customizing the initial screen based on a user’s preferences, usage history, location or language. Developers can define audiences based, in part, on user behaviours and show targeted content to each audience.
- Progressively roll out new features – developers can launch new features with minimal risk by first testing those features on a few users to see how they work and how users respond. Then, when developers are satisfied, they can roll out their apps to the rest of their users.

Chapter 3

Requirements and Analysis

3.1 Problem Definition:

Due to the rapid growth in technology. It becomes important we make the best use of this technology.

Due to the ongoing pandemic of widely spread Covid-19 virus. it is become import to maintain social distancing in crowded areas such as markets, mall, superstores etc.

As the population of India is around 1.32 billion in present. It is possible that the necessity of food is also going to increase, these leads to various life problems.

To manage time and reduce the long waiting period at the cash counter at the superstore we need to make the best use of this technology.

Due to the overcrowded supermarkets the number of covid infected cases can also increase. Thus, it necessary we make the best use of technology.

3.2 Requirement Speciation:

- The app to be developed must act on an easily platform that would be able to provide Internet access for the payment process.
- The app should be a light-weight and that supports all mobile platforms so it could be installed on every device in the market that a person can use and make his life much easier.
- The app should be accurate and real-time.
- The app should all time strong internet connectivity to make the payment process with lots of security.

3.3 Planning and Scheduling

The Software Development Life-Cycle model being used for the development of this project is the Agile model.

In software development, Agile model practices include requirements discovery and solutions improvement through the collaborative effort of self-organizing and cross-functional teams with their customers/end users, adaptive planning, evolutionary development, early delivery, continual improvement, and flexible responses to changes in requirements, capacity, and understanding of the problems to be solved.

Agile SDLC model is a combination of iterative and incremental process models with the processes of adaptability and customers satisfaction by rapid delivery of working software products.

Agile model breaks the products into small incremental build. These builds are provided by iterations. Each iteration usually last for one to three weeks. Every iteration involves cross functional teams working continuously on various areas like

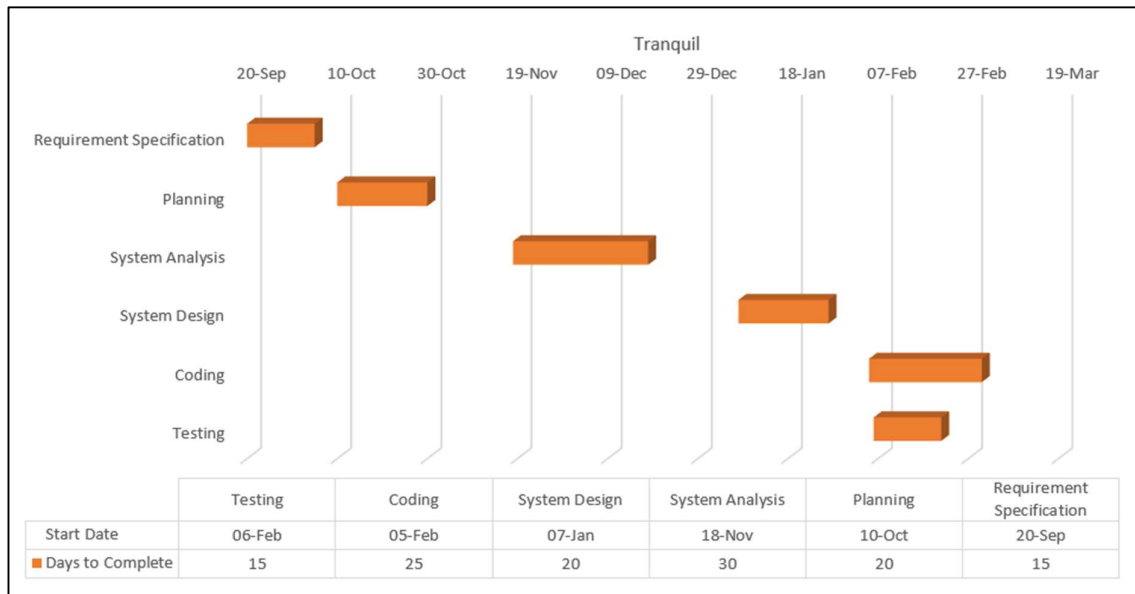
- Planning
- Reporting analysis
- Design
- Coding
- Unit testing
- Acceptance testing

At the end iterations, a working product is displayed to the customer and the important stakeholders.

Agile model is being widely accepted in the software world recently. The main advantages of an Agile SDLC are

- Functionally can be developed rapidly and demonstrated.
- Suitable for fixed or changing requirement.
- Delivers early partial working solution.

Gantt Chart



3.4 Hardware and Software Requirements

3.4.1 Hardware Requirement

Hardware Requirements Minimum hardware required for running Android Studio smoothly and efficiently is

1. Microsoft Windows 10 Operating System (32 or 64 bit).
2. 12 GB RAM minimum, 8 GB RAM recommended.
3. 600 MB hard disk space plus at least 1 GB for Android SDK, emulator system images, and caches.
4. Latest Version of Java Development Kit (JDK).
5. Optional for accelerated emulator: Intel processor with support for Intel VT-x, Intel EM64T (Intel 64), and Execute Disable (XD) Bit functionality.

3.4.2 Software Requirement

1. Microsoft Windows 10 Operating System (32 or 64-bit).
2. The Android Emulator only supports 64-bit Windows.

3. 12 GB RAM minimum, 8 GB RAM recommended.
4. 2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image).
5. 1280 x 800 minimum screen resolution.
6. Android Studio Software

If you do not need Android Studio, you can download the basic Android command line tools. You can use the included SD manager to download other SDK packages. These tools are included in Android Studio.

Offline components:

Download the latest versions of the Android Gradle plugin and Google Maven dependencies to build your project offline.

❖ Android Gradle plugin-

The Android Studio build system is based on Gradle, and the Android Gradle plugin adds several features that are specific to building Android apps. Although the Android plugin is typically updated in lock-step with Android Studio, the plugin (and the rest of the Gradle system) can run independent of Android Studio and be updated separately.

❖ Google Maven dependencies-

The Gradle build system in Android Studio makes it easy to include external binaries or other library modules to your build as dependencies. The dependencies can be located on your machine or in a remote repository, and any transitive dependencies they declare are automatically included as well.

❖ Android Emulator-

Android Emulator is included with Android Studio. Versions of the emulator prior to 25.3.0 were distributed as part of the Android SDK Tools. To ensure you have the latest version, check the SDK Manager for updates. For Android Emulator versions prior to 25.3.0, see the notes. For details of bugs fixed in each release, see the Android Studio release updates blog.

3.5 Product Preliminary Description

Successful use of a self-checkout system can differentiate a retailer from competitors, optimise the checkout process, and attract more customers. Self-checkout solutions have been key to retail success for years, especially for grocery stores and supermarkets.

Everyone knows online shopping has never been easier or more accessible. But despite the surge in e-commerce capabilities, mobile applications other technology advancement including voice-activated shopping and the proliferation of Amazon Dash buttons the majority of Indian consumers still want the tactile experiences offered by physical stores.

The ability to see, touch and feel products as well as take items home immediately rank highest among the reasons consumers choose to shop in stores versus online, according to Retail Dive's Consumer Survey.

For the first question in a six-part series looking at consumer shopping habits, we surveyed 1,425 consumers via Google Surveys about the reasons why they choose to shop in stores over online.

