

# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018



**An Internship Project Report  
on**

***Plant App***

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

**Submitted By**

**Jyotsna Singh**

**1RN18IS057**

**Under the Guidance of**

**Mrs. Chandan Rani S R**

**Assistant Professor**

**Department of ISE**



ESTD: 2001

*An Institute with a Difference*

**Department of Information Science and Engineering**

**RNS Institute of Technology**

**Dr. Vishnuvaradhana Road, Rajarajeshwari Nagar post,  
Channasandra, Bengaluru-560098**

**2021-2022**

# RNS INSTITUTE OF TECHNOLOGY

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

## DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING



## CERTIFICATE

Certified that the Internship work entitled *Plant App* has been successfully completed by **Jyotsna Singh (1RN18IS057)** a bonafide student of **RNS Institute of Technology, Bengaluru** in partial fulfillment of the requirements of 8<sup>th</sup> semester for the award of degree in **Bachelor of Engineering in Information Science and Engineering** of **Visvesvaraya Technological University, Belagavi** during the 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.

---

**Mrs. Chandan Rani S R**

Internship Guide  
Assistant Professor  
Department of ISE

---

**Dr. Suresh L**

Professor and HoD  
Department of ISE  
RNSIT

---

**Dr. M K Venkatesha**

Principal  
RNSIT

### External Viva

**Name of the Examiners**

**Signature with Date**

1. \_\_\_\_\_

1. \_\_\_\_\_

2. \_\_\_\_\_

2. \_\_\_\_\_

# DECLARATION

I, **Jyotsna Singh [USN: 1RN18IS057]** student of VIII Semester BE, in Information Science and Engineering, RNS Institute of Technology hereby declare that the Internship work entitled ***Plant App*** has been carried out by me 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 the academic year 2021-2022.

Place: Bengaluru

Date:

**JYOTSNA SINGH**  
**(1RN18IS057)**

## **ABSTRACT**

I propose online plant exploration app which enables customers to straightforwardly know about all the different kinds of plant species available. It's practically not possible to go physically and explore these things. Our app gives a brief idea about the products available along with other kinds of stuff such as sprays or pots etc. This makes life much easier for plant lovers.

There are many articles available currently on the internet. But there are few which provide a better understanding of different species of plants, seeds, and fertilizers or sprayers used for the same all in one place. To develop a User-friendly plant application that will contain descriptions of products.

To develop an application that will contain solutions to the above problems. With this application, the user will come to know about varieties of plants and seeds, etc., and can learn additional information about them. Also by this application, a user can expand his/her knowledge of the world.

# 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 **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 with all the facilities.

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

I place our heartfelt thanks to **Mrs. Chandan Rani S R** Assistant Professor and HOD, 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 has guided me to complete the same successfully.

I also thank our internship coordinator, **Dr. R Rajkumar**, Associate Professor, Dept 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.

**JYOTSNA SINGH**

**(1RN18IS057)**



# TABLE OF CONTENTS

CERTIFICATE	i
DECLARATIONS	iii
ABSTRACT	iv
ACKNOWLEDGMENT	v
TABLE OF CONTENTS	vi
LIST OF FIGURES	vii
LIST OF TABLES	viii
LIST OF ABBREVIATIONS	ix
1. INTRODUCTION	1
2. LITERATURE REVIEW	4
3. ANALYSIS	5
4. SYSTEM DESIGN	7
5. IMPLEMENTATION	10
6. TESTING	17
7. RESULTS	20
8. CONCLUSION AND FUTURE ENHANCEMENTS	22
REFERENCES	23

# LIST OF FIGURES

<b>Figure No.</b>	<b>Figure Description</b>	<b>Page No.</b>
Figure 4.1	Greeting page	7
Figure 4.2	Login page	8
Figure 4.3	Schema design of the system	9
Figure 5.1	Greeting screen code	10
Figure 5.2	Login screen code	11
Figure 5.3	Explore page code	13
Figure 5.4	Browse page code	14
Figure 5.5	Setting page code	14
Figure 5.6	Detail screen code	15
Figure 5.7	Colors code	16
Figure 7.1	Home page	20
Figure 7.2	Display details	21
Figure 7.3	Browse page	21



## LIST OF TABLES

Table No.	Table Description	Page No.
Table 6.1	Testing table	19

# ABBREVIATIONS

Acronym	Description
JSON	JavaScript Object Notation
FCL	Flutter cycle Length
UI	User Interface
SDK	Software development Kit
VM	Virtual Machine
CLR	Common Language runtime
IE	Internet Explorer
API	Application Programming Interface
IoT	Internet of Things
VS	Visual Studio

# CHAPTER 1

## INTRODUCTION

### Introduction to Flutter

Flutter is Google's Mobile SDK to build native iOS and Android, Desktop (Windows, Linux, macOS), and Web apps from a single codebase. When building applications with Flutter everything towards Widgets – the blocks with which the flutter apps are built.

They are structural elements that ship with a bunch of material design-specific functionalities and new widgets can be composed out of existing ones too. The process of composing widgets together is called composition. The User Interface of the app is composed of many simple widgets, each of them handling one particular job. That is the reason why Flutter developers tend to think of their flutter app as a tree of widgets.

### History

Flutter launched a project called Sky which at the beginning worked only on Android. Flutter's goal is to enable developers to compile for every platform using its own graphic layer rendered by the Skia engine. Here's a brief presentation of Flutter's relatively short history.

Flutter is a free and open-source mobile UI framework created by Google and released in May 2017. In a few words, this allows you to create a native mobile application with only one code. It means that you can use one programming language and one codebase to create two different apps (IOS and Android).

The first version of Flutter was known by the codename "Sky" and ran on the Android operating system. It was unveiled at the 2015 Dart developer summit with the stated intent of being able to render consistently at 120 frames per second. During the keynote of Google Developer Days in Shanghai in Sept 2018, Google announced Flutter Release Preview 2, which is the last big release before Flutter 1.0. On December 4th of that year, Flutter 1.0 was released at the Flutter Live event, denoting the first "stable" version of the Framework. On Dec 2019, Flutter 1.12 was released at the Flutter Interactive event.

