VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018

An Internship Project Report
on

The Flower Shop

Submitted in partial fulfillment of the requirements for the VIII Semester of degree of **Bachelor of Engineering in Information Science and Engineering** of Visvesvaraya Technological University, Belagavi

by

Pallavi N
1RN18IS072

Under the Guidance of

Ms Sowmya S K
Assistant Professor
Department of ISE

Department of Information Science and Engineering

RNS Institute of Technology

Dr. Vishnuvaradhan Road, Rajarajeshwari Nagar post, Channasandra, Bengaluru-560098

2021-2022

RNS INSTITUTE OF TECHNOLOGY

Dr. Vishnuvaradhan Road, Rajarajeshwari Nagar post,
Channasandra, Bengaluru - 560098

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the Internship work entitled ***The Flower Shop*** has been successfully completed by **Pallavi N (1RN18IS072)** a bonafide student of **RNS Institute of Technology, Bengaluru** in partial fulfillment of the requirements of 8th semester for the award of degree in **Bachelor of Engineering in Information Science and Engineering** of **Visvesvaraya Technological University, Belagavi** during academic year **2021-2022**. The internship report has been approved as it satisfies the academic requirements in respect of internship work for the said degree.

Ms Sowmya S K	**Dr. Suresh L**	**Dr. M K Venkatesha**
Internship Guide	Professor and HoD	Principal
Assistant Professor	Department of ISE	RNSIT
Department of ISE	RNSIT	

External Viva

Name of the Examiners	**Signature with Date**
1. ______________________	**1.** ______________________
2. ______________________	**2.** ______________________

DECLARATION

I, **PALLAVI N [USN: 1RN18IS072]** student of VIII Semester BE, in Information Science and Engineering, RNS Institute of Technology hereby declare that the Internship work entitled ***The Flower Shop*** has been carried out by us and submitted in partial fulfillment of the requirements for the *VIII Semester degree of* ***Bachelor of Engineering in Information Science and Engineering*** *of Visvesvaraya Technological University*, *Belagavi* during academic year 2021-2022.

Place : Bengaluru

Date :

PALLAVI N

(1RN18IS072)

ABSTRACT

Flutter is an open-source cross-platform mobile application development SDK created by Google. It is highly user-friendly and builds high-quality mobile applications. It is used to develop cross platform applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia,Web platform.

Dart is a programming language designed for client development, such as for the web and mobile apps. It is developed by Google and can also be used to build server and desktop applications. Dart is an object-oriented, class-based, garbage-collected language with C -style syntax. Dart can compile to either native code or JavaScript.

FloraGoGo is a flower shop management system project. It is a cross platform application which is developed using Flutter . This application focuses only on the basic front-end of the application and the flutter application is developed on Visual Studio by using Dart as the programming language .

FloraGoGo application has a total of three screens on focus: the loading page, login and the menu screen which displays all the available products.The entire application is set to be made out of widgets , which describes the view of the application would look like in a given state. Depending on the interaction with the user the screens are designed to be either stateless widget or stateful widget . Navigation to the following screen is performed on Button Click using the built-in method.

ACKNOWLEDGMENT

At the very onset I would like to place our gratefulness to all those people who helped me in making the Internship a successful one.

Coming up, this internship to be a success was not easy. Apart from the sheer effort, the enlightenment of the very experienced teachers also plays a paramount role because it is they who guided me in the right direction.

First of all, I would like to thank the **Management of RNS Institute of Technology** for providing such a healthy environment for the successful completion of internship work.

In this regard, I express sincere gratitude to our beloved Principal **Dr. M K Venkatesha,** for providing us all the facilities.

We are extremely grateful to our own and beloved Professor and Head of Department of Information science and Engineering, **Dr. Suresh L**, for having accepted to patronize me in the right direction with all her wisdom.

We place our heartfelt thanks to **Ms Sowmya S K** Assistant Professor , Department of Information Science and Engineering for having guided internship and all the staff members of the department of Information Science and Engineering for helping at all times.

I thank **Mr. Akshay D R** ENMAZ , for providing the opportunity to be a part of the Internship program and having guided me to complete the same successfully.

I also thank our internship coordinator **Dr. R Rajkumar,** Associate Professor, Department of Information Science and Engineering. I would thank my friends for having supported me with all their strength and might. Last but not the least, I thank my parents for supporting and encouraging me throughout. I have made an honest effort in this assignment.

PALLAVI N

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

UI - User Interface

HTML - Hyper Text MarkUp Language

SDK - Software Development Kit

OS - Operating system

XML - Extensible MarkUp Language

DB - Database

IDE - Integrated Development Environment

SRS - Software Requirement Specification

API - Application Programming Interface

MVC - Model View Controller

Chapter 1

INTRODUCTION

1.1 Introduction to Flutter

Flutter is an open source framework to create high quality, high performance mobile applications across mobile operating systems - Android and iOS. It provides a simple, powerful, efficient and easy to understand SDK to write mobile applications in Google's own language, *Dart*.

In general, developing a mobile application is a complex and challenging task. There are many frameworks available to develop a mobile application. Android provides a native framework based on Java language and iOS provides a native framework based on Objective-C / Swift language

However,to develop an application supporting both the OSs, we need to code in two different languages using two different frameworks. To help overcome this complexity, there exists mobile frameworks supporting both OS. One such framework which overcomes this challenge is Flutter which is a cross platform application

Flutter also offers many ready to use widgets (UI) to create a modern application. These widgets are optimized for mobile environments and designing the application using widgets is as simple as designing HTML.Flutter widgets also supports animations and gestures.

1.2 Features of Flutter

Flutter framework offers the following features to developers –

- Modern and reactive framework.
- Uses Dart programming language and it is very easy to learn.
- Fast development.
- Beautiful and fluid user interfaces.
- Huge widget catalog.
- Runs same UI for multiple platforms

1.3 Advantages of Flutter

- Flutter comes with beautiful and customizable widgets for higher performance and outstanding mobile applications by fulfilling the custom needs and requirements
- Dart has a large repository of software packages which lets you to extend the capabilities of your application.
- Developers need to write just a single code base for both applications and web development as well (both Android and iOS platforms).
- Simplicity of Flutter, makes it a good candidate for fast development
- Developers of Flutter have full control over the widgets and its layout
- Flutter offers great developer tools, with amazing hot reloads
- Flutter needs lesser testing. Because of its single code base,it is sufficient if we write automated tests once for all the cross platforms

1.4 Architecture Applications

Widgets,the core concept of the Flutter Framework .Widgets are basically user interface components used to create the user interface of the application.The Flutter application is itself a widget. The application is the top- level widget and its UI is build using one or more children widgets , which again build using its children widgets. This composability feature helps us to create a user interface of any complexity.

Flutter widgets support interaction through a special widget, GestureDetector. GestureDetector is an invisible widget having the ability to capture user interactions such as tapping, dragging, etc., of its child widget.

Flutter widgets support State maintenance by providing a special widget, StatefulWidget. Widget needs to be derived from StatefulWidget widget to support state maintenance and all other widget should be derived from StatefulWidget.