When comparing self-checkout systems, the main aspects you should take into account are cost of space, their efficiency, and the customer experience they create.

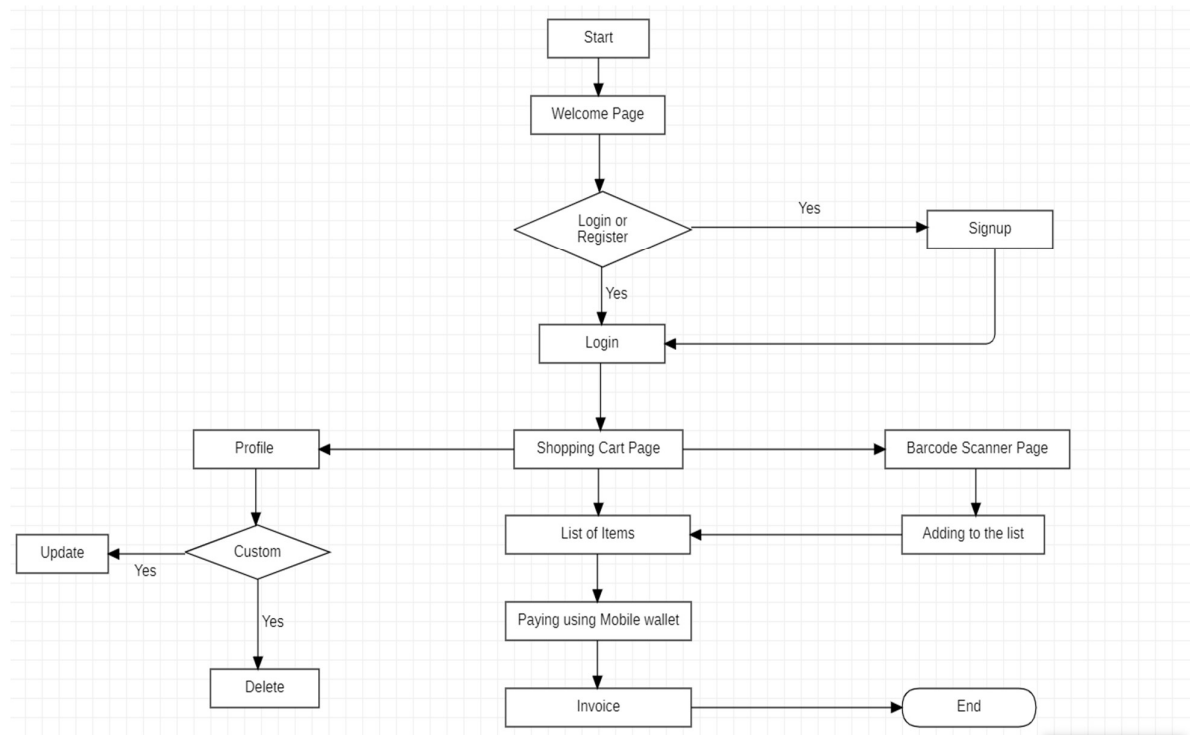
The main advantage of this solution is that it provides a real-time overview of how much all the items picked up cost. This is something that contributes to the practicality and enjoyment of this solution.

The speed of transaction, perceived control and reliability are also quite high when it comes to this system.

When it comes to cost of space, checkout counters definitely take up less space than registers. Since self-checkout is typically faster than registers, there is typically one queue per several checkout counters, which saves additional space.

3.6 Conceptual Model

Game Flow Chart



Chapter 4

System Design

4.1 Basic Modules:

Features

This program is divided into nine modules. i.e., Start-up Logo, Main menu, login, Signup, Barcode Scanner, Cart view, Profile, Checkout, Invoice.

Start-up Logo:

This is the into page. Where I have created my own logo and UI design. The Logo design is developed in Canva, it's a free application software, the logo was created to make the project interesting and appealing. The start-up logo page will only be visible for 5 sec, after which the user will be directed to the main menu page.

Main menu:

After the successful appearance of the start-up page the user is directed to the Main menu page. On the page the user will be Greeted and will be the given the option to Register or Login. On click the user will be directed to the desired page.

Login:

The login page simply consists of Email and Password. The user can login using their registered Email ID and Password.

Sign-up:

The Sign-up page simply consists of the specific user details. i.e., Full name, Email, Username, Password. The user can fill in with their original details.

Cart view:

On successful login the user will be directed to a cart view page, where the user can store the scanned barcode data and can successfully checkout.

Profile:

The cart view page consists of an option where the user can check it profile and successfully update it time to time.

Barcode scanner:

The cart view page consists of an option where the user can scan the barcode data and prepare the list of items the user wishes to purchase in the cart view page.

Checkout:

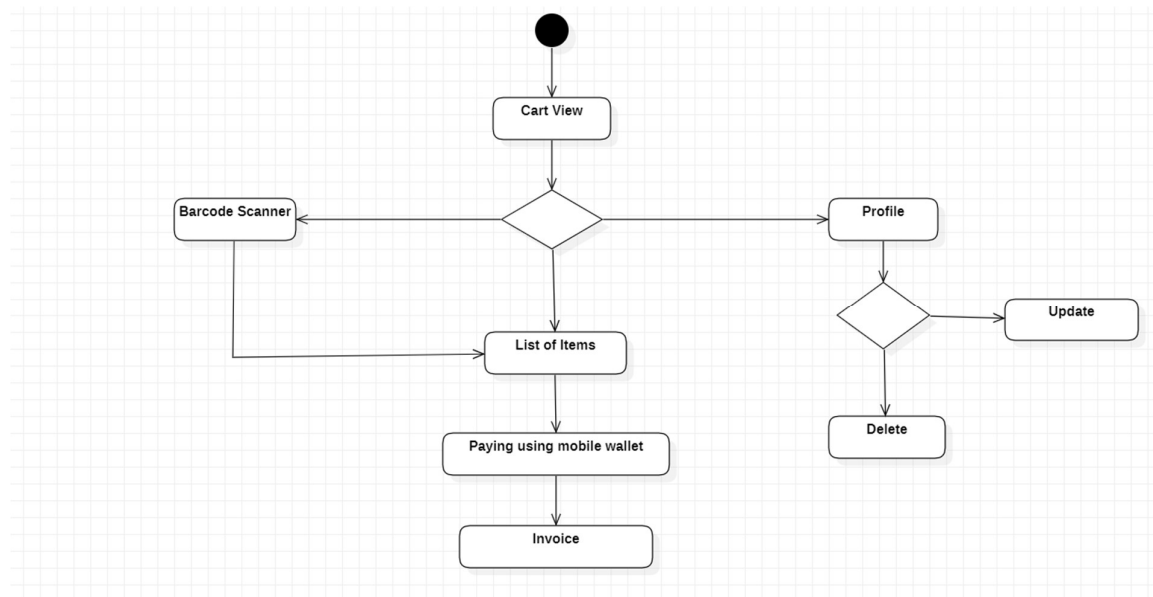
After a successfully scanning the barcode the user can will have a checkout option where the user can pay using their mobile wallet.

Invoice:

After the checkout is successful the user will be directed to the final invoice page where there will the list of items the user has purchased.