On September 8th, 2021, Dart 2.14 and Flutter 2.5 were released by Google. The update brought improvements to the Android full-screen mode and the latest version of Google's Material Design called Material You. Dart received two new updates, standardizing lint conditions and marking support for Apple Silicon as stable.

The current stable channel of Flutter is 3.0.2 and the Dart version is 2.17.3.

## **Framework-Architecture**

The major components of Flutter include:

- Dart platform
- Flutter engine
- Foundation library
- Design-specific widgets
- Flutter Development Tools (DevTools)

## **Dart platform**

Flutter apps are written in the Dart language and make use of many of the language's more advanced features.

On Windows, macOS, and Linux Flutter runs in the Dart virtual machine, which features a just-in-time execution engine. While writing and debugging an app, Flutter uses Just in Time compilation, allowing for "hot reload", with which modifications to source files can be injected into a running application. Flutter extends this with support for stateful hot reload, where in most cases changes to source code are reflected immediately in the running app without requiring a restart or any loss of state.

For better performance, release versions of Flutter apps targeting Android and iOS are compiled with ahead-of-time (AOT) compilation.

On the subject of desktop, we've made much progress with macOS support. For the first time, you can use the release mode to build a fully-optimized macOS application using Flutter, and we've been working to expand the Material design system in Flutter to support apps that are designed for desktop-class form factors. It's now possible to build and consume web plug-ins, enabling Flutter apps to take advantage of a growing ecosystem of Dart components for everything from Firebase to the latest web APIs.

## **Flutter engine**

The Flutter Engine is a portable runtime for hosting Flutter applications. It implements Flutter's core libraries, including animation and graphics, file and network I/O, accessibility support, plugin architecture, and a Dart runtime and compile toolchain. Most developers will interact with Flutter via the Flutter Framework, which provides a modern, reactive framework, and a rich set of platform, layout and foundation widgets. Flutter works with existing code, is used by developers and organizations around the world, and is free and open source. The Flutter Engine is a portable runtime for hosting Flutter applications.

## **Foundation library**

The Foundation library, written in Dart, provides basic classes and functions that are used to construct applications using Flutter, such as APIs to communicate with the engine. The features defined in this library are the lowest-level utility classes and functions used by all the other layers of the Flutter framework.

## **Design-specific widgets**

The Flutter framework contains two sets of widgets that conform to specific design languages: Material Design widgets implement Google's design language of the same name, and Cupertino widgets implement Apple's iOS Human interface guidelines.

## **CHAPTER 2**

# **LITERATURE REVIEW**

### **Flutter**

Flutter as a cross-compiler compared to two native applications made of Kotlin and Android studio and swift and XCode, in terms of CPU performance.

A survey was created to see if there was a difference in the perception of users with regards to appearance and animations. A literature study was conducted to strengthen the results from the experiment and survey and to give a background to the subject.

Flutter is best to use when building smaller to medium-sized applications but has the potential to grow to overcome its current drawbacks in the animation department.

### **Plant App**

Biodiversity is declining steadily throughout the world. The current rate of extinction is largely the result of direct and indirect human activities.

Building accurate knowledge of the identity and the geographic distribution of plants is essential for future biodiversity conservation.

Therefore, rapid and accurate plant identification is essential for effective study and management of biodiversity.

Our app helps people to know about different varieties of plants, seeds, fertilizers, flowers, sprayers, pots, etc which is very useful as well as interesting for today's generation as we are very less aware within this modern world.

## CHAPTER 3

# ANALYSIS

### 3.1 Hardware and Software Requirements

The Hardware requirements are very minimal and the program can be run on most of the machines.

Processor	:	Pentium 4 Processor
Processor Speed	:	2.4 GHz
RAM	:	2 GB
Storage Space	:	40 GB

The software requirements are very minimal and the program can be run on the machines with these requirements satisfied:

Editor	:	Visual Studio Code
Operating System	:	Windows/Mac OS
IDE	:	VS Code
Backend Tool	:	SQLite

### 3.2 Tools/ Languages/ Platform

Various tool used in making this project is given below:

Editor/IDE	:	Visual Studio Code
Operating System	:	Windows/Mac OS
Languages	:	Dart, Swift, SQLite
Backend Tool	:	SQLite

### **3.3 Functional Requirements Flutter**

Flutter is Google's Mobile SDK to build native iOS and Android apps from a single codebase. When building applications with Flutter everything towards Widgets – the blocks with which the flutter apps are built. The User Interface of the app is composed of many simple widgets, each of them handling one particular job. That is the reason why Flutter developers tend to think of their flutter app as a tree of widgets.

Compared to its contemporary technologies like React Native, Kotlin, and Java, Flutter is much better in regard to having a Single Codebase for Android and iOS, Reusable UI and Business Logic, high compatibility, performance, and productivity.

### **3.4 Dart**

Dart is an open-source general-purpose programming language developed by Google. It supports application development in both client and server-side. But it is widely used for the development of android apps, iOS apps, IoT(Internet of Things), and web applications using the Flutter Framework.

Syntactically, Dart bears a strong resemblance to Java, C, and JavaScript. It is a dynamic object-oriented language with closure and lexical scope. The Dart language was released in 2011 but came into popularity after 2015 with Dart 2.0.

### **3.5 SQLite**

SQLite is a self-contained, high-reliability, embedded, full-featured, public- domain, SQL database engine. It is the most used database engine in the world. It is an in-process library and its code is publicly available.