The most important concept of the Flutter framework is that the framework is grouped into multiple categories in terms of complexity and clearly arranged in layers of decreasing complexity. A layer is built using its immediate next level layer. The top most layer is widget specific to Android and iOS. The next layer has all flutter native widgets. The next layer is the Rendering layer, which is low level renderer component and renders everything in the flutter app. Layers goes down to core platform specific code

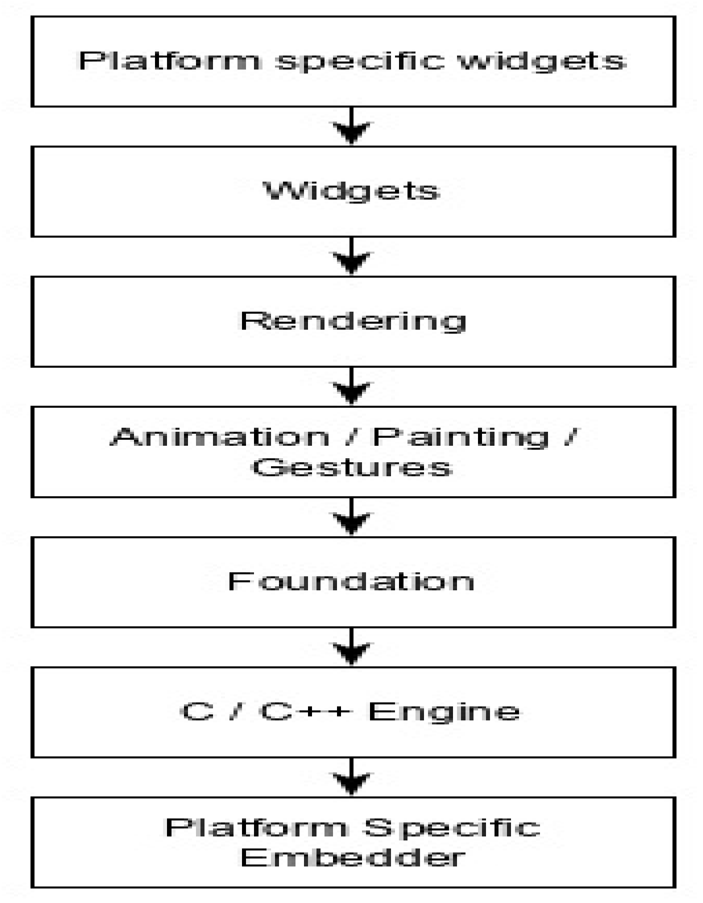

Platform specific widgets
Widgets
Rendering
Animation / Painting / Gestures
Foundation
C / C++ Engine
Platform Specific Embedder

Fig 1.1 : General Overview of Layer in Flutter

1.5 Flutter Installation

Step 1: Download the installation bundle of the Flutter Software Development Kit for macOS. To download Flutter SDK.

Step 2: When your download is complete, extract the zip file and place it in the desired installation folder or location.

Step 3: To run the Flutter command, you need to update the system path to include the flutter bin directory.

Step 4: Next, enable the updated path in the current terminal window using the below command and then verify it also.

Step 5: Now, run the $ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation.

Step 6: When you run the above command, it will analyze the system and the details of all missing tools, which are required to run Flutter as well as the development tools that are available but not connected with the device.

Step 7: Next, you need to set up an iOS simulator or connect an iPhone device to the system for developing an iOS application.

Step 8: Again, set up an android emulator or connect an android device to the system for developing an android application.

Step 9: Now, install the Flutter and Dart plugin for building Flutter applications in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself.

Chapter 2

Literature Review

2.1 Android Studio

Android Studio is an integrated development environment (IDE) for Google Android Operating System. It is built based on JetBrains' IntelliJ IDEA Community Edition, and it is specifically designed for creating applications on Android devices. Some of the key features of Android Studio are as follows:

1. Instant Run – a feature that pushes code and resource changes to the running app. It allows changes to be made to the app without the need to restart the app, or rebuilding the APK, so that the effects can be seen instantly.
2. An Emulator – a virtual android device that can simulate a variety of hardware features such as GPS location, network latency, motion sensors, and multi-touch input that can be used to run and install the app. It can then be used for testing purposes.
3. Testing Tools and Frameworks – extensive testing tools such as, JUnit 4 and functional UI test frameworks are included with Android Studio. Espresso Test Recorder can generate UI test code by recording the developer's interactions with the app on a device or emulator. The tests can be run on a device, an emulator, in Firebase Test Lab, or on a continuous integration environment.

2.2 Java Programming Language

Java is an object-oriented programming language created by James Gosling, Mike Sheridan, and Patrick Naughton in 1991. In the paper The Java Language Specification Java SE 8 Edition James Gosling states, "Java programming language is a general-purpose, concurrent, class based, object-oriented language. It is designed to be simple enough that many programmers can achieve fluency in the language. The Java programming language is related to C and C++ but is organized rather differently, with a number of aspects of C and C++ omitted and a few ideas from other languages included. It is intended to be a production language, not a research language." Java is a very flexible programming language which is used to create many different types of applications for many different operating systems. This is possible because Java can be run on any operating system, as long as the Java Runtime Environment is available. The

application created for Android devices must be coded using Java programming language. This allows these apps to work on variety of different devices, no matter the company that has manufactured the device.

2.3 XML

XML or Extensible Markup Language is a text language that can be used to describe the behavior of programming languages that process them. XML was developed XML working group in 1996. According to World Wide Web Consortium there are ten design goals for XML. These design goals are:

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.
4. It shall be easy to write programs which process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.
8. The design of XML shall be formal and concise.
9. XML documents shall be easy to create.
10. Terseness in XML markup is of minimal importance.

XML is used when transferring data from the database to the client, and in designing the visual aspect of Android applications. When data is sent from the database, it is sent using XML. This allows the data to be processed by any programming language the same way, since the data is always sent using XML. As mentioned, XML is also used to design the user interface of

Android applications. This means that all the visual aspects such as, the layout of the page, the position of all button and text fields, as well as the color of anything on the page is specified using XML. Since XML is human-legible, it makes the process of designing a page in the app relatively easy and intuitive.

Chapter 3

Analysis

3.1 Introduction

The project deals with the basic User Interface for online ordering of flowers using a mobile application.

The user must login in order to proceed further with the application. The splash screen is followed by the login screen where the user can login using the existing account details like email ID and password or if they are new can sign up into the application by clicking on sign up and creating a new account. On login, they are directed to the menu screen with all details of flowers along with the price and the quantity they prefer. This is the last screen which consists of the details of flowers and the quantity the user prefers.

3.2 Software requirement specification

3.2.1 Introduction

A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform.

3.2.1.1 Flutter

Flutter is an open source framework to create high quality, high performance mobile applications across mobile operating systems - Android and iOS. It provides a simple, powerful, efficient and easy to understand SDK to write mobile applications in Google's own language, *Dart*. Flutter also offers many ready to use widgets (UI) to create a modern application. These widgets are optimized for mobile environments and designing the application using widgets is as simple as designing HTML.Flutter widgets also supports animations and gestures.

3.2.1.2 Visual Studio

Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js, Python and C++.It is based on the Electron framework which is used to develop Node.js Web applications that run on the Blink layout engine. It allows users to open one or more directories, which can then be saved in workspaces for future reuse.Visual Studio Code can be extended via extensions, available through a central repository. A notable feature is the ability to create extensions that add support for new languages, themes, and debuggers, perform static code analysis, and add code linters using the Language Server Protocol. Visual Studio Code includes multiple extensions for FTP, allowing the software to be used as a free alternative for web development. Code can be synced between the editor and the server, without downloading any extra software.

3.2.1.3 SQFlite

SQFlite is a plugin for flutter. It allows us store, retrieve and manipulate our SQLite databases via flutter code. SQFlite supports both Android and iOS platforms.

Features of SQFlite:

- SQFlite provides for both database transactions as well as batches.
- SQlite has inbuilt automatic version managment.
- SQFlite provides easy to use methods for inserting, querying, updating as well as deleting data from database.
- CRUD operations are performed in the background thread on both iOS and Android. This frees the UI to remain responsive.

3.2.1.4 Dart

Dart is a client-optimized language for fast applications on any platform.Dart is a programming language designed for client development, such as for the web and mobile apps. It is developed by Google and can also be used to build server and desktop applications. Dart is an object-oriented, class-based, garbage-collected language with C -style syntax. Dart can compile to either native code or JavaScript.