4.2 Procedural Design:

Activity Diagram



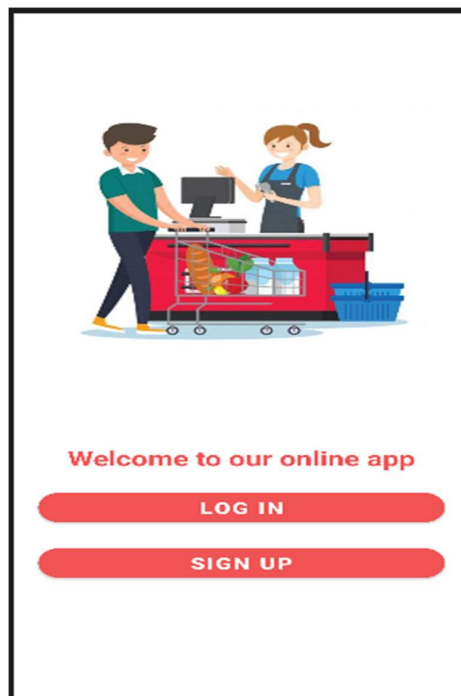
4.3 User Interface Design:

The Android application Tranquil consist of six user Interface.


➤ Intro Page



➤ Main Menu Page



➤ Login Page



Login

Email :

Password :

LOG IN

SIGNUP HERE

➤ Signup Page

SignUp

Full Name :

Email :

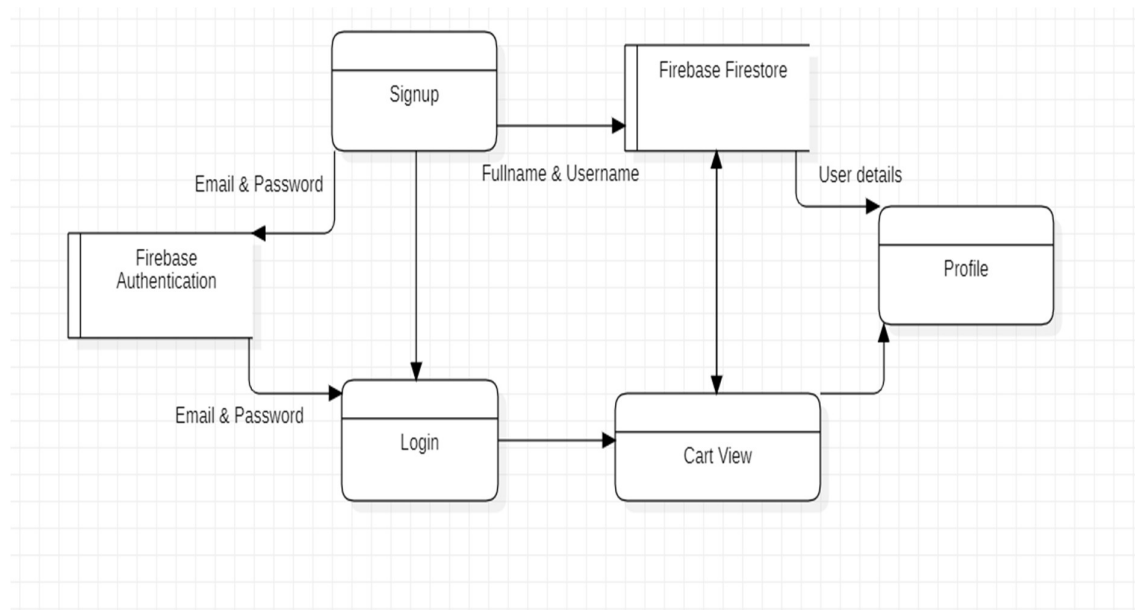
Username :

Password :

SIGN UP

LOG IN

Data Flow Diagram



4.4 Security Issues

There are some security issues in Tranquil Application:

- Privacy: information exchanged must be kept from unauthorized parties
- Integrity: the exchange information must not be altered or tampered with
- Authentication: both sender
- and recipient must prove their identities to each other
- Non-repudiation: proof is required that the exchanged information was indeed received

However, an active data connection is needed to access my application. These are some of the steps to take care so that data is properly stored, and correct user get access to his account.

1. During the signup, if the user enters invalid details
2. Email address, the format of email address should be followed or else he gets an error message.

1. Leakage of important credentials

The safety of password of firebase database, Wi-Fi password is very important. Using this information, the system can be hacked, and the system will be compromised.

And if the attacker gets the APL key of the database, then they can delete the data or even worse changes the data and the administrator wont's able to notice.

This can overcome by not sharing password with unauthorized people and increasing the strength of the password.

The APL is by default compiled and encrypted before it is loaded into the device and reverse engineering it is very difficult.

2. No Internet Connection

Unavailability of internet can cause the system to report inaccurate information like incorrect location of pickup and drop point.

This can be overcome by making that as no internet connectivity in the application this will allow user to make informed decision.

3. Malfunctioning of software

The application may stop working at some point in time. So, it is not easy for technician to detect which software is not working.

The application has multiple modules. Modules will be divided in as a way that it will be easy to detect where the error.

4.5 Test Cases Design

Software testing is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is Defect free.

It involves execution of a software component or system component to evaluate one or more properties of interest.

Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual requirements.

It can be either done manually or using automated tools. Some prefer saying Software testing as a White Box and Black Box Testing.

In simple terms, Software Testing means Verification of Application Under Test (AUT).

Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause monetary and human loss.

Types of Software Testing Typically Testing is classified into three categories.

Functional Testing

Non-Functional Testing or Performance Testing

Maintenance (Regression and Maintenance)

Analyse your test results thoroughly:

Do not ignore any test results. The final test result may be 'pass' or 'fail' but troubleshooting the root cause of 'fail' will give you the solution to the problem. Testers will be respected if they not only log the Bugs but also provide solutions.

Learn to maximize the Test Coverage each time you test any application. 100% test coverage might not be possible but still, try to reach near it.

In order to ensure maximum test coverage, Application is divided into smaller functional modules. Write test cases on such individual unit modules.

Also, if possible, break these modules into smaller parts. It is also known as Application Under Test (AUT).

Chapter 5

Implementation and Testing

5.1 Implementation Approach

In the Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements.

This process is then repeated, producing a new version of the software at the end of each iteration of the model. The advantage of this model is that there is a working model of the system at a very early stage of development, which makes it easier to find functional or design flaws.

Finding issues at an early stage of development enables to take corrective measures in a limited budget. The disadvantage with this SDLC model is that it is applicable only to large and bulky software development projects.