It is free for use for any purpose, commercial or private. It is basically an embedded SQL database engine. Theinternship\_final.docx SQLite database file format is cross-platform



## CHAPTER 4

# SYSTEM DESIGN

### 4.1 GREETINGS PAGE WIDGET TREE

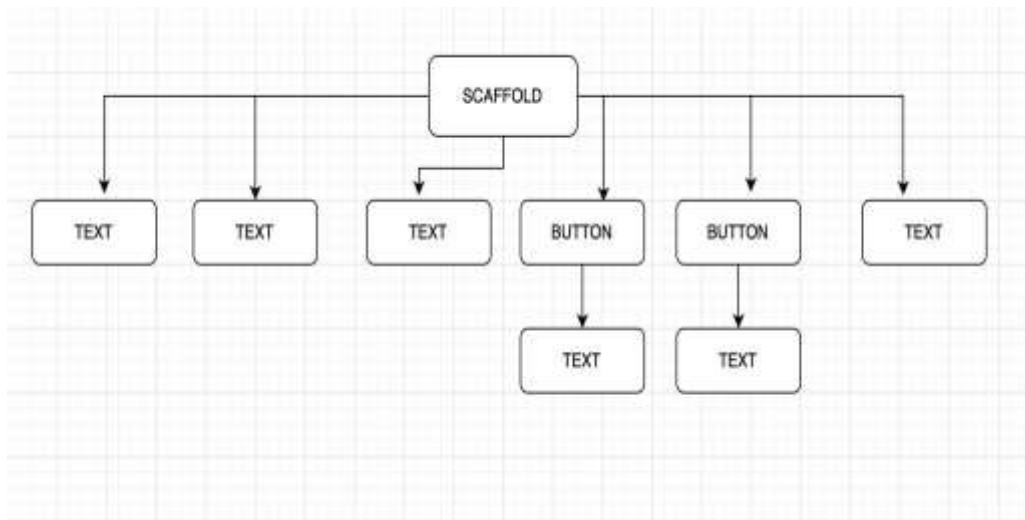


Figure 4.1: Greeting page

As shown in figure 4.1, we have the widget tree of the greeting page which has 4 text boxes and two buttons containing texts and all are connected in hierarchical order. Scaffold is the master and other text boxes and button boxes are child connections.

This text boxes in greeting page are various welcome messages for the user and button contains login and signup text in it.

## 4.2 LOGIN PAGE WIDGET PAGE

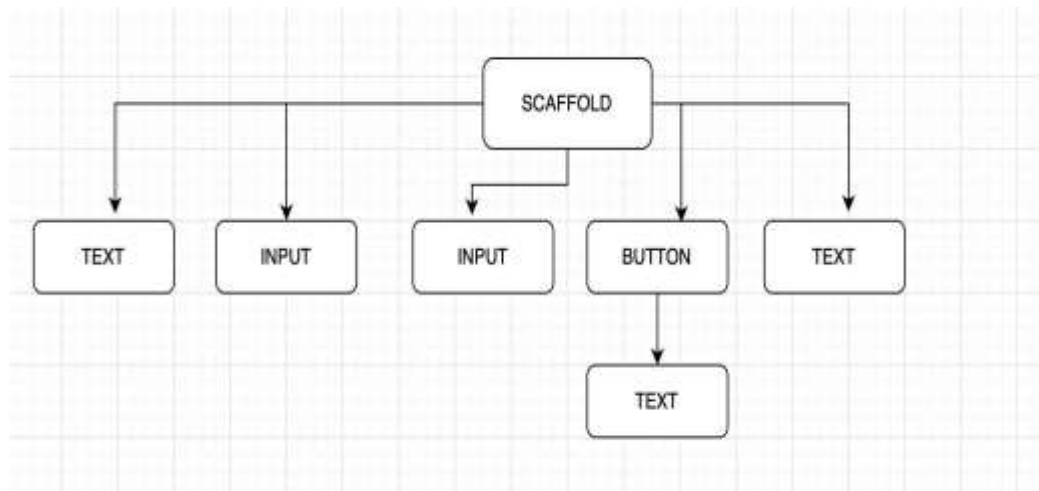


Figure 4.2: Login page

As shown in figure 4.2, we have the widget tree of the login page which has two textboxes where text element has user login text written in it along with that we have two input boxes for username and password entry and one button for login. Scaffold is the master and other text boxes and button boxes are child connections.