3.2.2 Requirements

3.2.2.1 Functional Requirements

1. Splash Screen : The loading screen needs to be navigated to the login screen with a time gap of few seconds
2. Sign Up : Account Creation if the user is new to the application
3. Login : If the user is already having the account
4. Email id and password validations upon insertion
5. Increasing and decreasing the quantity of the product displayed the menu screen
6. Display of the images , cost of the bouquet which the users want to purchase

3.2.2.2 Non-Functional Requirements

1. The concept of splash screen or Launch Screen is implemented using the built in method Timer().
2. Floating Action Button and its attribute onSelect() helps to implement the features on increment and decrement of the quantity.
3. Models are designed to display images,cost and to keep count of the number of products the user wants to purchase
4. Navigation has to implemented to traverse across the screens . The class Navigator has a lot of methods where each perform different actions . The route widget has all the pages or screens in the Flutter application

3.2.2.3 Software and Hardware Requirements

Operating System : Windows10

Programming Language : Dart

Processor : Intel Core i5 7th Gen

Speed : 2.33 Ghz

RAM : 8GB

Software Development Kit : Flutter

IDE : Visual Studio or Android Studio(Official)

Tools : Windows Powershell , Git

3.2.3 General Description

3.2.3.1 Assumptions and Dependencies

Flutter supports using shared packages contributed by other developers to the Flutter and Dart ecosystems. This allows quickly building an app without having to develop everything from scratch.

A Flutter app can depend on a plugin via a file system path: dependency. The path can be either relative or absolute. Relative paths are evaluated relative to the directory containing pubspec.yaml.

Scoped Model:

A set of utilities that allow you to easily pass a data Model from a parent Widget down to its descendants. In addition, it also rebuilds all of the children that use the model when the model is updated. This library was originally extracted from the Fuchsia codebase.

SQflite:

SQflite is the SQLite plugin for Flutter. Supports iOS, Android and MacOS. Support transactions and batches. Automatic version management during open. It helps in insert/query/update/delete queries. DB operation executed in a background thread on iOS and Android

Path_provider:

A Flutter plugin for finding commonly used locations on the filesystem. Supports Android, iOS, Linux, macOS and Windows. Not all methods are supported on all platforms.

Chapter 4

System Design

4.1 Introduction

The major aim of the application is to order flowers online. The focus of the application is on the UI design rather than on the Working of the application as an online flower ordering. It has a login screen which enables you to go to the next screen. The sign in screen allows you to create or register with the application. The home screen is the main focus which consist of all the details about the flowers available.

The system design can be represented as a widget tree. Flutter widgets are built using a modern framework that takes inspiration from React. The central idea is that you can build one UI out of widgets. Widgets describe what their view should look like given their current configuration and state. When a widget's state changes, the widget rebuilds its description, which the framework differs against the previous description in order to determine the minimal changes needed in the underlying render tree to transition from one state to the next.
There are different kinds of widget as follows:

1.**Container Widget:** Container Widget in Flutter is a parent widget that can contain multiple child widgets and manage them efficiently through various attributes.It is a widget that combines common painting, positioning, and sizing of the child widgets. It is also a class to store one or more widgets and position them on the screen according to our needs.
2.**Column Widget :** Column Widget in Flutter does not scroll . This widgets places the children widget one below the other .
3.**Row Widget :** This widget arranges its children in a horizontal direction on the screen.
4.**Text Widget :** Text Widget allows us to display a string of text with a single line in our application.5.**Icon widget** : This is used to create icons in our applications. We can create icons in Flutter, either using inbuilt icons or with the custom icons. Flutter provides the list of all icons in the Icons class.

6. **Scafflold Widget :**The Scaffold is a widget in Flutter used to implement the basic material design visual layout structure. This widget is able to occupy the whole device screen. It provides many widgets or APIs for showing Drawer, Snack Bar, Bottom Navigation Bar, App Bar, Floating Action Button.

7.**Box Decoration** : This widget in flutter is an in-built API. It describes how a box should be painted on the screen and its shape. It comes with a ton of properties to further enhance it.

8.**Align Widget:** Align Widget is the widget that is used to align its child within itself and optionally sizes itself based on the child's size. Align Widget is quite flexible and can change its size according to the size of its child.

9. **Stateless widget :** Stateless widget is a widget that describes part of the user interface by building a constellation of other widgets that describe the user interface more concretely. The widgets whose state can not be altered once they are built are called stateless widgets. These widgets are immutable once they are built . It is used when the UI depends on the information within the object itself.

10.**Stateful widget:** Stateful Widgets are the ones that change its properties during run-time. They are dynamic i.e., they are mutable and can be drawn multiple times within its lifetime. It can change its appearance in response to events triggered by user interactions or when it receives data.

4.2 System Architecture

4.2.1 Overview of Application

The application starts with the splash screen. It contains the company logo which is displayed for about 2 seconds.

This is followed by the login screen. It contains the basics details for login like email ID and password. If a user does not have an account can create an account by going to the sign up page from the login page. The user will have to enter the details in order to create an account and these details will be uploaded or saved in the database.

