CMR UNIVERSITY

Private University Established in Karnataka State by Act No. 45 of 2013

SCHOOL OF ENGINEERING AND TECHNOLOGY

Project Work

On

"Sweet Bliss"

For the requirement of 6th Semester (4BCS603 – Mobile Application Development)

B.Tech. in Computer Science and Engineering

Submitted By

Name		Reg No
ANSHU		(18BTCS014)
DEEPIKA V		(18BTCS029)
PRAJWAL SAI		(18BTCS046)

Submitted to

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR University

Off Hennur - Bagalur Main Road,
Near Kempegowda International Airport,
Chagalahatti, Bengaluru, Karnataka-562149

Academic Year - 2020-21

CMR UNIVERSITY
Private University Established in Karnataka State by Act No. 45 of 2013

SCHOOL OF ENGINEERING AND TECHNOLOGY

Chagalahatti, Bengaluru, Karnataka-562149

Department of Computer Science and Engineering

CERTIFICATE

Certified that the Project Work entitled "**Sweet Bliss"** carried out by **ANSHU (18BTCS014), DEEPIKA V(18BTCS029), PRAJWAL SAI(18BTCS046),** bonafide students of **SCHOOL OF ENGINEERING AND TECHNOLGY**, in partial fulfillment for the award of **BACHELOR OF TECHNOLOGY** in 6th Semester Computer Science and Engineering of **CMR UNIVERSITY**, Bengaluru during the year 2021. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the report. The project has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Signature of Course In-charge	Signature of HOD	Signature of Dean
……	……	……
Dept. of CSE SoET, CMRU, Bangalore	Dept. of CSE SoET, CMRU, Bangalore	SoET, CMRU, Bangalore

Name of the Examiners:	Signature with Date:
1. ……	……
2. ……	……

DECLARATION

I, **ANSHU (18BBTCS014)**,student of Bachelor of Technology, Computer Science and Engineering, CMR University, Bengaluru, hereby declare that the Project Work entitled "Sweet Bliss" submitted by me, for the award of the Bachelor's degree in Computer Science and Engineering to CMR University is a record of bonafide work carried out independently by me under the supervision and guidance of Dr. T. Parameswaran, Associate Professor, Dept of CSE. CMR University.

I further declare that the work reported in this mini project work has not been submitted and will not be submitted, either in part or in full, for the award of any other degree in this university or any other institute or University.

Place: Bengaluru (ANSHU)

Date: (18BBTCS14)

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of this project would be incomplete without the mention of the people who made it possible, without whose constant guidance and encouragement would have made efforts go in vain.

I express my sincere gratitude to my project guide **Dr. T. Parameswaran, Associate Professor, Dept of CSE**, without whose constant guidance and support the project would not be successful.

I also express my sincere thanks to my project co guide **Dr. R. Sathyaraj, Assistant Professor, Dept of CSE,** whose constant guidance and support for the successful completion of the project.

I would like to express my thanks to **Dr. Rubini P, Associate Professor and Head**, Department of Computer Science and Engineering, School of Engineering and Technology, CMR University, Bangalore, for his encouragement that motivated me for the successful completion of Project work.

I express my heartfelt sincere gratitude to **Dr. C. Prabhakar Reddy, Dean,** School of Engineering and Technology, CMR University for his support.

ANSHU (18BBTCS014)

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

The main aim of our project is to develop a Milkshake ordering App using Flutter Framework which adds on value to one's user experience as it's a perfect place to grab a Milkshake when in need by quick ordering and avoiding the chaos and the Queue for placing an order in the store/outlet.

This allows a user to experience an effortless and seamless experience with a range of variety to choose from. The power is in your hands: to customize your milkshake according to your mood and taste requirements. Sweet Bliss the Milkshake ordering App is an impeccable app for various milkshake outlets up in the market which provides the user a great experience.

CHAPTER 1

INTRODUCTON

1.1 Introduction

We all know what milkshakes are. They come in an array of shapes, sizes, colors, flavor's and most of the time give you brain freeze. This sweet and cold beverage is enjoyed by people around the globe, whether it's children, adults or even grandparents.

Milkshakes originated in the United States around the turn of the 20th century, and grew in popularity following the introduction of electric blenders in the following two decades. They became a common part of youth popular culture, as ice cream shops were a culturally acceptable meeting place for youth, and milkshakes became symbolic of the innocence of youth.

The milkshake has now grown into what you will probably drink today at your local store, restaurant or diner. Over the past few year's things have got even bigger and better as now we have completely dedicated outlets out in the market that serve milkshakes and makes one’s heart filled with complete joy.

A smart application embedded in any hand-held device, which will act as a link between the store and the customer to order a milkshake from their comfort zone and simultaneously customizing their own variety is a true delight to a customer and receiving an experience like never before, all of this desired experience is perfectly crafted using the beautiful user interface and designing with Flutter.

Flutter App Development Powered by Google, Flutter is an open-source UI framework to build native-like, flexible, and graphically-enhanced cross-platform apps for web, mobile, and desktop using a single codebase. Flutter is one of the fastest-growing UI frameworks to create expressive and flexible designs. Flutter offers fully-customizable widgets, native performance, and faster rendering, making app development incredibly faster and efficient.

1.2 Problem Statement

Designing a Milkshake ordering app using Flutter which provides a fully interactive user experience, driving customer engagement. On mobile devices, customers can choose and customize their own Milkshake according to their specific needs.

1.3 Objectives

The main objective of this application is an efficient management of details of shakes, cart, category, order, customer. It manages all the information securely and the project is totally built on the administrator's end .

- Promote and Upsell Your New Items : The best option to go for to sell and promote your newly introduced items on the menu. This helps in getting company to achieve what they are looking for.

- Convenient and Easy to Use : It’s quick and easy to update the digital menu. Upload images, add description and schedule the items as per session. With single click all the devices gets updated in no time and the order get confirmed. It works offline as well. More information shared means more purchases which will indirectly increase your profit margins.

- Reduce Wait Time : This application helps waiting customers to decide on what to order and instantly place their order once they are seated. Moreover company can use several other things like display fun facts or display promotional material to distract the waiting customers and get them less frustrated

- Better Visibility of Menu: The digital display have the ability to grab user’s attention with colorful UI and animations. On any given day, the digital boards are more eye pleasing than the regular paper menus.

CHAPTER 2

LITERATURE SURVEY

This survey is done to comprehend the need and prerequisite of the general population, and to do as such, we went through different sites and applications and looked for the fundamental data. Based on these data, we made an audit that helped us get new thoughts and make different arrangements for our task. We reached the decision that there is a need of such application and felt that there is a decent extent of progress in this field too.

