VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018

An Internship Project Report
on

Flight Ticket Booking Application

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

by

Vaidehi VL
1RN18IS114

Under the Guidance of

Mr. R Rajkumar
Associate Professor
Department of ISE

ESTD:2001
An Institute with a Difference

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

Channasandra, Dr.Vishnuvardhan Road, RR Nagar

Post, Bengaluru – 560 098

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

CERTIFICATE

Certified that the Internship work entitled ***Flight ticket booking*** has been successfully completed by Vaidehi VL(1RN18IS114) student of RNS Institute of Technology, Bengaluru in partial fulfillment of the requirements of 7th 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.

Mr. R Rajkumar	**Dr. Suresh. L**	**Dr. M K Venkatesha**
Internship Guide	Professor and HoD	Principal
Associate Professor	Department of ISE	RNSIT
Department of ISE	RNSIT	

External Viva

Name of the Examiners	**Signature with date**
1.	
2.	

DECLARATION

I, Vaidehi VL [USN: 1RN18IS114] student of VII Semester BE, in Information Science and Engineering, RNS Institute of Technology hereby declare that the Internship work entitled **Flight Booking Application** has been carried out by us and submitted in partial fulfillment of the requirements for the VII Semester degree of **Bachelor of Engineering in Information Science and Engineering** of Visvesvaraya Technological University, Belagavi during academic year 2021-2022.

Place: Bengaluru

Date:10/01/2022

Vaidehi VL (1RN18IS114)

ABSTRACT

The flight ticket booking application provides a user friendly, interactive system to book flight tickets over the internet. The main objective of the application is to allow users to login, book flight tickets and view the status of their current booking.

The project is built using Flutter, which is an open-source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase. The Flutter framework consists of both a software development kit (SDK) and their widget-based UI library. This library consists of various reusable UI elements, such as sliders, buttons, and text inputs.

This flight booking app has a sound UI design and quality features that render a seamless user experience. Flight booking is made simple as that of a few clicks and swipes. The user gets to book their flight tickets through the app anywhere and everywhere in the comfort of their house. The users booking status will be displayed in the booking status section, where the details about confirmation or cancellations of booking are displayed.

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, 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 **Dr. Mamatha G** 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 always helping.

I thank Mr. **Akshay DR, Director, Enmaz Engineering Services Private Limited,** 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 for supporting and encouraging me throughout. I have made an honest effort in this assignment.

VAIDEHI VL

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

ARM	Advanced RISC Machines
BLOC	Business Logic Component
GPU	Graphics Processing Unit
IDE	Integrated Development Environment
iOS	iPhone Operating System
SDK	Software Development Kit
UI	User Interface
VM	Virtual Machine

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Background

The flight ticket booking application provides a user friendly, interactive system to book flight tickets over the internet. The main objective of the application is to allow users to login, book flight tickets and also view the status of their current booking. The user gets to book their flight tickets through the app anywhere and everywhere in the comfort of their house. The users booking status will be displayed in the booking status section, where the details about confirmation or cancellations of booking are displayed.

1.1.1 History

Flutter is an open-source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase. Released in 2017, Flutter allows developers to build mobile applications for both iOS and Android with a single codebase and programming language. This capability makes building iOS and Android apps simpler and faster. The Flutter framework consists of both a software development kit (SDK) and their widget-based UI library. This library consists of various reusable UI elements, such as sliders, buttons, and text inputs.

The first version of Flutter was known by the codename "Sky" and ran on the Android operating system. During the keynote of Google Developer Days in Shanghai in September 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 December 11, 2019, Flutter 1.12 was released at the Flutter Interactive event. On March 3, 2021, Google released Flutter 2 during an online Flutter Engage event. On September 8th, 2021, the Dart SDK in version 2.14 and Flutter version 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. The major components of Flutter include: Dart platform, Flutter engine, Foundation library, Design specific widgets, Flutter Development tools.

1.2 Existing System

The existing system is that the passenger must fill up the data manually and must submit it to the reservation counter. It may take a lot of time to process it and to book the flight. Therefore there is wastage of time. Since the data is entered manually, the probability of error or mistakes is high.

Drawbacks of existing system:

- Entering Record: Entry of each record is done manually each time the record is done manual each time the record is maintained on paper and it maximizes the maintenance of additional files.
- Searching the record: Due to absence of unique identification of a flight, the searching of record takes much time and increases the time wastage.
- Deleting the Record: In the current system the concept of deleting record is tedious.
- Modification of Records: If any modification is required it is done directly on the documents being preserved in correspondence to account information.

1.3 Proposed System

To avoid the limitation of current system it's necessary to design and develop a new system which have the following benefit and the existing system.

1. Everything is automated which reduce the risk factor.
2. Flexibility in generating of information.
3. Quick retrieved and maintenance of data.
4. Highly accurate.
5. User satisfaction.