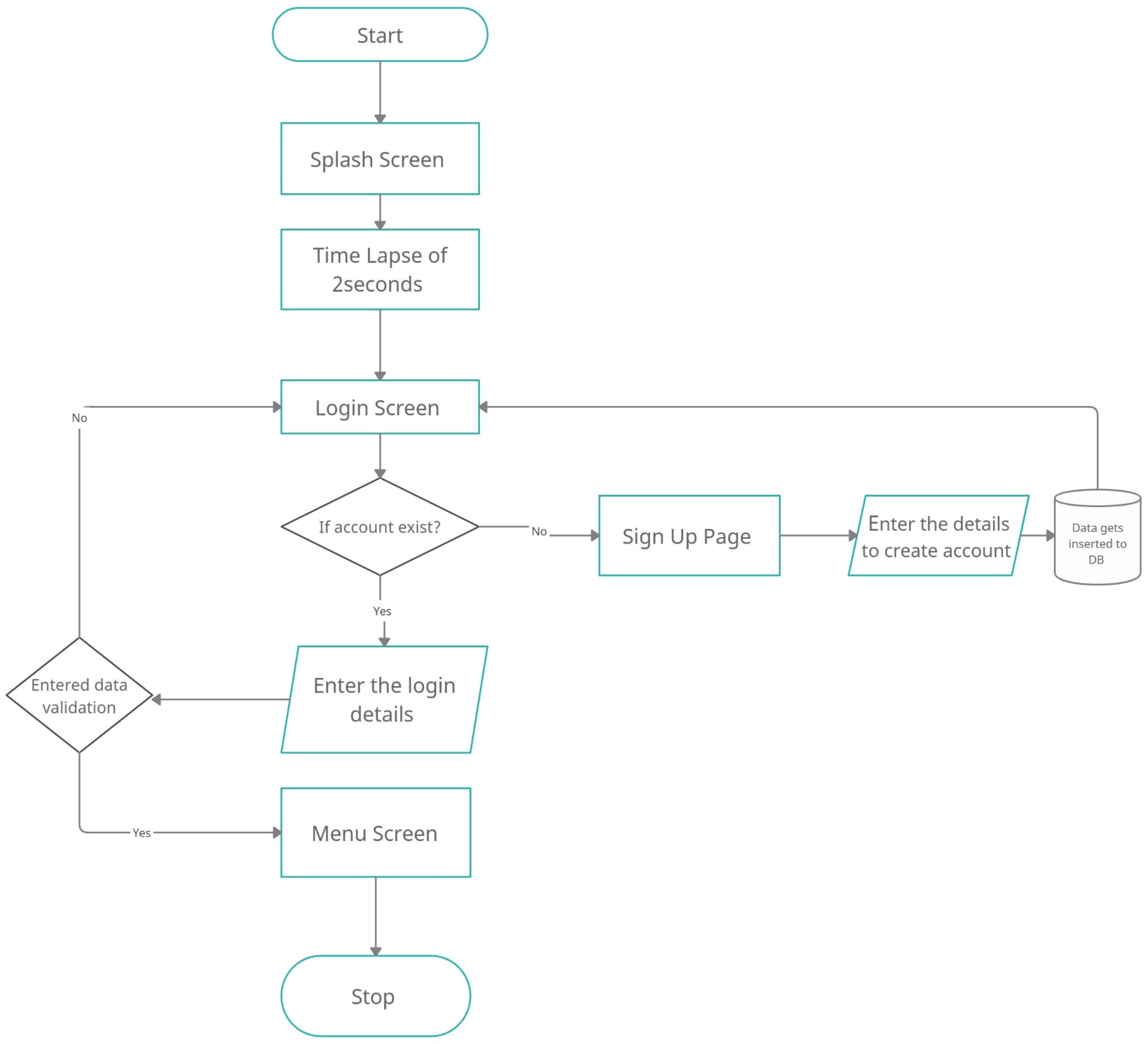


Fig 4.1 : General Overview of the Application

Once a person registers or creates an account, it directs back to the login page. In the login page the user enters the details like the registered email id and password. It will check if the entered emailId is registered in our database and if it is then checks if the password entered is correct. If the entered password matches, then goes the next page that is the menu page. Else it will remain in the login page itself.

Chapter 5

Detailed Design

5.1 High Level Design

5.1.1Design Considerations

5.1.1.1 Widget Tree for Splash Screen

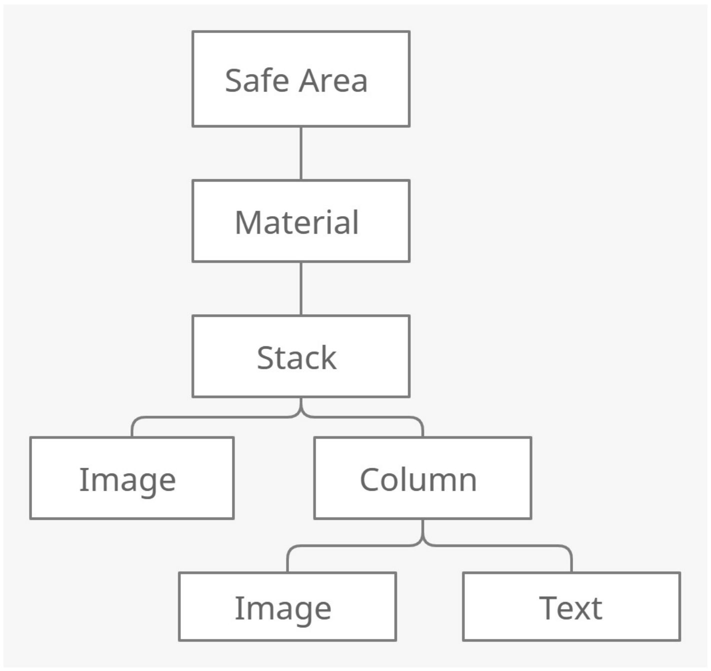


Fig 5.1 : Widget tree of splash screen

The screen is an introduction to the application. It contains the company name, company logo as well as the major design of the company.

It has a material widget inside which we use a stack widget because we want to have a background image and over that image we want to have the details of the company. So over the image we have a column widget which consists of the company logo and below which we have the company name.

5.1.1.2 Widget Tree for Login Screen

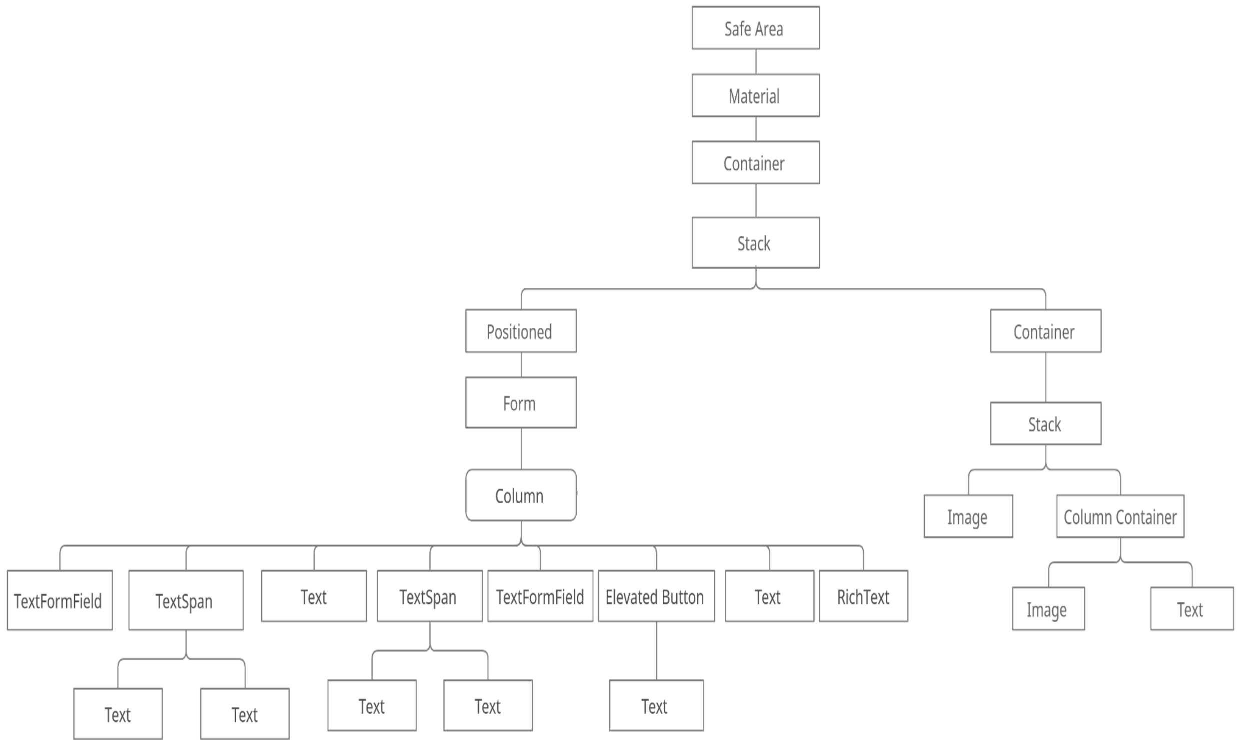


Fig 5.2 : Widget tree of Login screen

The figure 5.2 represents the widget tree of the login screen. It has a Stack in which we have two containers. First for the company logo and details and second for the login form.

The first part consists of another stack because we want to have a background image and over which we want the logo and the company name which is inside a column container so that they come one below the other.

The second part is a form which firstly has a text to show the introduction to the login screen. As we have two texts in different colors, we use two different text children of textspan. This is followed by two text widgets and textFormField widgets, where text widgets are for the heading of the formFields for the user to enter the details. It is followed by the forget password text and then the elevated button for login and lastly another textspan for signUp.