### 4.3 SCHEMA DESIGN

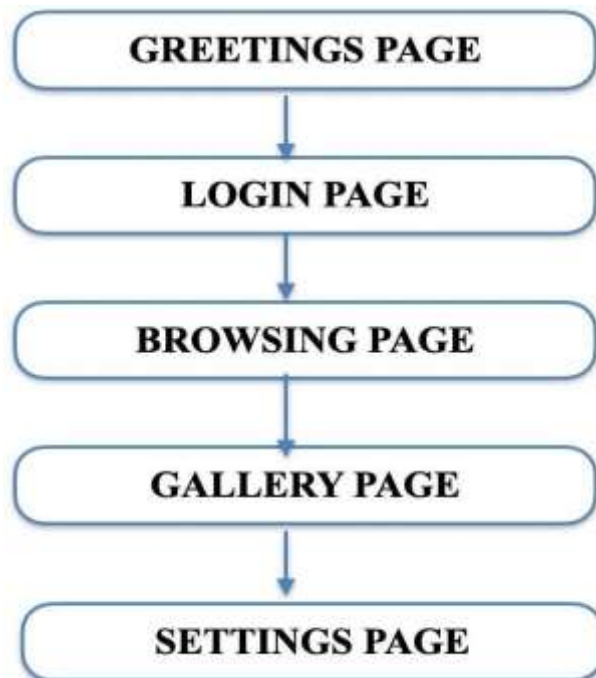


Figure 4.3: Schema design of the system

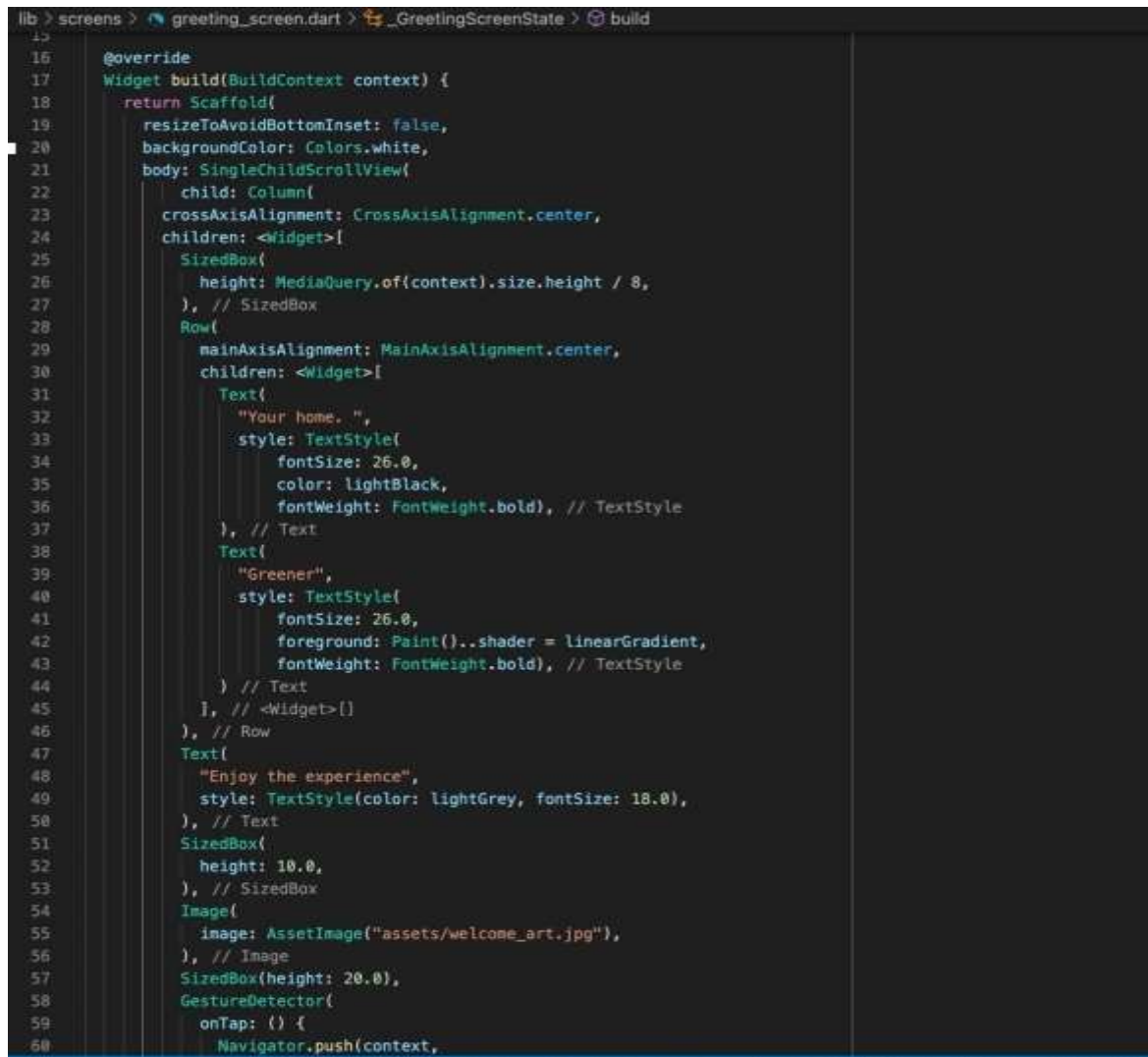
The adjacent figure is the schema diagram of the plant app application as shown in figure 4.3.

The sequence of the app pages is mentioned here, firstly user gets a greeting page having signup or login options then next is the login page after an user successfully sign in then the browsing page having plant images as samples and search bar if user wants to search something that is followed by the gallery page which has plenty of images for the searched item, and at last setting page where user can change his/her details. All pages are interconnected and directs to each other.

## CHAPTER 5

# IMPLEMENTATION

### 5.1. main.dart



```
lib > screens > greeting_screen.dart > _GreetingScreenState > build
15
16 @override
17 Widget build(BuildContext context) {
18   return Scaffold(
19     resizeToAvoidBottomInset: false,
20     backgroundColor: Colors.white,
21     body: SingleChildScrollView(
22       child: Column(
23         crossAxisAlignment: CrossAxisAlignment.center,
24         children: <Widget>[
25           SizedBox(
26             height: MediaQuery.of(context).size.height / 8,
27           ), // SizedBox
28           Row(
29             mainAxisAlignment: MainAxisAlignment.center,
30             children: <Widget>[
31               Text(
32                 "Your home. ",
33                 style: TextStyle(
34                   fontSize: 26.0,
35                   color: lightBlack,
36                   fontWeight: FontWeight.bold), // TextStyle
37               ), // Text
38               Text(
39                 "Greener",
40                 style: TextStyle(
41                   fontSize: 26.0,
42                   foreground: Paint()..shader = LinearGradient,
43                   fontWeight: FontWeight.bold), // TextStyle
44               ), // Text
45             ], // <Widget>[]
46           ), // Row
47           Text(
48             "Enjoy the experience",
49             style: TextStyle(color: lightGrey, fontSize: 18.0),
50           ), // Text
51           SizedBox(
52             height: 10.0,
53           ), // SizedBox
54           Image(
55             image: AssetImage("assets/welcome_art.jpg"),
56           ), // Image
57           SizedBox(height: 20.0),
58           GestureDetector(
59             onTap: () {
60               Navigator.push(context,
```

Figure 5.1: Greeting screen code

This is the home screen code as shown in figure 5.1, leading to the greeting screen.

Here we have stateful widget for elements that we need to change later else we use stateless widget. We defined widget sizing attributes to position different elements based on the grid of cells that's having a defined height and width. These include `targetCellWidth`, `targetCellHeight`. Text styles is for defining the color, font style and font size of the text.