This survey is done to comprehend the need and prerequisite of the general population, and to do as such, we went through different sites and applications and looked for the fundamental data. Based on these data, we made an audit that helped us get new thoughts and make different arrangements for our task. We reached the decision that there is a need of such application and felt that there is a decent extent of progress in this field too.

Menu being the first point of contact for a customer to order food , milkshakes ,etc, it needs to make an impression. In a outlet store, customers do not read 'the menu' but quickly scan them. While we follow the tradition of having paper menus and spend time, money and resources on it to get an appealing design and effect on the customer, after a couple of months the same menu is often unrecognizable.

Again, if a customer prefers to speak with the wait staff to decide on order rather than deciding from menu, outlet store needs to improve menu as early as possible.
As per recent survey, 86% of customers get confused what to order inside outlet store due to lack of information available in menu. With Paper menus, it is not possible to update instantly with your latest offerings or unavailable items and customers gets frustrated due to the long wait time at the outlets.

A tech revolution is on its way for restaurants and outlets. Be it online reservations, Mobile apps to place orders and make payment or providing feedback with a few taps on phone, there is a tech solution available for every need of a customer and restaurant/ outlets.
It's quick and easy to update the digital menu. Upload images, add description and schedule the items as per session. With single click all the devices gets updated in no time. It works offline as well. This modern application is a provider of innovative end-to-end solutions for customers.

CHAPTER 3

REQUIREMENTS

3.1 Software Requirements

3.1.1 **Flutter SDK :**

Flutter is an open-source UI software development kit. In addition to a react-style framework, this toolkit includes an own rendering engine, a rich set of ready-made widgets, unit and integration testing APIs, plugin APIs, and command-line tools for building and compiling apps.

3.1.2 **Dart :**

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. It supports interfaces, mixins, abstract classes, reified generics, and type inference.

3.1.3 **Android Studio :**

Android Studio is a new Android development environment based on IntelliJ IDEA. It provides new features and improvements over Eclipse ADT and will be the official Android IDE once it's ready. Android Studio is the official Integrated Development Environment (IDE) for android application development. Android Studio provides more features that enhance our productivity while building Android apps.

3.1.4 **Operating System :**

An Operating System (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers. Five of the most common operating systems are Microsoft Windows, Apple macOS, Linux, Android and Apple's iOS. But we use windows as an operating system in building this project.

3.1.5 **Tools:**

Flutter depends on these tools being available in your environment.

Windows PowerShell 5.0 or newer (this is pre-installed with Windows 10)

Git for Windows 2.x, with the Use Git from the Windows Command Prompt option. If Git for Windows is already installed, make sure you can run git commands from the command prompt or PowerShell.

3.2 HARDWARE REQUIREMENTS

3.2.1 Laptop

3.2.2 USB cable

3.2.3 Android Smartphone

3.2.4 Internet Connection

CHAPTER 4

SYSTEM ANALYSIS

In Android, the View is the foundation of everything that shows up on the screen. Buttons, toolbars, and inputs, everything is a View. In Flutter, the rough equivalent to a View is a Widget. Widgets don't map exactly to Android views, but while you're getting acquainted with how Flutter works you can think of them as "the way you declare and construct UI". However, these have a few differences to a View.

To start, widgets have a different lifespan: they are immutable and only exist until they need to be changed. Whenever widgets or their state change, Flutter's framework creates a new tree of widget instances. In comparison, an Android view is drawn once and does not redraw until invalidate is called.

Flutter's widgets are lightweight, in part due to their immutability. Because they aren't views themselves, and aren't directly drawing anything, but rather are a description of the UI and its semantics that get "inflated" into actual view objects under the hood.

Flutter includes the Material Components library. These are widgets that implement the Material Design guidelines. Material Design is a flexible design system optimized for all platforms, including iOS.

But Flutter is flexible and expressive enough to implement any design language. For example, on iOS, you can use the Cupertino widgets to produce an interface that looks like Apple's iOS design language.

4.1 Introduction to widgets:

Flutter widgets are built using a modern framework that takes inspiration from React. The central idea is that you build your 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 diffs 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.

The `runApp()` function takes the given `Widget` and makes it the root of the widget tree.

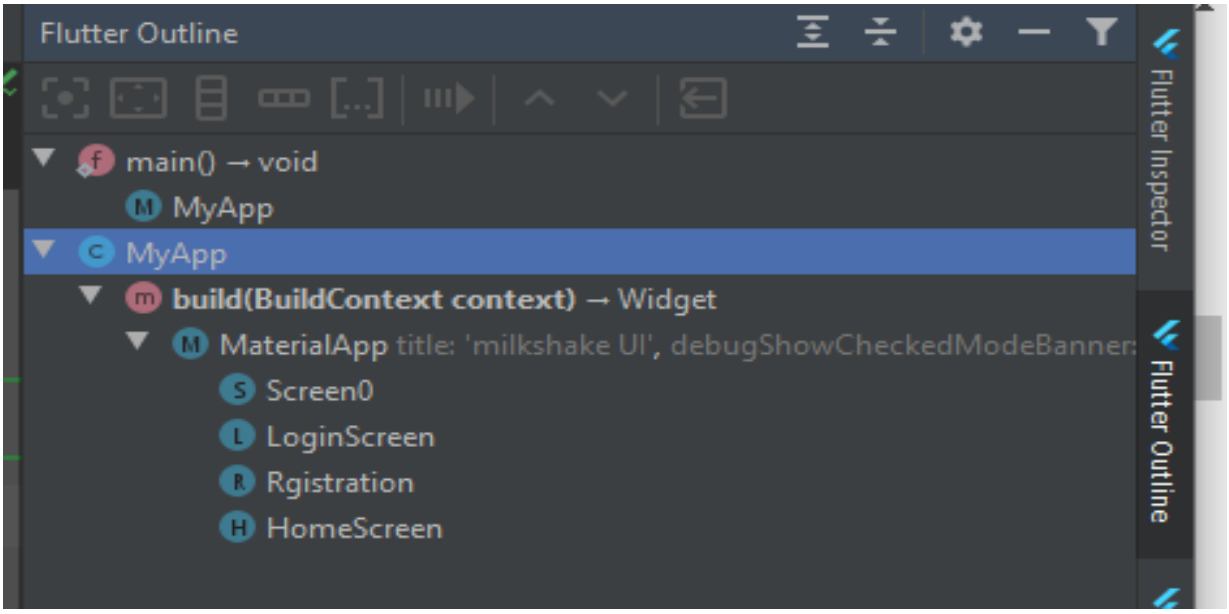


Figure 1

When writing an app, you'll commonly author new widgets that are subclasses of either `StatelessWidget` or `StatefulWidget`, depending on whether your widget manages any state. A widget's main job is to implement a `build()` function, which describes the widget in terms of other, lower-level widgets. The framework builds those widgets in turn until the process bottoms out in widgets that represent the underlying `RenderObject`, which computes and describes the geometry of the widget.