5.1.1.3 Widget Tree for Sign Up Screen

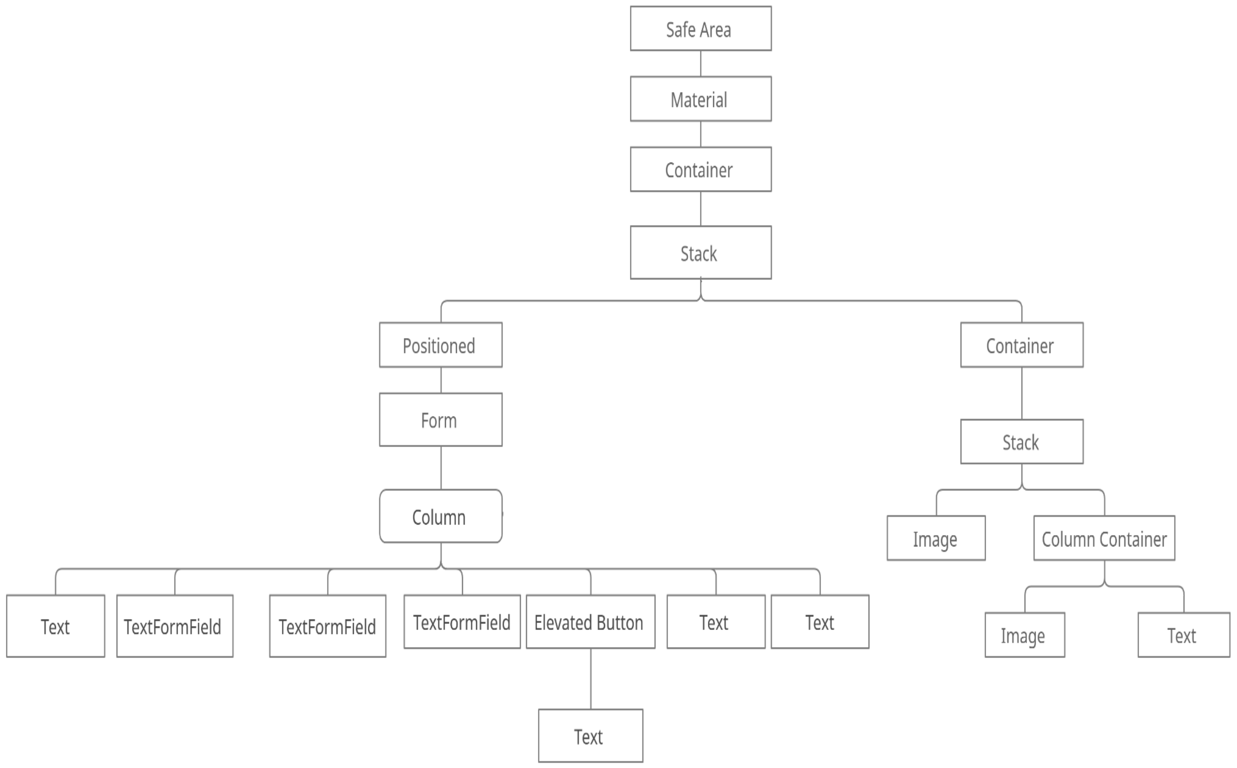


Fig 5.3 : Widget tree of Sign Up screen

The figure 5.3 represents the widget tree of the sign up screen. It has a Stack in which we have two containers. First for the company logo and details and second for the login form.

The first part consists of another stack because we want to have a background image and over which we want the logo and the company name which is inside a column container so that they come one below the other.

The second part is a form which firstly has a text to show the introduction to the sign up screen. This is followed by three text widgets and textFormField widgets, where text widgets are for the heading of the formFields for the user to enter the details. Lastly, an elevated button for sign up.

5.1.1.4 Widget Tree for Menu Screen

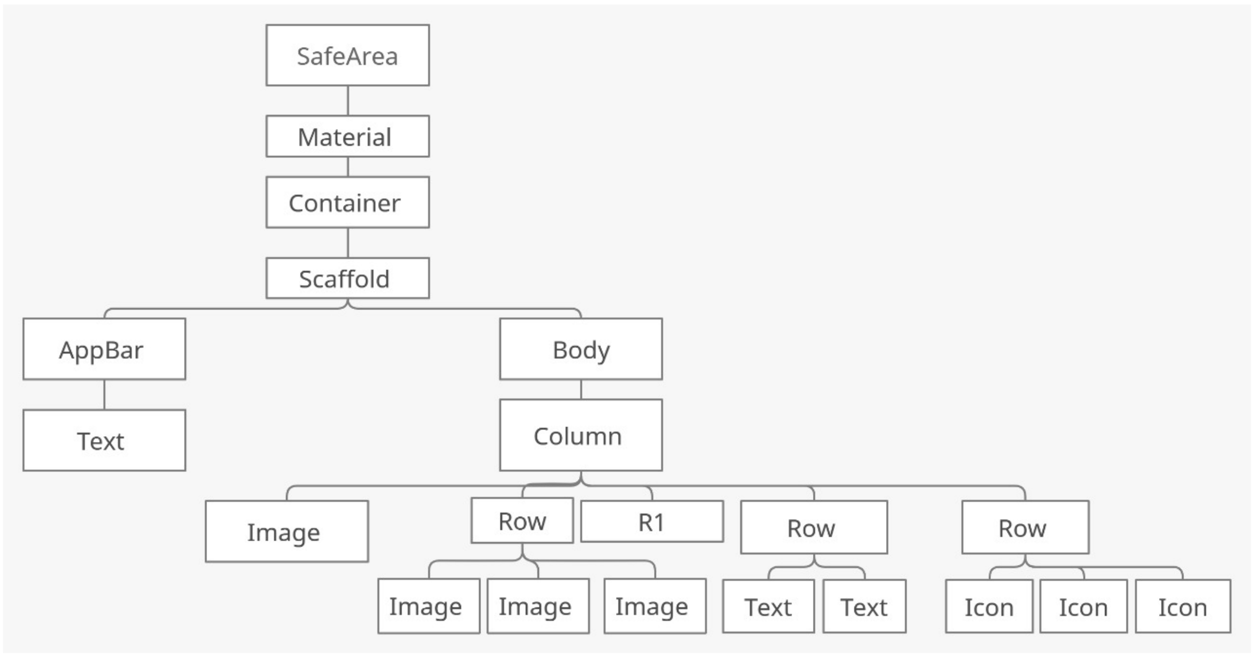


Fig 5.4.1 : Widget tree of Menu Screen

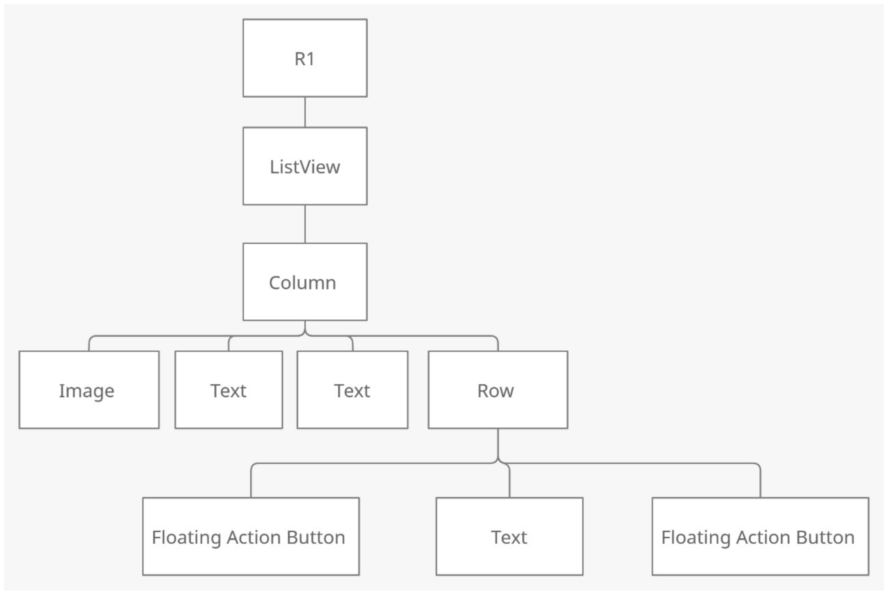


Fig 5.4.2 : Continuation of widget tree for home screen for row R1

Figure 5.4.1 and 5.4.2 together are for the home screen. It consists of a scaffold inside which we have the appBar and the body of the scaffold.