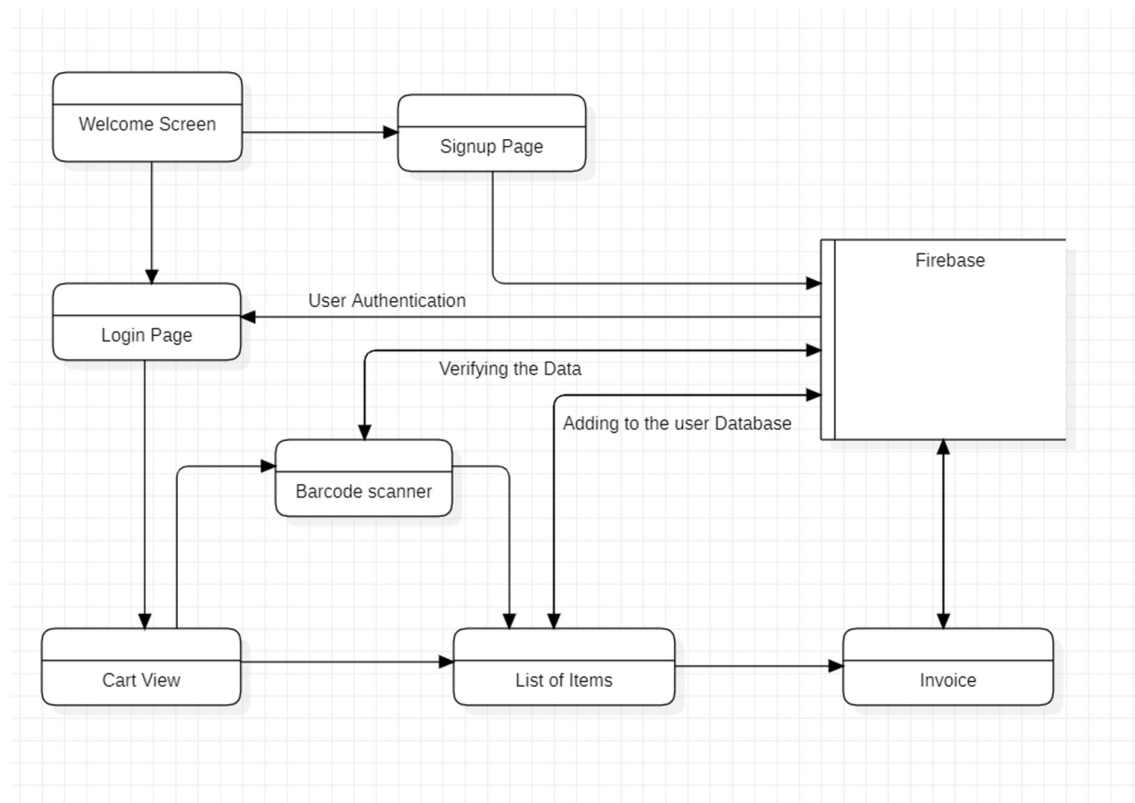
This is because it is hard to break a small software system into further small serviceable increments/modules.

The advantages of the Iterative and Incremental SDLC Model are as follows

- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Testing and debugging during smaller iteration is easy.

- Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.
- Easier to manage risk - High risk part is done first.
- With every increment, operational product is delivered.
- Issues, challenges and risks identified from each increment can be utilized/applied to the next increment.
- Risk analysis is better.
- It supports changing requirements.
- Initial Operating time is less.
- Better suited for large and mission-critical projects.
- During the lifecycle, software is produced early which facilitates User valuation and feedback.

5.2 Coding Details



5.2 CODING DETAILS AND CODE EFFICIENCY

Signing in using Custom Sign in method

```
package com.example.tranquilasmartsupertmarketapllication;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.button.MaterialButton;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.HashMap;
import java.util.Map;

public class SignupPage extends AppCompatActivity {

    public static final String TAG = "Added";

    EditText mFullName, mEmail, mPassword, mUsername;
    MaterialButton mSignupBtn;

    MaterialButton mLoginBtn;

    FirebaseAuth fAuth;
    FirebaseFirestore fStore;

    ProgressBar progressBar;
    String UserID;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_signup_page);

        mFullName = findViewById(R.id.fullname);
        mEmail = findViewById(R.id.email);
        mPassword = findViewById(R.id.password);
        mUsername = findViewById(R.id.username);
        mSignupBtn = findViewById(R.id.buttonSignUp);
        mLoginBtn = findViewById(R.id.loginText);
    }
}
```

```

fAuth = FirebaseAuth.getInstance();
progressBar = findViewById(R.id.progress);
fStore = FirebaseFirestore.getInstance();

mSignupBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String email = mEmail.getText().toString().trim();
        String password = mPassword.getText().toString().trim();
        String Fullname = mFullName.getText().toString();
        String Username = mUsername.getText().toString();

        if (TextUtils.isEmpty(email)) {
            mEmail.setError("Email is required");
            return;
        }

        if (TextUtils.isEmpty(password)) {
            mPassword.setError("Password is required");
            return;
        }

        if (password.length() < 6) {
            mPassword.setError("Password must be >=6 characters");
            return;
        }

        progressBar.setVisibility(View.VISIBLE);

        fAuth.createUserWithEmailAndPassword(email, password).addOnCompleteListener(
            new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task)
            {
                if (task.isSuccessful()) {
                    Toast.makeText(SignupPage.this, "User
Created.", Toast.LENGTH_SHORT).show();
                    UserID = fAuth.getCurrentUser().getUid();
                    DocumentReference documentReference =
fStore.collection("Users").document(UserID);

                    Map<String, Object> UserInfo = new HashMap<>();
                    UserInfo.put("Fullname", Fullname);
                    UserInfo.put("Username", Username);

                    documentReference.set(UserInfo).addOnSuccessListener(new
                    OnSuccessListener<Void>() {
                        @Override
                        public void onSuccess(Void aVoid) {
                            Log.d(TAG, "OnSuccess: user Profile is
created for " + UserID);
                        }
                    }).addOnFailureListener(new OnFailureListener()
            {
                @Override
                public void onFailure(@NonNull Exception e)
            {
                Log.w(TAG, "Error writing document",
e);

```

```

        }
    });
    startActivity(new
Intent(getApplicationContext(), MainActivity.class));

    }else {
        Toast.makeText(SignupPage.this, "Error !" +
task.getException().getMessage(), Toast.LENGTH_SHORT).show();
    }
    }
    });
}
});

mLoginBtn.setOnClickListener(v -> {
    Intent intent=new Intent(SignupPage.this, LoginPage.class);
    startActivity(intent);
});

}
}

```

SignUp

Full Name :

Email :

Username :

Password :

SIGN UP

LOG IN

Logging in using Email and Password Codes

```
package com.example.tranquilasmartsupertmarketapllication;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.button.MaterialButton;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.FirebaseFirestore;

public class LoginPage extends AppCompatActivity {

    EditText mEmail, mPassword;
    MaterialButton mLogInBtn;
    MaterialButton mSignUpBtn;
    FirebaseAuth fAuth;
    ProgressBar progressBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login_page);

        mEmail = findViewById(R.id.email);
        mPassword = findViewById(R.id.password);
        progressBar = findViewById(R.id.progress);
        fAuth = FirebaseAuth.getInstance();
        mLogInBtn = findViewById(R.id.buttonLogin);
        mSignUpBtn = findViewById(R.id.signUpText);

        mLogInBtn.setOnClickListener(view -> {
            String email = mEmail.getText().toString().trim();
            String password = mPassword.getText().toString().trim();

            if (TextUtils.isEmpty(email)) {
                mEmail.setError("Email is required");
                return;
            }

            if (TextUtils.isEmpty(password)) {
                mPassword.setError("Password is required");
                return;
            }

            if (password.length() < 6) {
                mPassword.setError("Password must be >=6 characters");
                return;
            }
        });
    }
}
```

```

    }

    progressBar.setVisibility(View.VISIBLE);

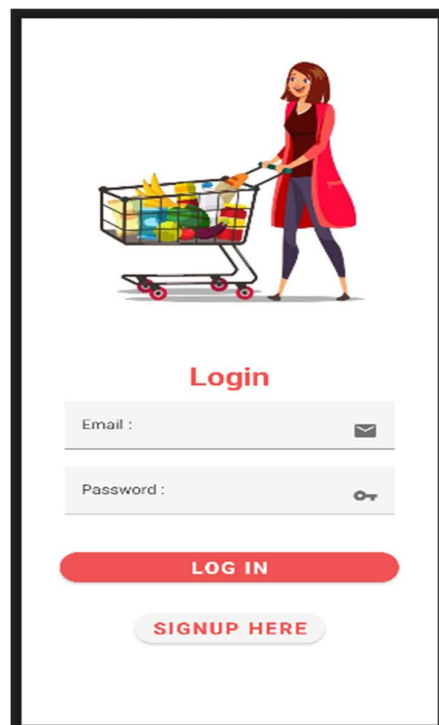
    // authenticate the User

    FirebaseAuth.signInWithEmailAndPassword(email,password).addOnCompleteListener(task
-> {
        if(task.isSuccessful()){
            Toast.makeText(LoginPage.this, "Logged In
Successfully.",Toast.LENGTH_SHORT).show();
            startActivity(new Intent(getApplicationContext(),
BarcodeScanner.class));
        }
        else{
            Toast.makeText(LoginPage.this, "Error !" +
task.getException().getMessage(), Toast.LENGTH_SHORT).show();
        }
    });
});

FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
if (user != null) {
    // FirebaseAuth.getIdToken() instead.
    String uid = user.getId();
}

mSignUpBtn.setOnClickListener(v -> {
    Intent intent=new Intent(LoginPage.this, SignupPage.class);
    startActivity(intent);
});
}
}