4.2 Stateless Widget class :

A widget that does not require mutable state.

A 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 building process continues recursively until the description of the user interface is fully concrete (e.g., consists entirely of RenderObjectWidgets, which describe concrete RenderObjects).

Stateless widget are useful when the part of the user interface you are describing does not depend on anything other than the configuration information in the object itself and the BuildContext in which the widget is inflated. For compositions that can change dynamically, e.g. due to having an internal clock-driven state, or depending on some system state, consider using StatefulWidget

4.3 Stateful Widget class :

A widget that has mutable state.

State is information that (1) can be read synchronously when the widget is built and (2) might change during the lifetime of the widget. It is the responsibility of the widget implementer to ensure that the

State is promptly notified when such state changes, using State.setState.

A stateful 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 building process continues recursively until the description of the user interface is fully concrete (e.g., consists entirely RenderObjectWidgets, which describe concrete RenderObjects).

Stateful widgets are useful when the part of the user interface you are describing can change dynamically, e.g. due to having an internal clock-driven state, or depending on some system state. For compositions that depend only on the configuration information in the object itself and the BuildContext in which the widget is inflated, consider using StatelessWidget.

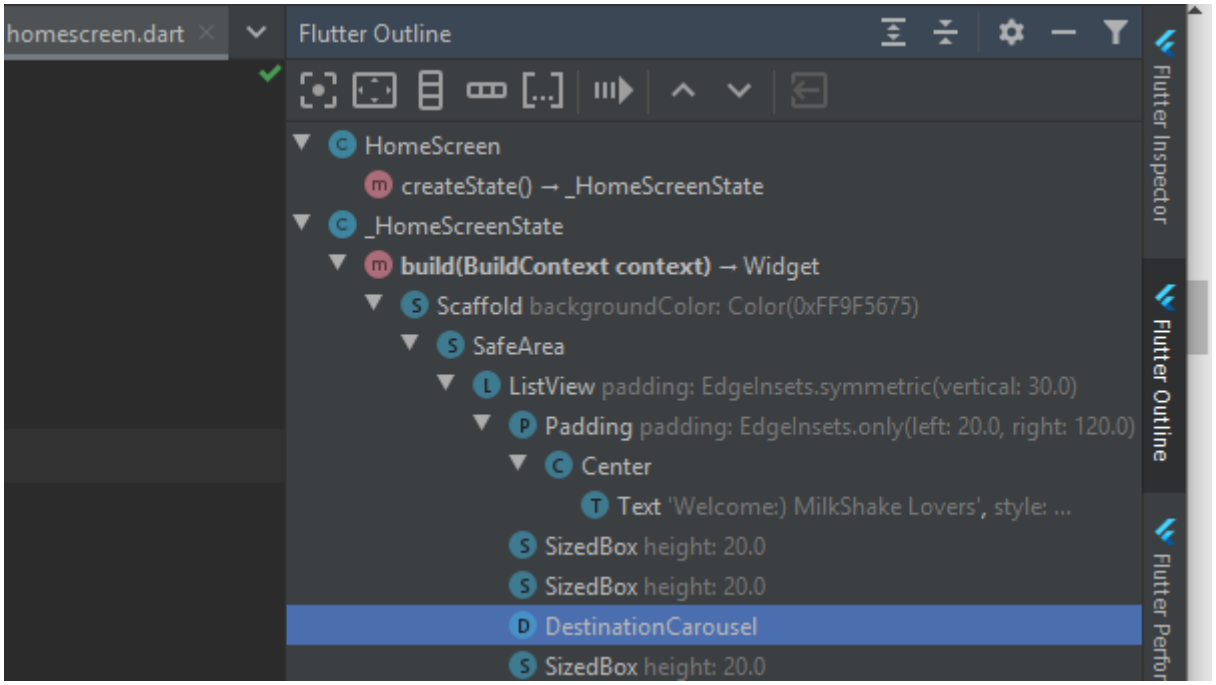


Figure 2

Flutter uses Stateful Widgets to capture this idea. Stateful Widgets are special widgets that know how to generate State objects, which are then used to hold state. In Flutter, these two types of objects have different life cycles. Widgets are temporary objects, used to construct a presentation of the application in its current state. State objects, on the other hand, are persistent between calls to build(), allowing

them to remember information.

4.4 Basic widgets:

Flutter comes with a suite of powerful basic widgets, of which the following are commonly used:

Text: The Text widget lets you create a run of styled text within your application.

Row, Column: These flex widgets let you create flexible layouts in both the horizontal (Row) and vertical (Column) directions. The design of these objects is based on the web's flexbox layout model.

Stack: Instead of being linearly oriented (either horizontally or vertically), a Stack widget lets you place widgets on top of each other in paint order. You can then use the Positioned widget on children of a Stack to position them relative to the top, right, bottom, or left edge of the stack. Stacks are based on the web's absolute positioning layout model.

Container : The `Container` widget lets you create a rectangular visual element. A container can be decorated with a `BoxDecoration`, such as a background, a border, or a shadow. A `Container` can also have margins, padding, and constraints applied to its size. In addition, a `Container` can be transformed in three dimensional space using a matrix.

4.5 Layouts in Flutter :

Widgets are classes used to build UIs. Widgets are used for both layout and UI elements.
Compose simple widgets to build complex widgets. The core of Flutter's layout mechanism is widgets. In Flutter, almost everything is a widget—even layout models are widgets. The images, icons, and text that you see in a Flutter app are all widgets. But things you don't see are also widgets, such as the rows, columns, and grids that arrange, constrain, and align the visible widgets. You create a layout by composing widgets to build more complex widgets.
Here's a diagram of the widget tree for this UI:

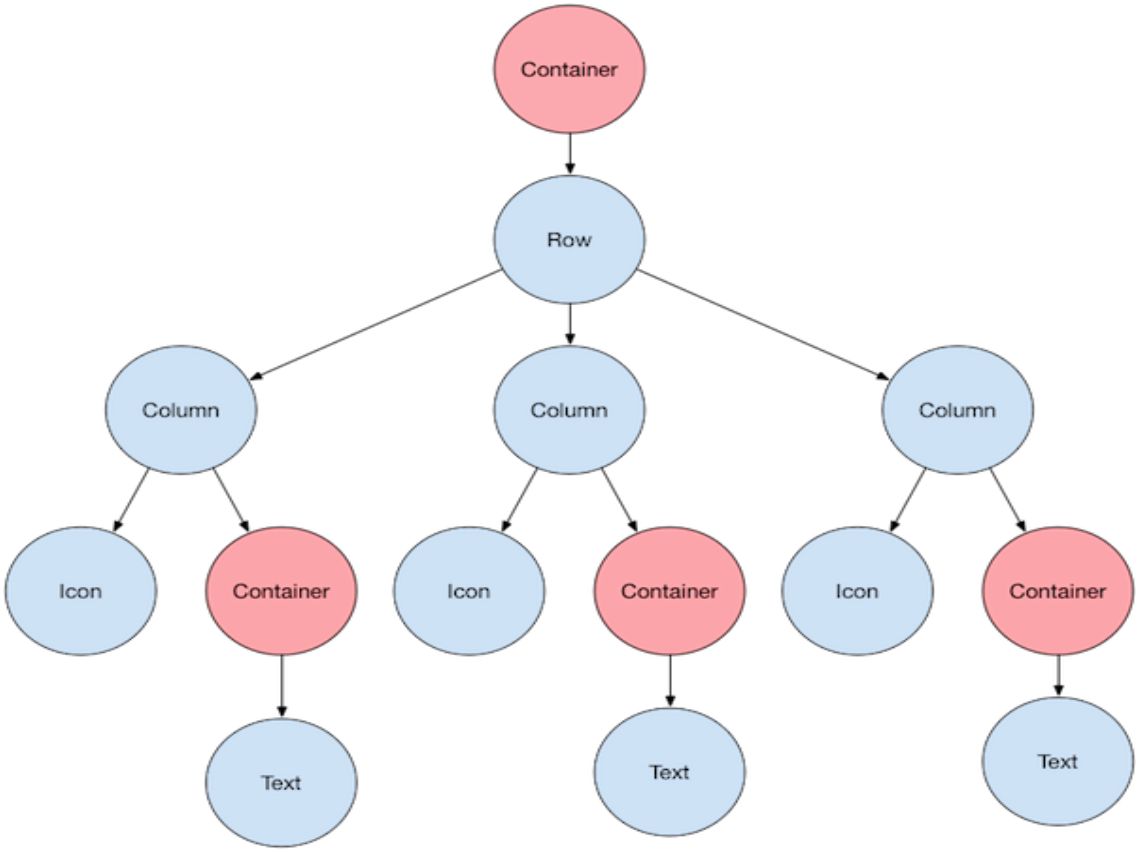


Figure 3

`Container` is a widget class that allows you to customize its child widget. Use a `Container` when you want to add padding, margins, borders, or background color, to name some of its capabilities.
In this example, each `Text` widget is placed in a `Container` to add margins. The entire `Row` is also placed in a `Container` to add padding around the row.

4.6 Adding assets and images :

Flutter apps can include both code and *assets* (sometimes called resources). An asset is a file that is bundled and deployed with your app, and is accessible at runtime. Common types of assets include static data (for example, JSON files), configuration files, icons, and images (JPEG, Web, GIF, animated Web/GIF, PNG, BMP, and WBMP).

4.7 Specifying assets :

Flutter uses the pubspec.yaml file, located at the root of your project, to identify assets required by an app.

Here is an example:

flutter: assets: - assets/my_icon.png - assets/background.png

4.8 The pub spec file :

Every pub package needs some metadata so it can specify its dependencies. Pub packages that are shared with others also need to provide some other information so users can discover them. All of this metadata goes in the package's pub spec: a file named `pubspec.yaml` that's written in the YAML language.

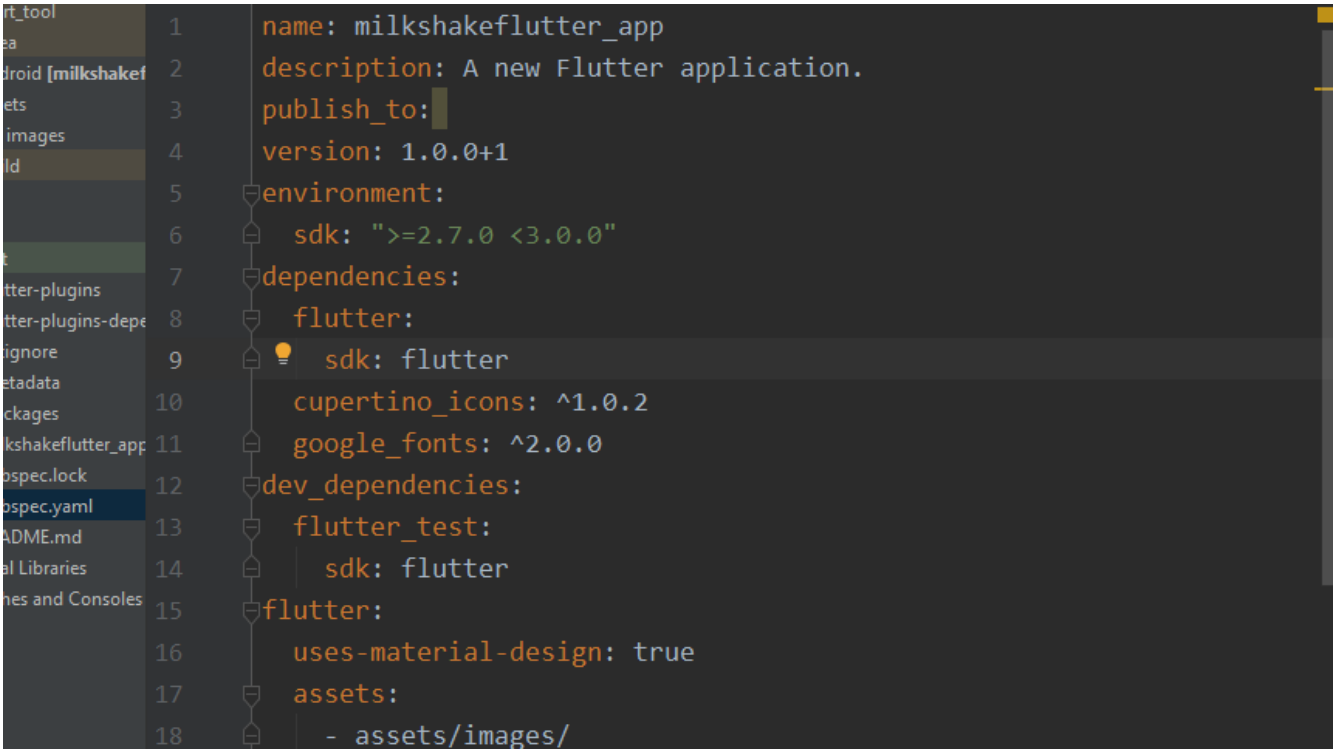


Figure 4

4.9 Navigation and routing :

In Flutter, *screens* and *pages* are called *routes*. The remainder of this recipe refers to routes.