Advantages of proposed system:

- The proposed system due to computerized is much faster in reservation process, cancellation process and transactions.
- Transfer of information from various branches would become easier and faster.
- Managing and maintaining data becomes easier and cost effective due to very high amount and reliability of storage space available in the proposed system.
- Customer services can not only be satisfied but also enhanced to the extent that one can obtain or cancel a reservation from any given time.

Chapter 2

LITERATURE REVIEW

A Literature review describes the concept of how the concept has emerged, how it has been implemented and what the current status is.

In paper [1], the author discussed the usability of an online airline reservation system cannot be predicted without considering the flexible behavior of travelers. However, travelers' flexible behavior is molded by a number of service quality attributes. In this paper, service quality attributes along with external variables were computed to determine their association with flexible behavior of travelers and to also ascertain their individual range and strength of association. Pearson Correlation Coefficients and Multiple Regression Analysis was computed using the ten service quality attributes along with external variables to determine their association with flexible behavior of travelers. The results showed that all of these correlations are statistically significant. Furthermore, significance level was recorded while investigating the association between service quality attributes of airlines, external variables and flexible behavior of travelers. The finding indicates that service quality attributes of airlines and external variables jointly predict flexible behavior of travelers.

Paper [2] discussed how Integrated Development Environments (IDEs) provide a convenient standalone solution that supports developers during various phases of software development. Integrated Development Environments (IDEs) are very popular among software developers since they provide support for many of their daily development or maintenance tasks. Modern IDEs provide integrated debuggers, automated refactoring, assistance tools like code completion, and even integrated version control. The focus in this paper is on Visual Studio IDE. The author instrumented the previously unexplored Visual Studio IDE and tracked the interactions of developers at an industry partner's software-development department. The captured events are transformed into high-level activities such as development, navigation, IDE configuration, and project management to identify how much time developers spend on each activity.

Paper [3] discussed about cross platform development using Flutter. Cross-platform mobile development today is full of compromise. Developers are forced to choose between either building the same app multiple times for multiple operating systems, or to accept a lowest common denominator solution that trades native speed and accuracy for portability. Flutter is an open-source SDK for creating high-performance, high- fidelity mobile apps for iOS and Android. Few important features of flutter are - Just-in-time compilation is a way of executing computer code that involves compilation during execution of a program – at run time – rather than prior to execution.

Flutter’s hot reload feature helps you quickly and easily experiment, build UIs, add features, and fix bugs. Hot reload works by injecting updated source code files into the running Dart Virtual Machine (VM). After the VM updates classes with the new versions of fields and functions, the Flutter framework automatically rebuilds the widget tree, allowing you to quickly view the effects of your changes.

Paper [4] discussed how Smart phones have introduced great easiness in our daily life by mobile applications. Nowadays, it is possible to complete tasks on-the-go, without the need of computer. One of the facilities provided is the mechanism to buy flight tickets via mobile phones. However, in the development phase, often poor consideration of end-users' usability requirements leads to underutilization of such facility, thus decreasing potential profit of companies. The purpose of this work is to investigate usability problems for mobile flight booking applications on touch-screen phones and suggest solutions. Main expectations of users are presented from HCI (Human Computer Interaction) perspective and discussed through a case study. Questionnaire and interview methods were used for collecting data. Results reveal that the users are very much concerned with the easiness and the lucidness of functions. Usability is a highly considerable subject for users to prefer a mobile flight booking application over booking tickets via online/ or agencies.

Paper [5] discussed the historical development of airlines reservations systems is traced. Characteristics are defined which are those of any real-time, conversational, highly interactive system. The structure of the system is described--agent terminal area, communications facilities, and central site. Lessons learned in the design, development, testing, implementation, and tuning of two generations of systems are discussed. These discussions include initial system design, simulations and systems measurement tools, systems stability and reliability, serial processing, parallel- or multiprocessors, split front-end back end processing, storage hierarchy, standard and special communications disciplines, and flexibility versus performance. Unlike batch processing the real-time system cannot simply take more run time for a job that has been underestimated. It may well not be able to perform the job at all. Design of these systems is as much an art based on experience as a science and is an iterative process. Systems now being installed have capacities of 180 messages/s, accommodate 10 000 terminals which have access to a 10-billion-byte data base, and provide response times of less than 3s.

Chapter 3

ANALYSIS

3.1 Introduction

The flight ticket booking application provides a user friendly, interactive system to book flight tickets over the internet. The main objective of the application is to allow users to login, book flight tickets and view the status of their current booking. The user gets to book their flight tickets through the app anywhere and everywhere in the comfort of their house. The users booking status will be displayed in the booking status section, where the details about confirmation or cancellations of booking are displayed.