```



Barcode Scanner Page

```
package com.example.tranquilasmartsupertmarketapllication;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.media.AudioManager;
import android.media.ToneGenerator;
import android.os.Bundle;
import android.util.Log;
import android.util.SparseArray;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.android.gms.vision.CameraSource;
import com.google.android.gms.vision.Detector;
import com.google.android.gms.vision.barcode.Barcode;
import com.google.android.gms.vision.barcode.BarcodeDetector;
import com.google.api.LabelDescriptor;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;
import com.google.firebase.firestore.v1.Value;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class Barcodescanner extends AppCompatActivity{

    public static final String TAG = "Added";
    private SurfaceView surfaceView;
    private BarcodeDetector barcodeDetector;
    private CameraSource cameraSource;
    private static final int REQUEST_CAMERA_PERMISSION = 201;
    private ToneGenerator toneGen1;
    private TextView barcodeText;
    private String barcodeData;
    Button Send_button;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_barcodescanner);
        toneGen1 = new ToneGenerator(AudioManager.STREAM_MUSIC, 100);
        surfaceView = findViewById(R.id.surface_view);
    }
}
```

```

barcodeText = findViewById(R.id.barcode_text);
Send_button = findViewById(R.id.Save_data);
initialiseDetectorsAndSources();
    // Getting Data From FireBaseFirestore
    FirebaseFirestore DataBase = FirebaseFirestore.getInstance();
    CollectionReference db = DataBase.collection("Product");
    //Getting the Current User
    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    if (user != null) {
        // Getting the current user ID
        String uid = user.getId();
        Send_button.setOnClickListener(view -> {
            // Converting the Scanned BarcodeText to String
            String number = barcodeText.getText().toString();
            // Verifying if the Scanned Barcode Data is Present in
the Product Collection
            db.whereEqualTo("Code", number)
                .get()
                .addOnCompleteListener(task -> {
                    if (task.isSuccessful()) {
                        for (QueryDocumentSnapshot document :
task.getResult()) {
                            Log.d(TAG, document.getId() + " => " +
document.getData());
                            //After Verifying Getting the
document name to string & getting the data
                            String str = document.getId();
                            // Creating a new collection called
"Cart" to the current user
                            Map<String, Object> Barcode = new
HashMap<>();
                            Barcode.put("Pname", str);
                            Barcode.put("Code", number);

                            DataBase.collection("Users").document(uid)

                            .collection("Cart").document(str)

                                .set(Barcode)
                                .addOnSuccessListener(aVoid ->
Log.d(TAG, "DocumentSnapshot successfully written!"))
                                .addOnFailureListener(e ->
Log.w(TAG, "Error writing document", e));
                                startActivity(new
Intent(getApplicationContext(), DisplayingBarcodeData.class));
                                }
                            } else {
                                Log.d(TAG, "Error getting documents: ",
task.getException());
                            }
                        });
                    });
                } // No user is signed in
            }

    // Creating another private class to detect the Barcode using Barcode-
Detector
    private void initialiseDetectorsAndSources() {
        // Setting up a Toast message to Let the User Know the data id
present
        Toast.makeText(getApplicationContext(), "Barcode scanner started",

```

```

Toast.LENGTH_SHORT).show();
    // The Detection Starts
    barcodeDetector = new BarcodeDetector.Builder(this)
        .setBarcodeFormats(Barcode.ALL_FORMATS)
        .build();

    cameraSource = new CameraSource.Builder(this, barcodeDetector)
        .setRequestedPreviewSize(1920, 1080)
        .setAutoFocusEnabled(true) //you should add this feature
        .build();

    // Using try and catch to check id the user has permitted the
    access to camera
    surfaceView.getHolder().addCallback(new SurfaceHolder.Callback() {
        @Override
        public void surfaceCreated(SurfaceHolder holder) {
            try {
                if
(ActivityCompat.checkSelfPermission(BarcodeScanner.this,
Manifest.permission.CAMERA) == PackageManager.PERMISSION_GRANTED) {
                    cameraSource.start(surfaceView.getHolder());
                } else {

ActivityCompat.requestPermissions(BarcodeScanner.this, new
String[]{Manifest.permission.CAMERA},
REQUEST_CAMERA_PERMISSION);
            }

            } catch (IOException e) {
                e.printStackTrace();
            }

        }

        @Override
        public void surfaceChanged(SurfaceHolder holder, int format,
int width, int height) {
        }

        @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
            cameraSource.stop();
        }
    });

    //processing the Barcode
    barcodeDetector.setProcessor(new Detector.Processor<Barcode>() {
        @Override
        public void release() {
            //Toast.makeText(getApplicationContext(), "To prevent
memory leaks barcode scanner has been stopped", Toast.LENGTH_SHORT).show();
        }

        // Receiving the detected barcode
        @Override
        public void receiveDetections(Detector.Detections<Barcode>
detections) {
            final SparseArray<Barcode> barcodes =
detections.getDetectedItems();
            if (barcodes.size() != 0) {

```

```

        barcodeText.post(new Runnable() {

            @Override
            public void run() {

                if (barcodes.valueAt(0).email != null) {
                    barcodeText.removeCallbacks(null);
                    barcodeData =
barcode.valueAt(0).email.address;
                    barcodeText.setText(barcodeData);

toneGen1.startTone(ToneGenerator.TONE_CDMA_PIP, 150);
                } else {

                    barcodeData =
barcode.valueAt(0).displayValue;
                    barcodeText.setText(barcodeData);

toneGen1.startTone(ToneGenerator.TONE_CDMA_PIP, 150);

                }
            }
        });
    }

    });
}

@Override
protected void onPause() {
    super.onPause();
    cameraSource.release();
}

@Override
protected void onResume() {
    super.onResume();
    initialiseDetectorsAndSources();
}
}