In Android, a route is equivalent to an Activity. In iOS, a route is equivalent to a ViewController. In

Flutter, a route is just a widget.

This recipe uses the `Navigator` to navigate to a new route.

The next few sections show how to navigate between two routes, using these steps:

Create two routes.

Navigate to the second route using Navigator.push().

Return to the first route using Navigator.pop().

Using packages:

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. What is the difference between a package and a plugin? A plugin is a *type* of package—the full designation is *plugin package*, which is generally shortened to *plugin*.

Packages

At a minimum, a Dart package is a directory containing a pubspec file. Additionally, a package can contain dependencies (listed in the pubspec), Dart libraries, apps, resources, tests, images, and examples. The pub.dev site lists many packages—developed by Google engineers and generous members of the Flutter and Dart community— that you can use in your app.

Plugins

A plugin package is a special kind of package that makes platform functionality available to the app. Plugin packages can be written for Android (using Kotlin or Java), iOS (using Swift or Objective-C), web, macOS, Windows, Linux, or any combination thereof. For example, a plugin might provide Flutter apps with the ability to use a device's camera.

Existing packages enable many use cases—for example, making network requests (`http`), custom navigation/route handling (`fluro`), integration with device APIs (`url_launcher` and `battery`), and using third-party platform SDKs like Firebase (FlutterFire).

To write a new package, see developing packages. To add assets, images or fonts, whether stored in files or packages, see Adding assets and images.