The appBar has the company name, back and search button. The body has a Column widget to get all the things one below the other. Firstly a image which is a banner, followed by the flowers available in a row. We get things for the row from the listview. Each of the rows contains another column to have the flower image, the price, name and the quantity. The quantity can be increased and decreased using the floating action buttons. This is followed by another row which consist of the basic icons of an application like the user, settings and home.

5.2 Low Level Design

5.2.1 Use Case Diagram for Login

User	**System**
1. The user has to choose to sign up if they don't have an account. If they have an account they can enter the email Id and password.	
	2. Once the user types the details it appears on the input box.
3. The user have to hit the login button to proceed to the next page.	
	4. The system is going to check if the entered email Id exists in the database and if it exists it checks if the password entered matches with password entered during signup. If it matches it is going to display the next page ie., the menu page.

Table 5.1 Use Case Diagram for Login

5.2.2 Use Case Diagram for Sign Up

User	**System**

1. The user has to enter all the necessary details like name , emailId, password .	
	2. The system enters all the details into the Database and navigates to the login screen

Table 5.2 Use Case Table for Sign Up

5.2.3 Use Case Diagram for Changing the quantity

User	System
1. The user needs to select on the ‘+’ or ‘-” button to increase or decrease the quantity respectively.	
	2. Based on the button pressed the quantity changes accordingly either increasing the count by 1 or decreasing count by 1 . The value of count gets updated automatically. The ‘-’ button has no response if count is 0 .

Table 5.3 Use Case Table for Changing the quantity

Chapter 6

Implementation

6.1 Introduction

Model in flutter refers to data flow and it corresponds to the MVC or model view controller architecture.Managing data becomes easier if the data flows comes from model else its become tedious . It is not mandatory to use models but by using models our job becomes easier .

In our application we have used the following models to manage our application.

1. **Image Model :**

```
class  Images{
            String imageName;
            Images (
            { this.imageName = ‘ ’,});
}
```

2. **Popular Model :**

```
class Popular {
            String imageN, flowerName;
             double cost;
              int count = 0;
              Popular(
              {this.imageN = '', this.flowerName = '', this.cost = 0, this.count = 0});
}
```

3. **User Model :**

```
class User{
            String userName;
            String email;
            String password;
            User
            ({ this.userName='',this.email='',this.password='' });
            Map<String, dynamic> toMap() {
            return {
                    'userName': userName,
                    'email':email,
                    'password':password
            };
     }
            @override
            String toString() {
                    return 'User{userName: $userName, email:$email ,password:$password}';
            }
    }
```

4. **Database Helper Model :**

```
class DatabaseHelper{
         static final _dbName='loinDB.db';
        static final _dbVersion=1;
        static final _tableName='User';
        static final _columnUserName='userName';
        static final _columnEmail='email';
        static final _columnPAssword='password';
         DatabaseHelper._privateConstructor();
 }
```

5. **Input Field Model**

```
class inputField extends StatelessWidget {
          final User user;
          final String label;
          final IconData iconData;
          final int which;
```

```
        final bool hide; }
```

6.2 Implementation Support

6.2.1 Installation of Visual Studio Code

- Download VS code from **https://code.visualstudio.com/download.**
- Download the Visual Studio Code installer for suitable OS. Once it is downloaded, run the installer (VSCodeUserSetup-{version}.exe). Then, run the file – it will only take a minute.
- Accept the agreement and click "next."
- After accepting all the requests press finish button. By default, VS Code installs under: the desires path the user wishes to download

6.2.2 Installation of Flutter

- Download the following installation bundle to get the latest stable release of the Flutter SDK **https://docs.flutter.dev/get-started/install/windows**
- Extract the zip file and place the contained flutter in the desired installation location for the Flutter SDK
- Update the Path
    - From the Start search bar, enter 'env' and select Edit environment variables for your account.
    - Under User variables check if there is an entry called Path:
        - If the entry exists, append the full path to flutter\bin using ; as a separator from existing values.
        - If the entry doesn't exist, create a new user variable named Path with the full path to flutter\bin as its value.

6.3 PseudoCodes

6.3.1 PseudoCode for Database Connectivity

```
Future<Database?> get database async{
        if(_database!=null)
              return _database;
      _database=await _initiateDatabase();
              return _database;
 }

 _initiateDatabase () async{
  Directory directory=await getApplicationDocumentsDirectory();
  String path=join(directory.path,_dbName);
  return await openDatabase(
       path,
      version:_dbVersion,
       onCreate:_onCreate
  );
 }
```

6.3.2 Creating Table in the Database

```
      Future? _onCreate(Database db, int version){
        db.execute(
         '''CREATE TABLE $_tableName(
           $_columnUserName text not null,
           $_columnEmail text not null,
           $_columnPAssword text not null
         )'''
        );
```

```
}
```

6.3.3 Inserting Data into Table

```
Future insert(Map<String,dynamic> row) async{
   Database? db=await instance.database;
   await db?.insert(_tableName,row);
 }
```

6.3.4 Navigation across screens

```
initialRoute: '/',
routes: {
        '/':(context)=>splashScreen(),
         '/login':(context)=>loginScreen(),
          '/home':(context)=>homeScreen(),
          '/signin':(context)=>signInScreen(),
 },
```

6.3.5 Validation for login

```
onSaved: (value){
        if(which==0)
                    user.userName=value as String;
        else if(which==1)
                     user.email=value as String;
        else
          user.password=value as String;
        return;
 },
```

6.3.6 Incrementing the Quantity count

```
FloatingActionButton(
            child: Icon(
                  Icons.add,
                    color: Colors.white,
               ),
               onPressed: () {
                       setState(() {
                        widget.popular.count++;
                       });
                },
}
```

6.3.7 Decrementing the count quantity

```
FloatingActionButton(
          child:  Icon(Icons.remove, color: Colors.white),
                 onPressed: () {
                       setState(() {
                        if (widget.popular.count > 0)
                          widget.popular.count--;
                   });
                  },
)
```

6.3.8 Splash Screen implementation

```
void nextScreen(){
        Future.delayed(Duration(seconds: 2),(){
        Navigator.pushReplacementNamed(context, '/login');
        };
```

Chapter 7

Testing

7.1 Introduction

Testing is a process of executing a program with the interest of finding an error. A good test is one that has high probability of finding the yet undiscovered error. Testing should systematically uncover different classes of errors in a minimum amount of time with a minimum number of efforts. Two classes of inputs are provided to test the process.

1. A software configuration that includes a software requirement specification, a design specification and source code.
2. A software configuration that includes a test plan and procedure, any testing tool and test cases and their expected results.

7.2 Levels Of Testing

7.2.1 Unit Testing

Unit testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

	Description	Input Data	Actual Output	Expected Output	Status
1	In login screen	Entering non existing email ID and non existing password	Stays on the login screen	Stays on the login screen	Pass
2	In login screen	Entering non existing email ID and existing password	Stays on the login screen	Stays on the login screen	Pass

3	In login screen	Entering existing email ID and correct password	Stays on the login screen	Stays on the login screen	Pass
4	In login screen	Entering existing email ID and corresponding password	Login	Login	Pass
5	In signup screen	Enter name, email Id and password	Sign up and proceed to login	Signup and proceed to login	Pass
6	In home screen	The plus signing clicking	Increments the values	Increments the values	Pass
6	In home screen	The minus signing clicking	Decrements the values	Decrements the values	Pass

Table 7.1 Unit Testing

7.2.2 Integration Testing

Integration testing is also taken as integration and testing. This is the major testing process where parts of the program are working fine or not. This testing can be done by choosing the options in the program and by giving suitable inputs.