```



Displaying the Barcode Data Codes (Cart View)

```
package com.example.tranquilasmartsupertmarketapllication;

import android.content.Intent;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.util.Log;
import android.view.View;
import android.widget.ListView;
import android.widget.TextView;

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

import com.firebase.ui.firestore.FirestoreRecyclerOptions;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.button.MaterialButton;
import com.google.firebase.auth.FirebaseAuth;
```

```

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreException;
import com.google.firebase.firestore.Query;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

public class DisplayingBarcodeData extends AppCompatActivity {
    public static final String TAG = "Added";

    RecyclerView recyclerView;
    cart_adapter cart_adapter;
    MaterialButton ClearAll;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_displaying_barcode_data);
        recyclerView = findViewById(R.id.recView);
        ClearAll = findViewById(R.id.Clear_Cart);

        recyclerView.setLayoutManager(new
LinearLayoutManager(DisplayingBarcodeData.this));

        FirebaseFirestore DataBase = FirebaseFirestore.getInstance();

        FirebaseAuth user = FirebaseAuth.getInstance().getCurrentUser();
        if (user != null) {
            // Getting the current User ID
            String uid = user.getId();
            // getting the data source
            Query query =
DataBase.collection("Users").document(uid).collection("Cart");
            // displaying data into a recycle view
            FirestoreRecyclerOptions<CartModel> options =
                new FirestoreRecyclerOptions.Builder<CartModel>()
                    .setQuery(query, CartModel.class)
                    .build();

            cart_adapter = new cart_adapter(options);
            recyclerView.setAdapter(cart_adapter);
            cart_adapter.startListening();

            // Setting up the clear Cart
            ClearAll.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {

                    DataBase.collection("Users").document(uid).collection("Cart")
                        .delete()
                        .addOnSuccessListener(new
OnSuccessListener<Void>() {
                            @Override
                            public void onSuccess(Void aVoid) {
                                Log.d(TAG, "DocumentSnapshot
successfully deleted!");
                            }
                        })
                })
            }
        }
    }
}

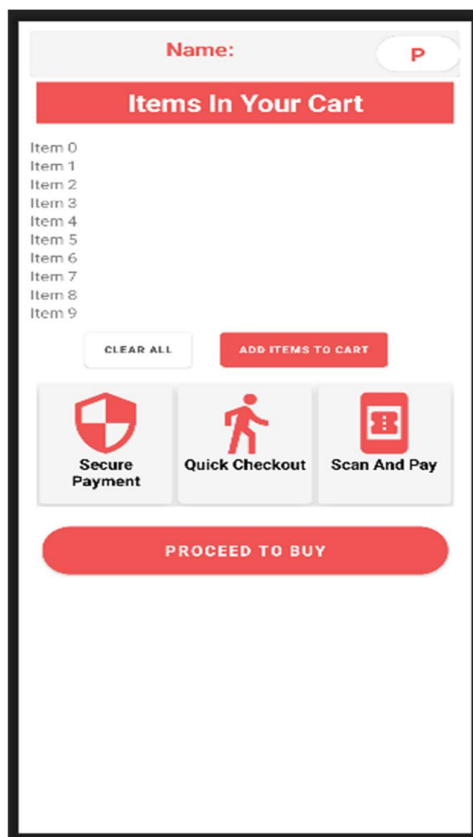
```



```

        .addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e)
            {
                Log.w(TAG, "Error deleting document",
                e);
            }
        });
    }
}
}
}
}

```



Profile Page

```
package com.example.tranquilasmartsupertmarketaplllication;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;

import android.content.Intent;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.util.Log;
import android.view.View;
import android.widget.TextView;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.button.MaterialButton;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.Objects;

public class ProfilePage extends AppCompatActivity {

    public static final String TAG = "Added";

    TextView Fullname;
    TextView Username;

    MaterialButton Update_User;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_profile_page);

        Fullname = (TextView) findViewById(R.id.fullname);
        Username = (TextView) findViewById(R.id.Username);
        Update_User = (MaterialButton)
findViewById(R.id.Update_User_Profile);

        // Setting up Firebase
        FirebaseFirestore Data = FirebaseFirestore.getInstance();
        // setting up to verify the current user
        FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
        // Verifying if the user id Signed in or not
        if (user != null){
            //getting the current Users ID
            String uid = user.getId();
            // setting up the path to get the current user
            DocumentReference Docdata =
Data.collection("Users").document(uid);
            // Getting the uer Data
            Docdata.get().addOnCompleteListener(new
OnCompleteListener<DocumentSnapshot>() {
                @Override
                public void onComplete(@NonNull Task<DocumentSnapshot>
```

```

task) {
    if (task.isSuccessful()) {
        DocumentSnapshot document = task.getResult();
        if (document.exists()) {
            Log.d(TAG, "DocumentSnapshot data: " +
document.getData());

            String fname = document.getString("Fullname");
            Fullname.setText(fname);
            String Uname = document.getString("Username");
            Username.setText(Uname);

        } else {
            Log.d(TAG, "No such document");
        }
    } else {
        Log.d(TAG, "get failed with ",
task.getException());
    }
}

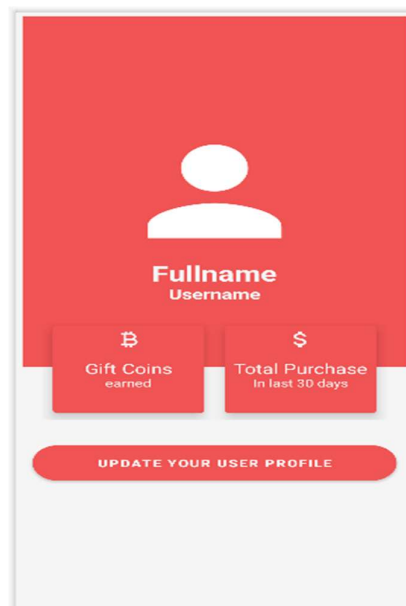
});

// Setting up the Update method
Update_User.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new
Intent(ProfilePage.this, MenuPage.class);
        startActivity(intent);
        finish();
    }
});

}

}
}

```



5.3 Testing Approach

5.3.1 Unit Testing

Unit testing, a testing technique using which individual modules are tested to determine if there are any issues by the developer himself. It is concerned with functional correctness of the standalone modules.

The main aim is to isolate each unit of the system to identify, analyse and fix the defects.

Unit Testing - Advantages:

- Reduces Defects in the Newly developed features or reduces bugs when changing the existing functionality.
- Reduces Cost of Testing as defects are captured in very early phase.
- Improves design and allows better refactoring of code.
- Unit Tests, when integrated with build gives the quality of the build as well.

I tested each module of my application. The validation of login and signup form is done individually which is working fine.

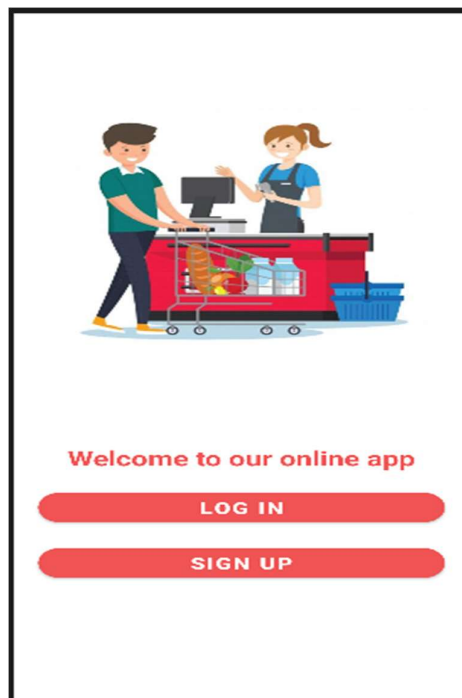
Let's have look at my project unit Testing.