## 5.2 Login\_screen.dart

```

lib > screens > login_screen.dart > _LoginScreenState > build
1  import 'package:flutter/material.dart';
2  import './colors.dart';
3  import './browse_screens.dart';
4
5  class LoginScreen extends StatefulWidget {
6    LoginScreen({Key key}) : super(key: key);
7
8    _LoginScreenState createState() => _LoginScreenState();
9  }
10
11 class _LoginScreenState extends State<LoginScreen> {
12   @override
13   Widget build(BuildContext context) {
14     return Scaffold(
15       backgroundColor: Colors.white,
16       appBar: PreferredSize(
17         preferredSize: Size.fromHeight(140.0),
18         child: AppBar(
19           centerTitle: false,
20           flexibleSpace: Container(
21             margin: EdgeInsets.symmetric(vertical: 35.0, horizontal: 20.0),
22             child: Align(
23               alignment: Alignment.bottomLeft,
24               child: Text(
25                 "Login",
26                 style: TextStyle(
27                   color: Color(0xFF323643),
28                   fontSize: 26.0,
29                   fontWeight: FontWeight.w700), // TextStyle
30               ), // Text
31             ), // Align
32           ), // Container
33           elevation: 0.0,
34           backgroundColor: Colors.white,
35           iconTheme: IconThemeData(color: Color(0xFFC5CCD6)),
36         ), // AppBar
37       ), // PreferredSize
38       body: SingleChildScrollView(
39         child: Container(
40           width: MediaQuery.of(context).size.width - 40,
41           margin: EdgeInsets.symmetric(horizontal: 20.0),
42           child: Center(
43             child: Column(
44               mainAxisAlignment: MainAxisAlignment.center,
45               crossAxisAlignment: CrossAxisAlignment.center,
46               children: <div data-bbox="148 664 624 682" data-label="Caption">


Figure 5.2: Login screen code


```

This is login screen code as shown in figure 5.2 where firstly we design the scaffold then it's child widget tree code.

It is stateful widget. Background color is white, and then we arrange horizontally all the text boxes and in the last we place login button in the central alignment.

We created one container for user details and other one for password entry, then one login button after the details they acts as the child of the scaffold container.

### 5.3 explore.dart

```

lib > screens > explore_screen.dart > ...
1  import 'package:flutter/material.dart';
2  import 'package:colors.dart';
3
4  class ExploreScreen extends StatefulWidget {
5    ExploreScreen({Key key}) : super(key: key);
6
7    _ExploreScreenState createState() => _ExploreScreenState();
8  }
9
10 class _ExploreScreenState extends State<ExploreScreen> {
11   @override
12   Widget build(BuildContext context) {
13     return Scaffold(
14       backgroundColor: Colors.white,
15       appBar: PreferredSize(
16         preferredSize: Size.fromHeight(130.0),
17         child: AppBar(
18           centerTitle: false,
19           flexibleSpace: Container(
20             margin: EdgeInsets.only(left: 20.0, bottom: 30.0, right: 20.0),
21             child: Align(
22               alignment: Alignment.bottomLeft,
23               child: Row(
24                 mainAxisAlignment: MainAxisAlignment.spaceBetween,
25                 children: <Widget>[
26                   Text(
27                     "Browse",
28                     style: TextStyle(
29                       color: Color(0xFF323643),
30                       fontSize: 26.0,
31                       fontWeight: FontWeight.w700), // TextStyle
32                   ), // Text
33                   Container(
34                     width: 160.0,
35                     height: 40,
36                     padding: EdgeInsets.all(5.0),
37                     decoration: BoxDecoration(
38                       color: Color(0xFF8BE93).withOpacity(0.06),
39                       borderRadius: BorderRadius.circular(6.0)), // BoxDecoration
40                     child: TextField(
41                       cursorColor: Color(0xFF8BE93).withOpacity(0.06),
42                       decoration: InputDecoration(
43                         labelText: "Search",
44                         labelStyle: TextStyle(
45                           color: Color(0xFFC5C6D6), fontSize: 14), // TextStyle
46
47       screens > explore_screen.dart > ...
48       Image(image: AssetImage("assets/images/explore0.png")),
49       Image(image: AssetImage("assets/images/explore1.png")),
50       ], // <Widget>[]
51     ), // Row
52     SizedBox(height: 10.0),
53     Image(
54       image: AssetImage("assets/images/explore0.png"),
55     ), // Image
56     ], // <Widget>[]
57   ), // Column
58   ), // Center
59   ), // SingleChildScrollView
60   ), // Container
61   Positioned(
62     bottom: 0.0,
63     left: 0.0,
64     width: MediaQuery.of(context).size.width,
65     height: 100.0,
66     child: Container(
67       decoration: BoxDecoration(
68         gradient: LinearGradient(colors: [
69           Colors.white.withOpacity(0.01),
70           Colors.white.withOpacity(1)
71         ], begin: Alignment.topCenter, end: Alignment.bottomCenter), // LinearGradient // BoxDecoration
72       child: Center(
73         child: Container(
74           width: 150.0,
75           height: 40,
76           decoration: BoxDecoration(
77             gradient: LinearGradient(
78               colors: [primaryColor1, secondaryColorGreen],
79               begin: Alignment.topLeft,
80               end: Alignment.bottomRight), // LinearGradient
81             borderRadius: BorderRadius.circular(6.0),
82             boxShadow: [
83               BoxShadow(
84                 color: Color(0xFF9DA3B4).withOpacity(0.1),
85                 blurRadius: 65.0,
86                 offset: Offset(0.0, 15.0)) // BoxShadow
87             ], // BoxDecoration
88           child: Center(
89             child: Text(
90               "Filters",
91               style: TextStyle(
92                 color: Color(0xFFBFBFB),