	Description	Input Data	Actual Output	Expected Output	Status
1	In login screen	Entering non existing email and non existing password	Checks with all values on the database if the email Id exists. Stays on screen as invalid entry	Checks with all values on the database if the email Id exists. Stays on screen as invalid entry	Pass
2	In login screen	Entering existing	Checks if the	Checks if the	Pass

		email and non existing password	email Id exists and then checks if the password entered matches with the database value. fails check and remains on same page.	email Id exists and then checks if the password entered matches with the database value. fails check and remains on same page	
3	In login screen	Entering existing Email Id and password	Check if the email Id exists and then check if the password entered matches with the database value. Goes to home page	Check if the email Id exists and then check if the password entered matches with the database value. Goes to home page	Pass
4	In HomeScreen	Clicking the add button	Increments the values present for that particular flower object from the existing value	Increments the values present for that particular flower object from the existing value	Pass

Table 7.2 Integration Testing

7.2.3 System Testing

System testing is defined as testing of a complete and fully integrated software product. This testing falls in black-box testing wherein knowledge of the inner design of the code is not a prerequisite and is done by the testing team. System testing is done after integration testing is complete. System testing should test functional and non-functional requirements of the software.

	Description	Input Data	Actual Output	Expected Output	Status
1	In Sign Up screen	Enter user name,	The data gets	The data gets	Pass

		email and password	stored into the database and goes to the next page	stored into the database and goes to the next page	
2	In login screen	Enter username and password	Check if the email Id and password exists in the database	Check if the email Id and password exists in the database	Pass

Table 7.3 System Testing

7.2.4 Validation Testing

In this, requirements established as part of software requirements analysis are validated against the software that has been constructed. Validation testing provides final assurance that software meets all functional, behavioral and performance requirements. Validation can be defined in many ways but a simple definition is that validation succeeds when software Function in a manner that can be reasonably by the customer.

	Description	Input Data	Actual Output	Expected Output	Status
1	In Sign Up screen	If username not entered	Message stating enter username	Message stating enter username	Pass
2	In Sign Up screen	If email not entered	Message stating enter email	Message stating enter email	Pass
3	In Sign Up screen	If password not entered	Message stating enter password	Message stating enter password	Pass
2	In login screen	If email not entered	Message stating enter email	Message stating enter email	Pass
3	In login screen	If password not entered	Message stating enter password	Message stating enter password	Pass

Table 7.4 Validation Testing

7.2.5 Output Testing

After preparing test data, the system under study is tested using the test data. While testing the system using test data, errors are again uncovered and corrected by using above testing and corrections are also noted for future use.

	Description	Input Data	Actual Output	Expected Output	Status
1	In splash Screen		Moves two login screen after a time lapse of 2 seconds	Moves two login screen after a time lapse of 2 seconds	Pass
2	In Login screen	On invalid data entry	Stays on same screen	Stays on same screen	Pass
3	In Login Screen	Clicking on sign up	Goes to sign up page	Goes to sign up page	Pass
4	In signup screen	On entering valid data	Goes to login screen	Goes to login screen	Pass
4	In Login Screen	Entering valid data for fields	Goes to home page	Goes to home page	Pass

Table 7.5 Output Testing

7.2.6 User Acceptance Testing

User acceptance testing is a type of testing performed by the end user or the client to verify/accept the software application to the production environment.

	Description	Input Data	Actual Output	Expected Output	Status
1	Login Screen : Is the username text field taking input ?	String	The entered string should be displayed	The entered string should be displayed	Pass
2	Login Screen : Is the password field taking input ?	String	The entered text should be hidden	The entered text should be hidden	Pass

3	Sign Up Screen : Is the username text field taking input ?	String	The entered string should be displayed	The entered string should be displayed	Pass
4	Sign Up Screen: Is the name text field taking input ?	String	The entered string should be displayed	The entered string should be displayed	Pass
5	Sign Up Screen: Is the password field taking input ?	String	The entered text should be hidden	The entered text should be hidden	Pass

Table 7.6 Use Acceptance Testing

Chapter 8

Results

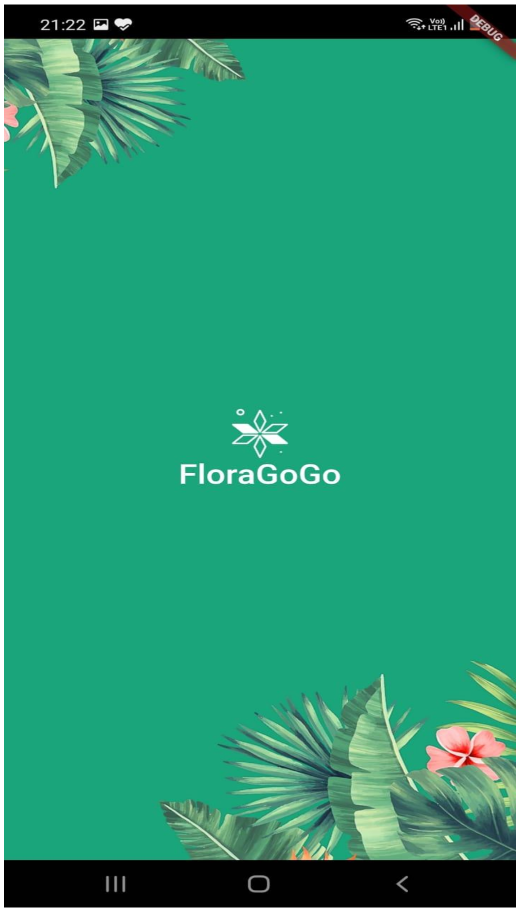
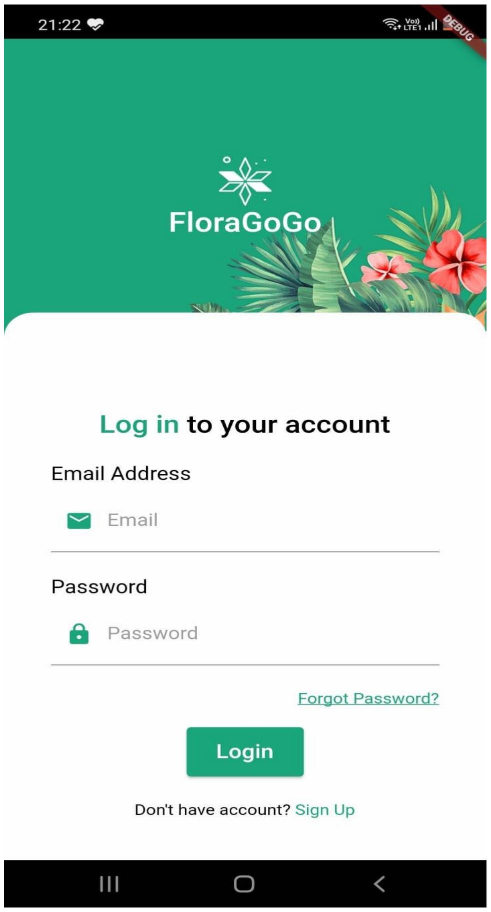


Fig 8.1 Splash/Launch Screen



Log in to your account

Email Address

Email

Password

Password

Forgot Password?

Login

Don't have account? Sign Up

Fig 8.2 LogIn Screen

The above fig 8.1 gets displayed when the app is opened and the screen is called the Launch or the Splash screen which automatically navigates to the login screen after a time lapse of 2 seconds.