The Inro Page:



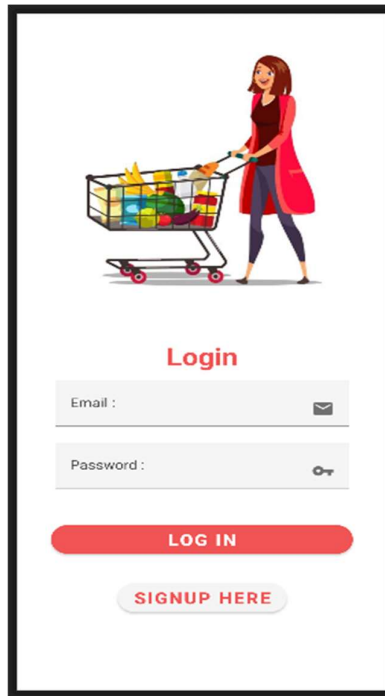
On this page I have displayed a custom logo which I have made in Canva and the I have set a Timer Method for 5 sec, in which this Page will appear only for 5 sec. This page is working as expected. Thus, this module was a success.

The Main Menu Page:



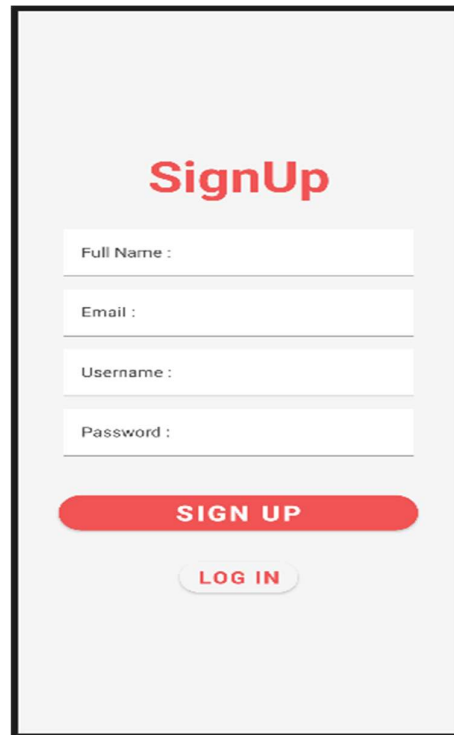
On this page will have added an image and I have set two options for the user to login if he/she is an existing User and Signup is the user is not an existing user. On click on the desired option the user is directed to the login and signup page. This page is working as expected. Thus, this module was a success.

The Login Page:



Once the User clicks on login option the Login page appears. On this page I have added two input fields, in which the user can fill in the login details. On adding the correct ID and Password a toast message is shown as “Successfully Logged In” after which the user is directed to Cart view page. The Data get Authenticated in Firebase. There is a signup button below the login button. If the user is a new user the user can click to register. This page is working as expected. Thus, this module was a success.

Signup Page:



SignUp

Full Name :

Email :

Username :

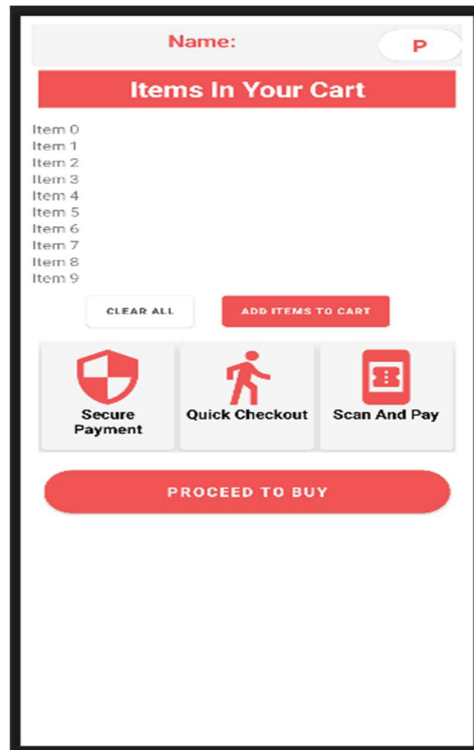
Password :

SIGN UP

LOG IN

On this Page is have added four input field. In which the suer can add their personal details. This page is working as expected. Thus, this module was a success.

Cart View Page:



On this page I have added many functions. The upper left corner I have added a current user full name Text View. The Text View is connected to the firebase Database and it display the current user. The right-hand side have a material button which onclick directs the user to its profile page. The list of scanned items is shown in recycle view. There is “CLEAR ALL” button right below the recycle view, which on click clears the entire scanned Items. On the right -hand side there is “ADD ITEMS TO CART” button which on click directed the user to the Barcode scanner page. This page is working as expected. Thus, this module was a success.

5.3.2 Integrated Testing

- Integration Testing is a level of software testing where individual units are combined and tested in a group.
- The purpose of this level of testing is to expose the faults in the interaction between integrated units.
- **Big Bang** is an approach to integration testing where all the or most of the units are combined and tested at one go.
- **Top Down** is an approach where top-Level units are tested first, and lower-level units are tested step by step after that.
- **Bottom Up** is an approach where the bottom level units are tested first, and the upper-level units are tested step after that.
- **Sandwich/Hybrid** is an approach where it is a combination of Top down and bottom-up approaches.

Chapter 6

Results and Discussion

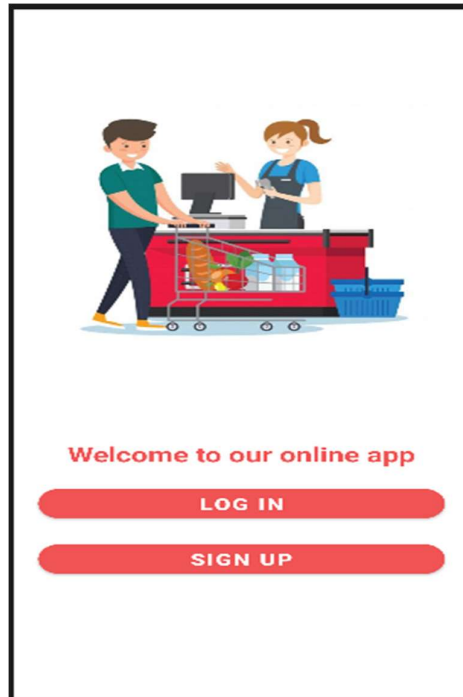
6.1 User Documentation:

- **Main Intro Page**



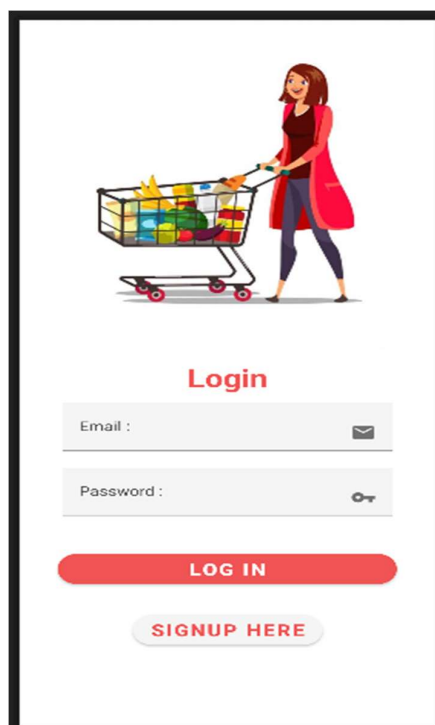
This is how the Main Intro Page looks, this page appears for 5 sec on start.