```

Explore page code having different plants and related items that user can search in search box.

```

> screens > explore_screen.dart > ...
1      Image(image: AssetImage("assets/images/explore4.png")),
2      Image(image: AssetImage("assets/images/explore5.png")),
3    ], // <Widget>[]
4  ), // Row
5  SizedBox(height: 10.0),
6  Image(
7    image: AssetImage("assets/images/explore6.png"),
8  ), // Image
9  ], // <Widget>[]
10 ], // Column
11 ], // Center
12 ], // SingleChildScrollView
13 ], // Container
14 Positioned(
15   bottom: 0.0,
16   left: 0.0,
17   width: MediaQuery.of(context).size.width,
18   height: 100.0,
19   child: Container(
20     decoration: BoxDecoration(
21       gradient: LinearGradient(colors: [
22         Colors.white.withOpacity(0.01),
23         Colors.white.withOpacity(1)
24       ], begin: Alignment.topCenter, end: Alignment.bottomCenter)), // LinearGradient // BoxDecoration
25     child: Center(
26       child: Container(
27         width: 150.0,
28         height: 40,
29         decoration: BoxDecoration(
30           gradient: LinearGradient(
31             colors: [primaryColor1, secondaryColorGreen],
32             begin: Alignment.topLeft,
33             end: Alignment.bottomRight), // LinearGradient
34           borderRadius: BorderRadius.circular(6.0),
35           boxShadow: [
36             BoxShadow(
37               color: Color(0xFF9DA384).withOpacity(0.1),
38               blurRadius: 65.0,
39               offset: Offset(0.0, 15.0)) // BoxShadow
40           ], // BoxDecoration
41       child: Center(
42         child: Text(
43           "Filters",
44           style: TextStyle(
45             color: Color(0xFFFBFBFB),

```

Figure 5.3: Explore page code

As shown in figure 5.3 we have explore page code which makes connects 6 containers each leading to 6 different pages.

In this code, in the scaffold we have a text container acting as a search box, and is followed by child containers containing images in AssetImage.

The boxes are placed in linear gradient style that is one next to each other. And the boxes have rounded corners which is set by the property borderRadius, and the position is set by offset property.







## 5.4. browse.dart

```
lib > screens > % browse_screens.dart > [e] products.dart
1 import 'package:flutter/material.dart';
2 import './colors.dart';
3 import './detail_screen.dart';
4 import './settings_screen.dart';
5
6 const products = [
7   {'image': 'assets/plants.jpg', 'count': '147', 'name': 'Plants'},
8   {'image': 'assets/seeds.jpg', 'count': '16', 'name': 'Seeds'},
9   {'image': 'assets/flowers.jpg', 'count': '68', 'name': 'Flowers'},
10  {'image': 'assets/sprayers.jpg', 'count': '17', 'name': 'Sprayers'},
11  {'image': 'assets/pots.jpg', 'count': '47', 'name': 'Pots'},
12  {'image': 'assets/fertilizers.jpg', 'count': '9', 'name': 'Fertilizers'},
13 ];
14
15 const inspirations = [
16   {'image': 'assets/plants.jpg', 'count': '147', 'name': 'Plants'},
17   {'image': 'assets/flowers.jpg', 'count': '68', 'name': 'Flowers'},
18 ];
19
20 const shop = [
21   {'image': 'assets/seeds.jpg', 'count': '16', 'name': 'Seeds'},
22   {'image': 'assets/sprayers.jpg', 'count': '17', 'name': 'Sprayers'},
23   {'image': 'assets/pots.jpg', 'count': '47', 'name': 'Pots'},
24   {'image': 'assets/fertilizers.jpg', 'count': '9', 'name': 'Fertilizers'},
25 ];
26
27 class BrowseScreen extends StatefulWidget {
28   const BrowseScreen({Key key}) : super(key: key);
29
30   @override
31   _BrowseScreenState createState() => _BrowseScreenState();
32 }
33
34 class _BrowseScreenState extends State<BrowseScreen>
35   with SingleTickerProviderStateMixin {
36   TabController controller;
37
38   final Shader linearGradient = LinearGradient(
39     colors: <Color>[primaryColor1, secondaryColorGreen],
40   ).createShader(new Rect.fromLTWH(0.0, 0.0, 200.0, 70.0)); // LinearGradient
41
42   @override
43   void initState() {
44     super.initState();
45     controller = TabController(vsync: this, length: 3);
46   }
47 }
```

Figure 5.4: Browse page code

This is the browsing page as shown in the figure 5.3 containing three functions for calling three different pages extending the same properties expect the content and the image collection.

## 5.5. settings.dart

```
class SettingsScreen extends StatefulWidget {
  SettingsScreen({Key key}) : super(key: key);
  _SettingsScreenState createState() => _SettingsScreenState();
}

class _SettingsScreenState extends State<SettingsScreen> {
  final Shader linearGradient = LinearGradient(
    colors: <Color>[primaryColor1, secondaryColorGreen],
  ).createShader(new Rect.fromLTWH(0.0, 0.0, 200.0, 70.0)); // LinearGradient

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      appBar: PreferredSize(
        preferredSize: Size.fromHeight(120.0),
        child: AppBar(
          centerTitle: false,
          flexibleSpace: Container(
            margin: EdgeInsets.only(left: 20.0, bottom: 20.0, right: 20.0),
            child: Align(
              alignment: Alignment.bottomLeft,
              child: Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: <Widget>[
                  Text(
                    "Settings",
                    style: TextStyle(
                      color: Color(0xFF323643),
                      fontSize: 20.0,
                      fontWeight: FontWeight.w700), // TextStyle
                ), // Text
                  CircleAvatar(
                    backgroundColor: Colors.black,
                    backgroundImage: AssetImage("assets/user.jpg"),
                  ), // CircleAvatar
                ], // Row
              ), // Align
            ), // Container
            elevation: 0.0,
            backgroundColor: Colors.white,
            // ...
          ), // Container
        ), // Row
      ), // Align
    ), // Container
  ); // Row
}
```

Figure 5.5: Setting page code

This figure 5.6 is setting page code where height, weight, axis of the page is specified.





Along with that we have other fonts and color details.

It contains setting of the user profile as well as log out option. User can change screen light for that there is slider provided. And other details are in the textboxes.