The figure 8.2 which is the login Screen accepts input from the user and validates the entered data with the data stored in the database.If the inserted data is correct than navigation to menu screen takes place. If there is a new user who don’t have an account this page gives the option of sign up which navigates to another page .

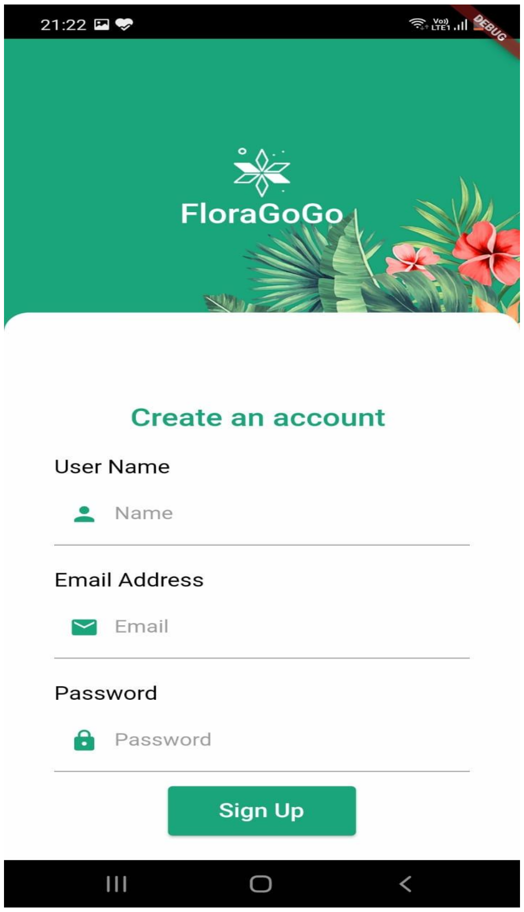


Fig 8.3 SignUp Page

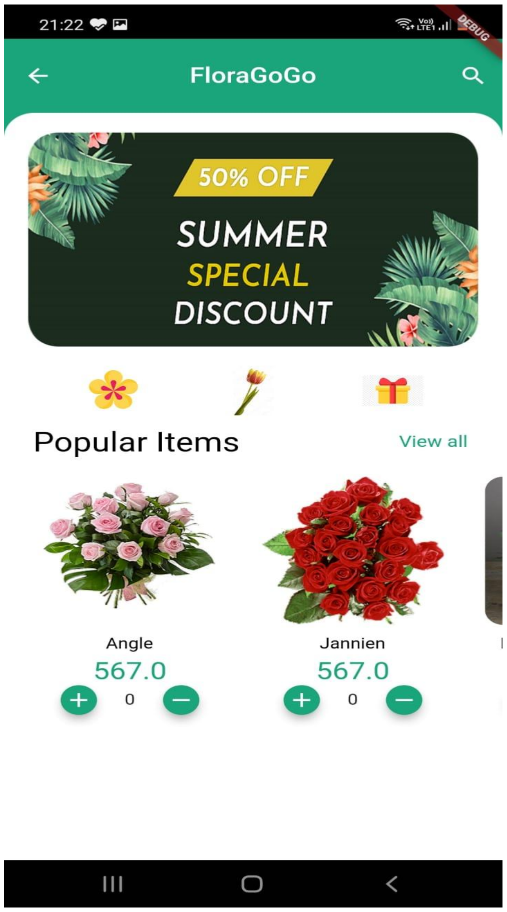


Fig 8.4 Menu Screen

The fig 8.3 shows our sign up page where the new user will have to enter details like username , email address and password . The same gets stored into the database.Once the data gets stored in the database the user is navigated back to the login page.

The figure 8.4 displays the menu card where all the bouquet images, the bouquet name and the cost is also displayed.

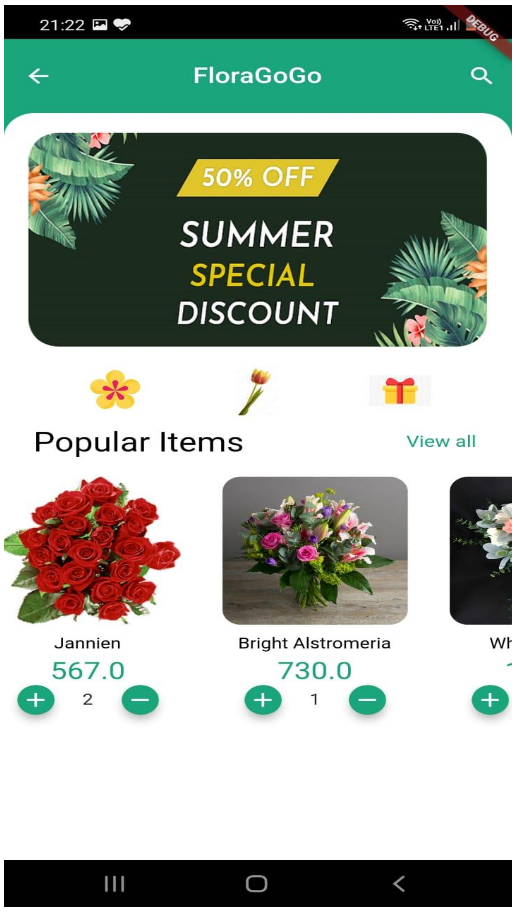


Fig 8.5 Quantity Updation

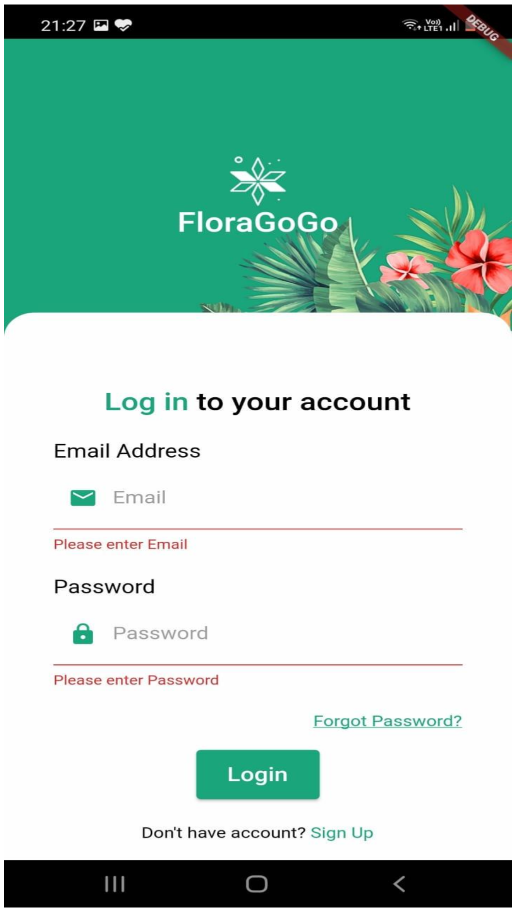


Fig 8.6 Validation of empty input

Fig 8.5 displays the screen where the user has increased the quantity of flower

Fig 8.6 displays the case of validation where the user has not entered the email address and the password . So error message pops up asking the user to enter the same .

Chapter 9

Conclusions and Future Work

Conclusions

FloraGoGo, the online flower selling application is a 4 screen application . The developed application focuses purely on the UI. The application performs basic validation of the user using the database. It has flowers that are available in the online application and has an option of choosing the quantity that is needed.

Future Work

1. Forgot Password Screen can be implemented
2. Menu screen can be designed a better way
3. Features to add items into cart can be done .
4. Payment Page and Billing Page can be designed.
5. Backend for this application can be developed.

REFERENCES

[1] https://flutter.dev/

[2] https://www.javatpoint.com/flutter

[3] https://www.geeksforgeeks.org/flutter-tutorial/

[4] https://medium.com/flutter

[5] https://stackoverflow.com/questions/tagged/flutter