- **Main Menu Page**

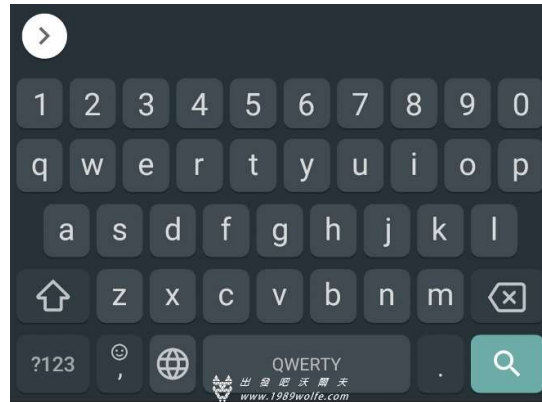


This Page consists of two buttons and a shopping Image, if the user clicks on the login button the user is directed to the login page and if the user clicks on the Signup button the user is directed to the signup page.

- **Login Page**



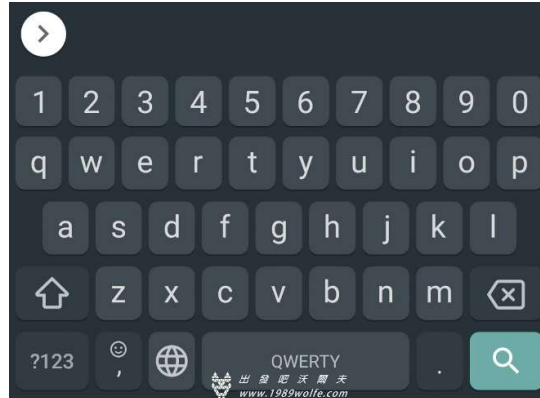
The login page consists of two User-Input fields where the user can fill in their personal details. Below the Use-input fields there are two Buttons Login and Sign in. If the user has entered the correct details and the user clicks on login the user is directed to the Cart View Page. A Toast message will appear on stating that the User has “Successfully Logged In”. If the user clicks on the sign-up button the user is directed to the sign-up page. The user can fill in their personal details using their android keyboard.



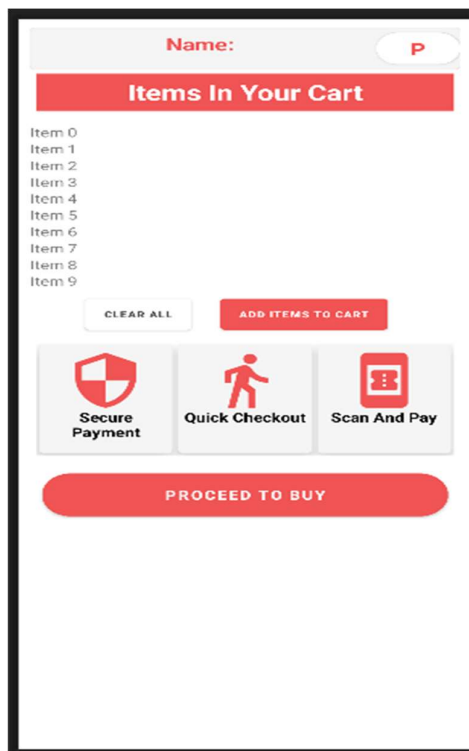
- **Signup Page:**

This Page consists of four User-input fields. The user can fill in their details using their android Keyboard. If the user has added their personal details and the user clicks

on the sign-up button, a toast message appears “You Have Successfully Signed In”. the page also consists of a login button. If the user is already an existing user. On click the user will be directed to the login page. The user can fill in their personal details using their android keyboard.



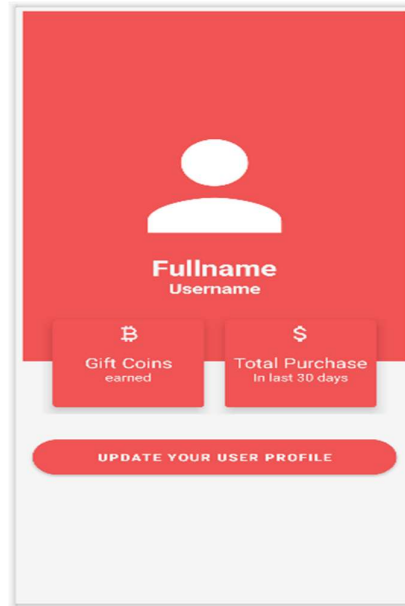
- **Cart View Page:**



This page consists of a recycle view in which the User added Items will appear. On the Top Left corner, the current user’s name will appear. On the Top left corner there is profile button which on click will direct the user to the Profile Page. below the recycle view there are two buttons. The “CLEAR ALL” button will clear all the added items in the cart. Besides the “CLEAR ALL” button there is another button “ADD

ITEMS TO CART” on click of this button the Barcode Scanner page will appear. After scanning the product details will be displayed on the Cart View page in the recycle view. Below the card view there is button “PROCEED TO BUY”, this button will direct you to the payment page.

Profile Page:



The profile page consists of text view in which the user Full Name will be Displayed and below the Full name there is another text view in which the User name will be displayed. There are two card view options the left-hand side car view consists of a gift coin which on purchase gifts you a gold coin. The right-hand side card view consists of a total purchase view which the user has purchased previously. On the bottom there is update user button which on click directs the user to the page where the user can update their personal details.

Chapter 7

Conclusion and Future Plans

Conclusion:

As of now there is no such software Application as Tranquil. Successful use of a self-checkout system can differentiate a retailer from competitors, optimise the checkout process, and attract more customers. Self-checkout solutions have been key to retail success for years, especially for grocery stores and supermarkets.

When it comes to cost of space, checkout counters definitely take up less space than registers. When it comes to cost of space, mobile scanners are at an advantage compared to cash counters. They undoubtedly take much less space.

The main advantage of this solution is that it provides a real-time overview of how much all the items picked up cost. This is something that contributes to the practicality and enjoyment of this solution.

The speed of transaction, perceived control and reliability are also quite high when it comes to this system.

A recent study showed that 60% of consumers globally would prefer having a “just walk out” self-service experience similar to Amazon Go.

A more cost-efficient solution that requires an incomparably smaller investment in technology than self-checkout sensors is the mobile scan and go app.

This solution is very simple to use: the customer downloads the app and fills out their personal data and payment information.

As the customer picks and scans goods in the store, prices are displayed and calculated. The customer pays in the app with a simple swipe and exits in the kiosk screen area.

Efficiency

When it comes to efficiency, here are some of considerations about mobile self-checkout:

- With mobile scanning, wait times are non-existent. The customer scans a code on their phone as they exit the store, and queues are completely eliminated.
- This system also reduces labour costs. Equipment costs are also reduced, as very little hardware is required and customers can use their own devices.
- With self-checkout mobile apps, customers scan product barcodes on the go and they get a clear real-time overview of their spending.
- Shoplifting is still a concern with this system, however intelligent scanning behaviour analysis can detect possible theft.
- Compared to other systems, this is one of the most cost-effective ones, which makes up for potential losses more efficiently.

One of the advantages this solution has compared to most self-checkout solutions is that the customers scan their goods on the go, and the cost of the items is calculated in real time.

Finalising the purchase can be done anywhere in the store. This is something that contributes to the reliability, practicality, and enjoyment of using this solution

Future Plans:

This Application came under consideration due to the ongoing Covid-19 pandemic situation.

Some of the plans for the future for this application could possibly be:

- To make an IOS version of this app.
- To make the User-Interface more attractive and more user-friendly.
- To make use of modern UI designs and implement it.