## 5.6 details.dart

```

1  import 'package:flutter/material.dart';
2  import './explore_screen.dart';
3
4  class DetailScreen extends StatefulWidget {
5    DetailScreen({Key key}) : super(key: key);
6
7    _DetailScreenState createState() => _DetailScreenState();
8  }
9
10 class _DetailScreenState extends State<DetailScreen> {
11   @override
12   Widget build(BuildContext context) {
13     return Scaffold(
14       appBar: AppBar(
15         actions: <Widget>[
16           IconButton(
17             icon: Icon(Icons.ware_horiz),
18             onPressed: () {},
19           ) // IconButton
20         ], // <Widget>[]
21       backgroundColor: Colors.white,
22       elevation: 8.0,
23       iconTheme: IconThemeData(color: Color(0xFFC5C0D6)),
24     ), // AppBar
25     body: SingleChildScrollView(
26       scrollDirection: Axis.vertical,
27       child: Column(
28         crossAxisAlignment: CrossAxisAlignment.start,
29         children: <Widget>[
30           Image(
31             image: AssetImage("assets/images/1.png"),
32           ), // Image
33           SizedBox(
34             height: 25.0,
35           ), // SizedBox
36           Container(
37             margin: EdgeInsets.only(left: 20.0, right: 20.0, bottom: 40.0),
38             child: Column(
39               crossAxisAlignment: CrossAxisAlignment.start,
40               children: <Widget>[
41                 Container(
42                   width: MediaQuery.of(context).size.width - 100,
43                   child: Text(
44                     "15 Best Plants That Thrive In Your Bedroom",
45                     style: TextStyle(
46                       color: Color(0xFFA4A4A4)
47                     )
48                   )
49                 ]
50             )
51           ]
52         ], // children
53       ), // Column
54     ), // SingleChildScrollView
55   ), // Scaffold
56 );

```

Figure 5.6: Detail screen code

Figure 5.6 is a detail specification where we store details of all the 6 items of our collection.

For that we have image assets and text boxes and arrow box to move from them.



## 5.7 color.dart

```
lib > screens > colors.dart > ...  
1  import 'package:flutter/material.dart';  
2    
3  const primaryColor1 = Color(0xFF0AC4BA);  
4  const secondaryColorGreen = Color(0xFF2BDA8E);  
5  const lightBlack = Color(0xFF323643);  
6  const lightGrey = Color(0xFFC5CCD6);  
7  const blueGrey = Color(0xFF9DA3B4);  
8
```

Figure 5.7: Colors code

Figure 5.7 are color constants which represent Material design's color palette. Instead of using an absolute color from these palettes, we considered using theme which defines the colors that most of the Material components use by default. Here we declared some color constants storing the color values. All the colors are mapped to their respective hexadecimal values.

Now instead of mentioning the hexadecimal values we will directly use the respective color variable names.





## CHAPTER 6

# TESTING

### 6.1 Introduction

Testing is a process of executing a program with the interest of finding an error. A good test is one that has a 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.

A software configuration that includes a software requirement specification, a design specification, and source code.

A software configuration that includes a test plan and procedure, any testing tool and test cases, and their expected results.

### 6.2 Levels of Testing

#### Unit Testing

Unit testing is a level of software testing where individual units/ components of the 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. Unit testing is commonly automated, but may still be performed manually. The objective of unit testing is to isolate a unit and validate its correctness.

A manual approach to unit testing may employ a step-by-step instructional document. Unit testing is the process of testing the part of the program to verify whether the program is working correctly not. In this part, the main intention is to check each and every input which we are inserting into our file.

Here the validation concepts are used to check whether the program is taking the inputs in the correct format or not. Unit testing may reduce uncertainty in the units themselves.

## **Integration Testing**

Integration testing is also taken as integration and testing this is the major testing process where the units are combined and tested. Its main objective is to verify whether the major parts of the program is working fine or not.

This testing can be done by choosing the options in the program and by giving suitable inputs.

## **System Testing**

System testing is defined as the 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 pre-requisite and is done by the testing team.

System testing is done after integration testing is complete. System testing should test the functional and non-functional requirements of the software.

## **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 Functions in a manner that can be reasonably by the customer.

1. Validation test criteria
2. Configuration review
3. Alpha and Beta testing (conducted by end-user)

## **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.

## 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.

User Acceptance Testing is done in the final phase of testing. Here an image is selected as input and then the corresponding output is given.

Serial no	Description	User input	Expected Output	Actual Output
1.	select an image	image selected	its description	description with image
2.	type in search bar	text typed	result found	text related result found

Table 6.1. Testing table

As shown in table 6.1 we test by selecting an image and searching in search bar. When we select an image it should show the description and it shows.

When we search some plant-related items it should display related results and it displays so it qualifies our test.