CHAPTER 5

SYSTEM DESIGN

5.1 Case Diagram :

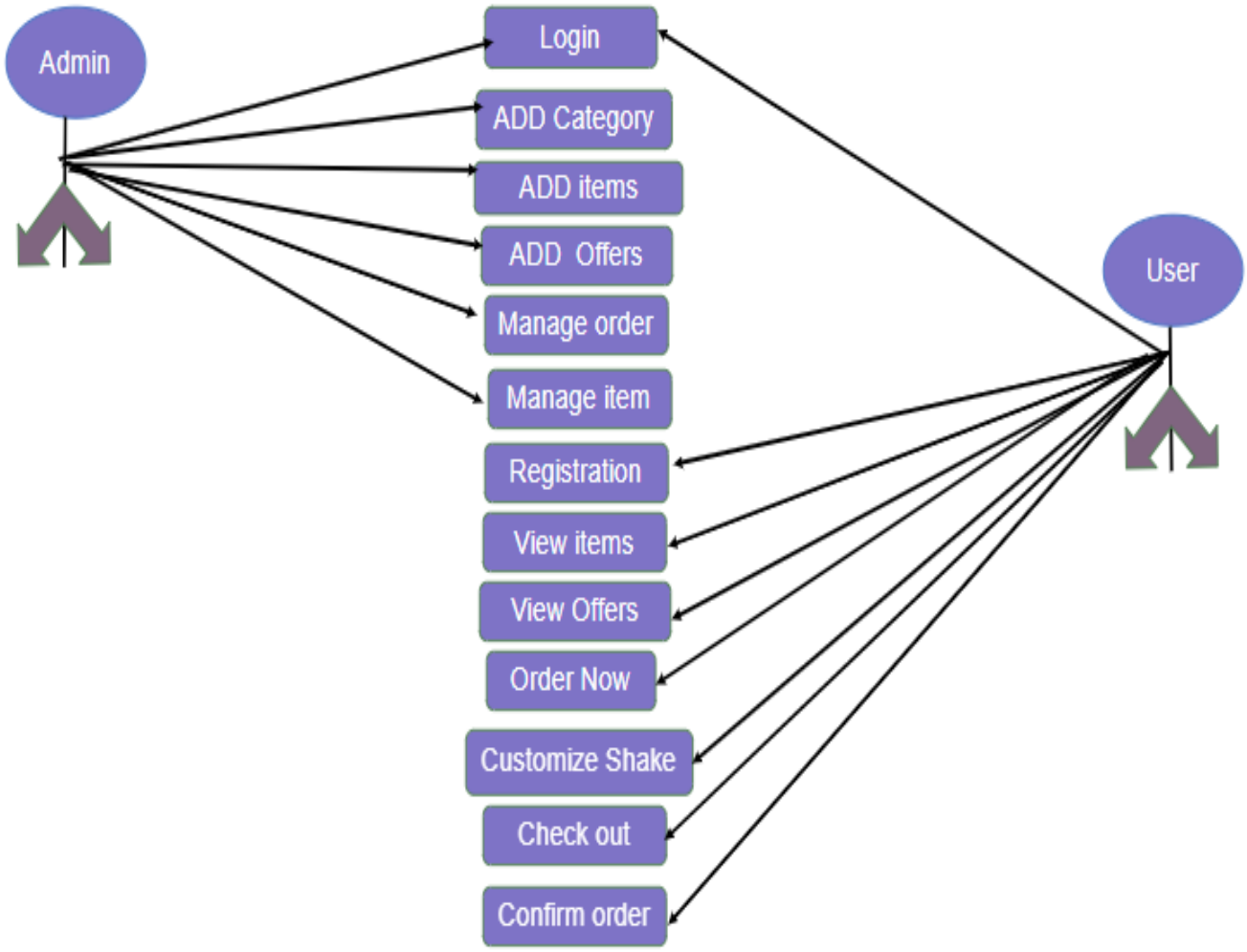


Figure 5

5.2 User Diagram:

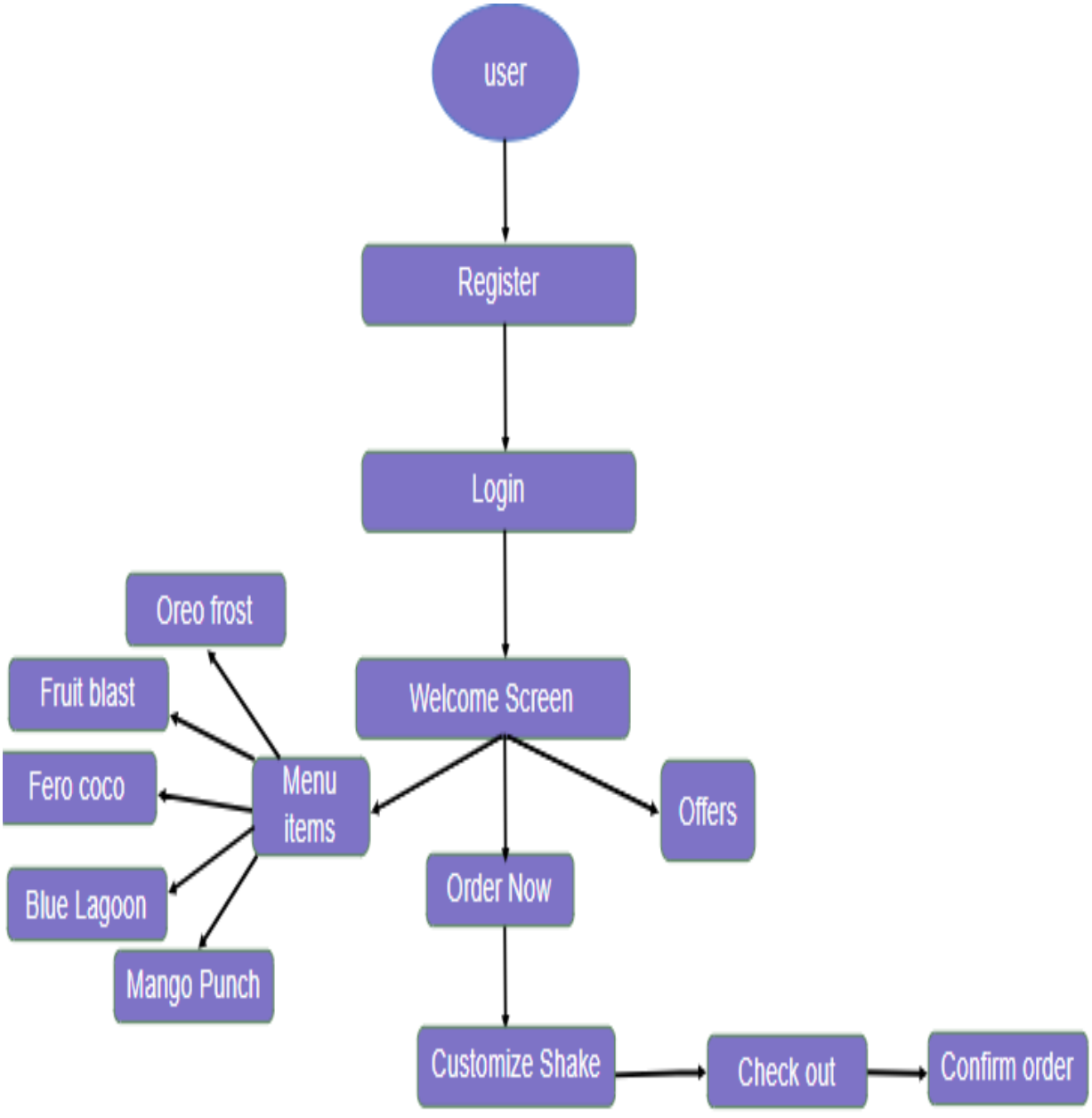


Figure 6

CHAPTER 6

ALGORITHM AND IMPLEMENTATION

6.1 **Algorithm steps:**

STEP 1: START

STEP 2: Register the User

STEP 3: Login the User

STEP 4: The user interacts with the App User Interface.

STEP 5: User can view the menu i.e., Taste of the Week.

STEP 6: User can view the offers

STEP 7: User can order the milkshake by pressing the “Order Now” button.

STEP 8: User can customize his/her Milkshake and add on Toppings by just clicking on the items provided in the list.

STEP 9: User can switch to check out screen and view the total and items that he added.

STEP 10: User can click on confirm order.

STEP 11: END.

6.2 **Implementation:**

We have totally 16 dart files in our project which are connected and a part of various screens present.
We are using two packages namely-
1)Cupertino icons
2)Google Fonts
As the user lands on the Welcome screen he/she can view the menu.
The menu pictures have an onClick()function which takes the user to a new page where the user can view more details like the ingredients in the Milkshake, preparation time, Starts/Ratings.
The user can click on order now and go ahead with personalization of his/her Milkshake and add on toppings by selecting and clicking as this will directly add all the items that are selected into to checkout list where the total is done.
And the user can view the total and confirm his order.

Codes :

- Homescreen.dart File :

```
import 'package:flutter/material.dart';
import 'milk_carousel.dart';
import 'offer_carouse.dart';

import 'package:google_fonts/google_fonts.dart';

class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Color(0xFF9F5675),
      body: SafeArea(
        child: ListView(
          padding: EdgeInsets.symmetric(vertical: 30.0),
          children: <Widget>[
            Padding(
              padding: EdgeInsets.only(left: 20.0, right: 120.0),
              child: Center(
                child: Text(
                  'Welcome:) MilkShake Lovers',
                  style: GoogleFonts.merienda(
                    textStyle: Theme.of(context).textTheme.headline2,
                    fontSize: 30,
                    fontWeight: FontWeight.w700,
                    fontStyle: FontStyle.italic,
                    color: Colors.black,
```

```
              ),
            ),
          ),
        ),
        SizedBox(height: 20.0),
        SizedBox(height: 20.0),
        DestinationCarousel(),
        SizedBox(height: 20.0),
        HotelCarousel(),
      ],
    ),
  ),
 );
 }
}
```

- main.dart file :

```
import 'package:flutter/material.dart';
import 'package:milkshakeflutter_app/homescreen.dart';
import 'login.dart';
import 'register.dart';

import 'welcome.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
 // This widget is the root of your application.
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
    title: 'milkshake UI',
```

```
      debugShowCheckedModeBanner: false,


  initialRoute: '/',
      routes: {
        '/': (context) => Screen0(),
        '/login': (context) => LoginScreen(),
        '/Second': (context) => Rgistration(),
        '/home': (context) => HomeScreen(),
      });
  }}
```

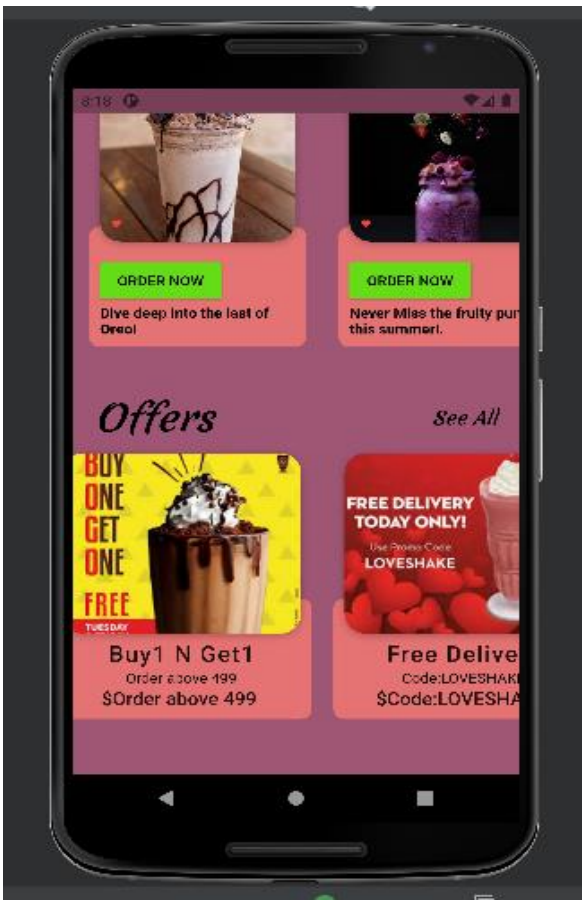
CHAPTER 7

RESULT AND SCREENSHOTS :

Figure 7


Enter your email
Enter your password.
Log In


Welcome:)
MilkShake Lovers
Taste Of The Week
See All
ORDER NOW
Dive deep into the last of Oreo!
ORDER NOW
Never Miss the fruity pur
this summer!.
Offers
See All
BUY ONE GET
FREE DELIVERY TODAY ONLY!


ORDER NOW
Dive deep into the last of Oreo!
ORDER NOW
Never Miss the fruity pur
this summer!.
Offers
See All
BUY ONE GET ONE FREE
TUESDAY
FREE DELIVERY TODAY ONLY!
Use Promo Code
LOVESHAKE
Buy1 N Get1
Order above 499
$Order above 499
Free Delive
Code:LOVESHAK
$Code:LOVESHA


Fruit Blast
Ingridents
₹ 250
Oreo,Milk,NaturalFlavours&
Essence
Ingridents
₹ 210
Strawberry

Figure 8

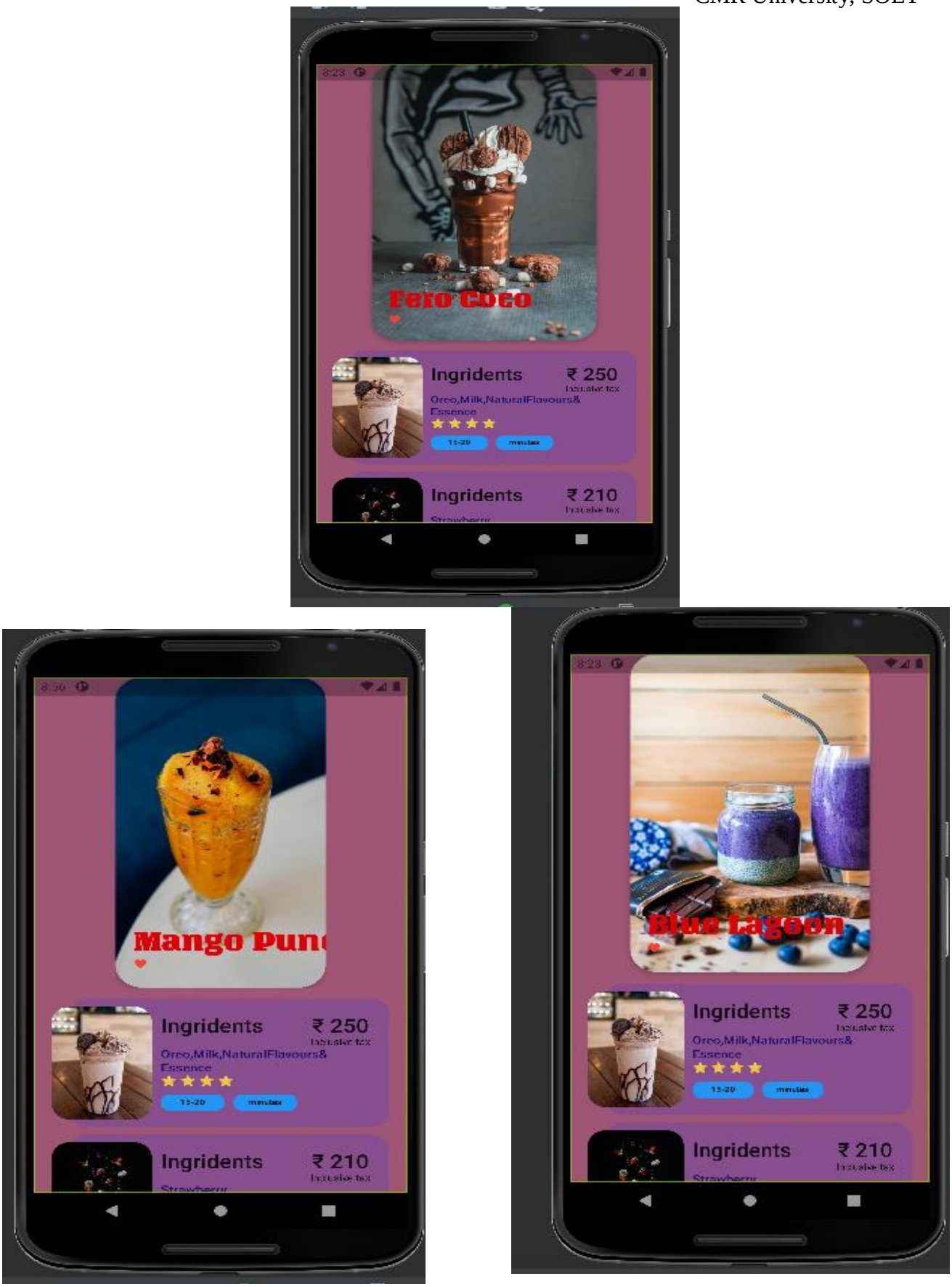
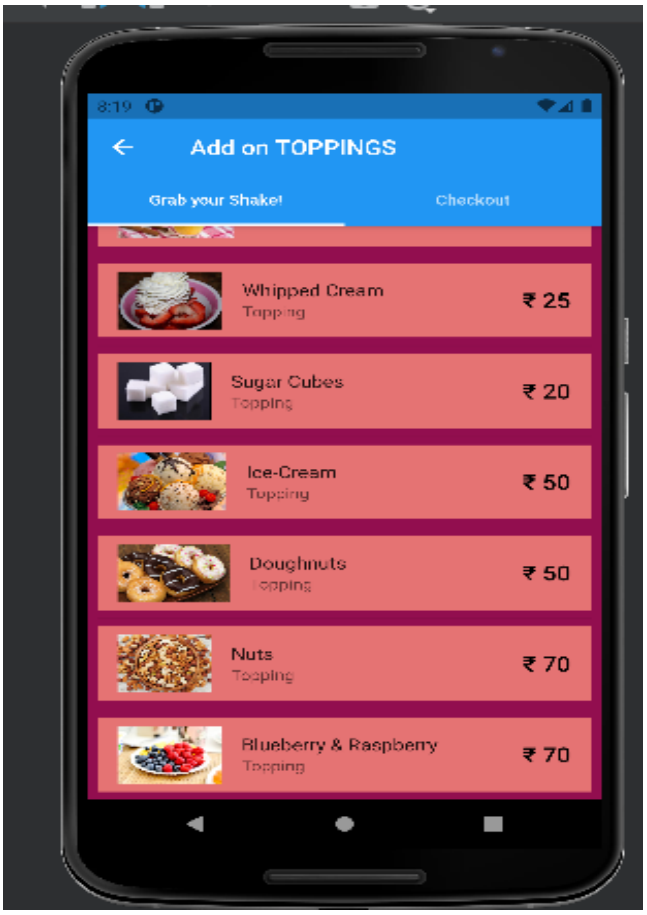