## CHAPTER 7

# RESULTS

### 7.1 Home page

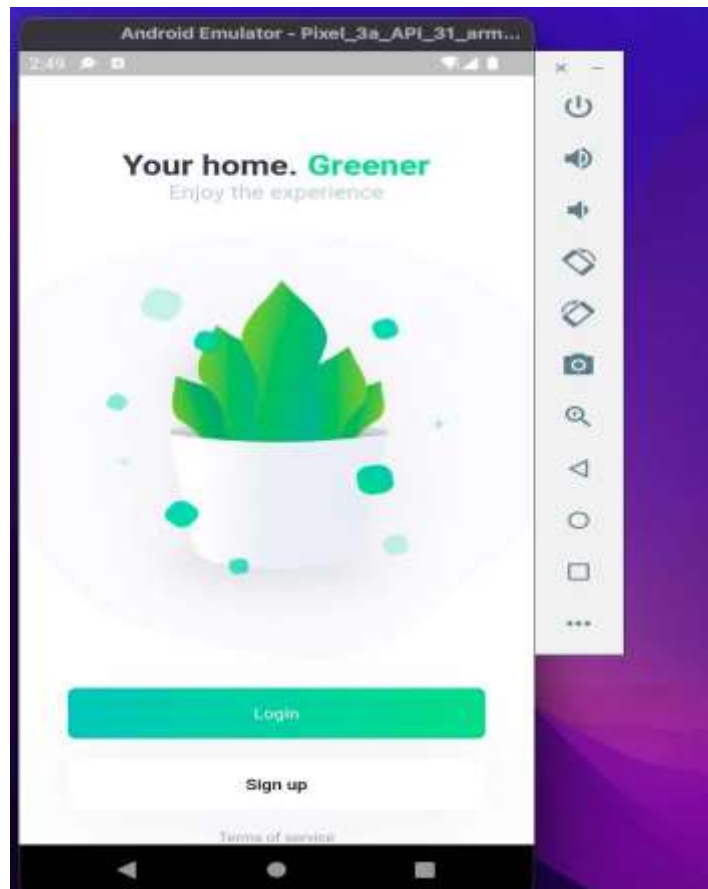


Figure 7.1: Home page

This is the landing page of the application as shown in figure 7.1 where we can enter the details of the user.

Then it directs to the main plant app. Here we have a text box written with “ Your home. Greener, Enjoy the experience”. And then one plant image is there followed by login and signup button.



## 7.2 Display Details

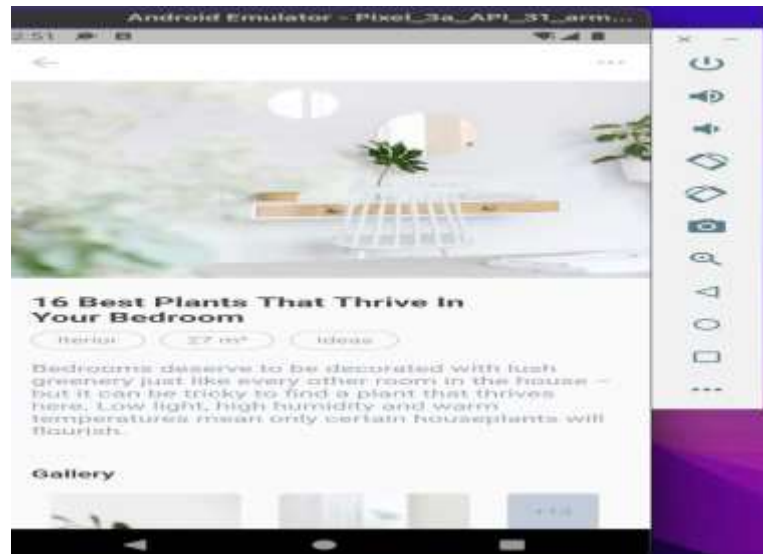


Figure 7.2: Display details

This is the details page of the app as shown in figure 7.2. There will be a number of different varieties of plants and seeds etc. Users will get to explore all the different varieties possible. Users can select a particular plant and get the details for the same.

## 7.3 Browse Page

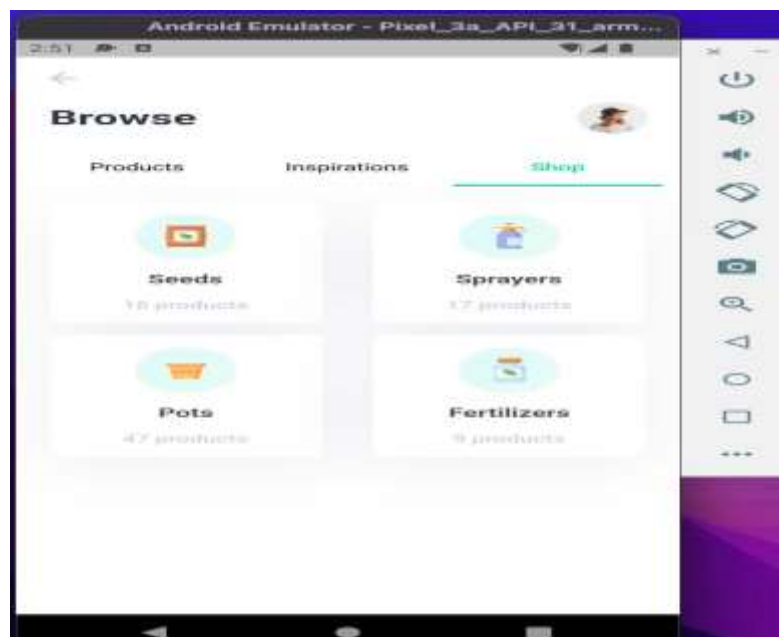


Figure 7.3: Browse page

Figure 7.1 is the explore page. On this page, we will see all the different sections in of our app. Each leads a different page of seeds, sprayers, pots and fertilizers.







## **CHAPTER 8**

# **CONCLUSION AND FUTURE ENHANCEMENTS**

### **Conclusion**

This plant app allows users to know about different plant varieties along with images. Along with that we also have flowers, seeds, sprayers, pots, and fertilizers along with their details and images. Users just need to login and at one place they can get all updates of new varieties then they can think of buying it for themselves. It can simply be used to boost their general knowledge about plants and different species.

### **Future Enhancements**

At the same time, there is some scope for improvement in the future. It can be possible to make it more users friendly by adding more variety of functions to it.

We can add backend and server. With that, all different users will have their details saved.

We can host our app on servers like AWS or Heroku. We can add purchasing functionality for all products available in the app.

We can do the payment gateway integration with payment application such as Phonepe, Paytm, Gpay etc.

# REFERENCES

- [1] Microsoft Visual Studio Step by Step (Developer Reference Edition)
- [2] RamezElmasri and Shamkant B. Navathe, Fundamentals of UI/UX design
- [3] Learn Flutter Tutorial - javatpoint
- [4] MS SQL Server Tutorial (tutorialspoint.com)
- [5] Flutter Tutorial - GeeksforGeeks
- [6] What is Dart Language? Explain Architecture & Components (guru99.com)