Pero Coco
Ingridents
₹ 250
Oreo,Milk,NaturalFlavours&
Essence
Ingridents
₹ 210
Mango Pun
Ingridents
₹ 250
Oreo,Milk,NaturalFlavours&
Essence
Ingridents
₹ 210
Blue Lagoon
Ingridents
₹ 250
Oreo,Milk,NaturalFlavours&
Essence
Ingridents
₹ 210

Figure 9


Add on TOPPINGS
Grab your Shake!
Checkout
Whipped Cream
Topping
₹ 25
Sugar Cubes
Topping
₹ 20
Ice-Cream
Topping
₹ 50
Doughnuts
Topping
₹ 50
Nuts
Topping
₹ 70
Blueberry & Raspberry
Topping
₹ 70


Add on TOPPINGS
Grab your Shake!
Checkout
Trio Thunder
Milkshake
₹ 200
Strawberry Queen
Milkshake
₹ 215
Fruit Blast
Milkshake
₹ 230
Blue Lagoon
Milkshake
₹ 250
Mango Punch
Milkshake
₹ 200
Whipped Cream
Topping
₹ 25

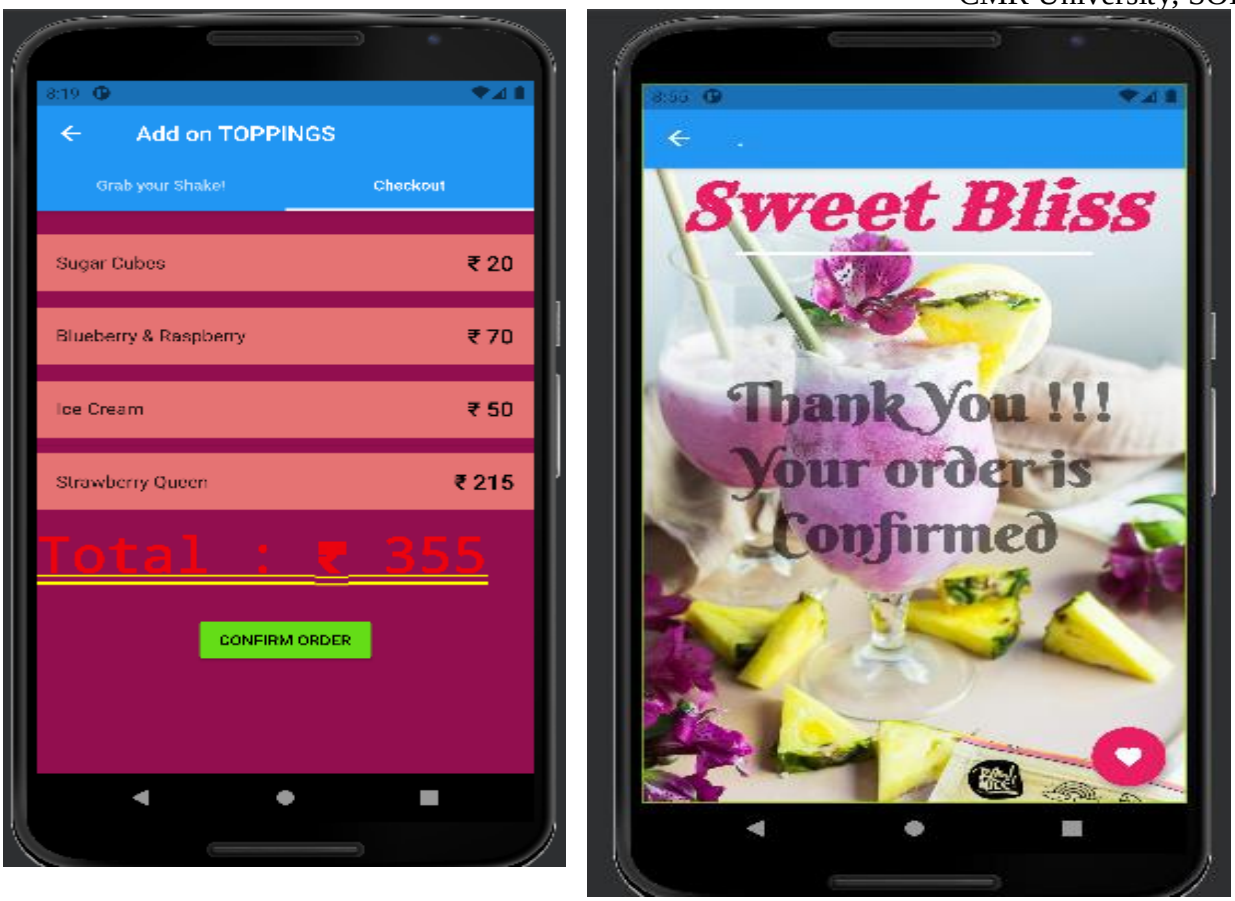


Figure 10

CHAPTER 8

Conclusion

Technology has substantially revolutionized the signage industry. Although we live in a digital world, many restaurants/outlets feel an understandable reluctance when it comes to adopting new technology. It's important to recognize how digital technology drives our society and its major impact on consumer reactions and ordering habits. To avoid falling behind the competition, forward thinking restaurants/outlets are embracing digital signage technology and taking full advantage of its invaluable capabilities to maximize their business profits.
Free from the limitations of static signage, outlets can effectively and visually educate their customers on menu items, pricing structures, and ordering protocols. Leveraging custom content to inform consumers before they reach the register is a proven way to minimize confusion and speed customers through those wait line rushes.

CHAPTER 9

Future Scope

This framework can be stretched out further to have abundant security highlights.
The application can be further improvised by adding different types of UI and convenient user interface features in future. And allow customers to login via social platforms such as Google and Facebook.

Other interaction features could be added like entertainment, which could be linked to social network such as Facebook and Twitter, and also more personalization features like setting up personal account and diet records, saving personal preferences for regular customers.

CHAPTER 10

Reference

[1] https://flutter.dev/

[2] https://pub.dev/

[3] https://material.io/develop/flutter

[4] https://flutterawesome.com/tag/ui/