3.1.1 Purpose

The purpose of this project is to implement or to design an app for a flight reservation system to check the flight details, book and cancel flight tickets. It makes the process of booking and cancelling flight tickets simple and easy for the passengers

3.1.2 Scope

The Scope of this project is very broad in terms of other manually checking yourself. Few of them are:

- The passenger can get the schedules of every flight available to the destination he wishes to travel on any day.
- Flight booking made as simple as that of a few clicks and swipes. The user gets to book their flight tickets through the app anywhere and everywhere.
- The users booking status will be displayed in the booking status section.

3.2 Description

3.2.1 User Characteristics

The main actors in the system are (1) the user, (2) a flight and (3) a Flight Seat:

- User: Has properties like Name, Address, Age
- Flight: Has properties like Departing/Arriving City, Departure/Arrival dates and times, Flight number.
- Flight Seat: Has properties of identifying seat number, reserved and flight number.

3.2.2 Assumptions and Dependencies

We assume that the users of our application have a minimal knowledge of computer system and should have an availability of Internet. We are dependent on the sources from where we have gathered data and the data are authenticated.

3.3 Requirements

3.3.1 Hardware Requirements

The processor used is Intel(R) Core (TM) i5-8265U CPU @ 1.60GHz. The capacity of Random Access Memory (RAM) is 4GB. The capacity of the storage element of disk space is 2.99GB. The monitor used is HDMI monitor. The keys available in the keyboard are 104 keys.

3.3.2 Software Requirements

Operating System: Windows 10
Tools: Flutter
Platform or IDE: Visual Studio Code
Network: Internet connection required

Chapter 4

System Design

4.1 Flutter Architecture

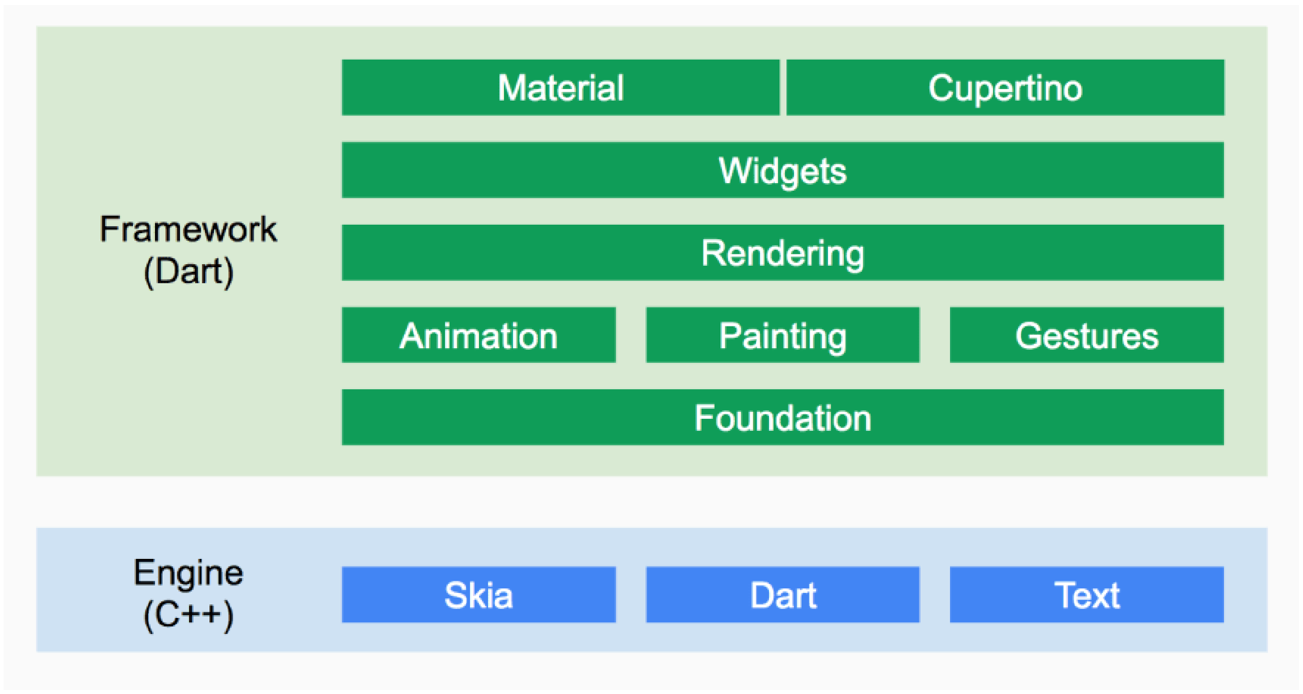

Fig 4.1 Flutter Architecture

Flutter is organized around layers. Each layer is built upon the previous. From the fig 4.1we can see the low-level part of Flutter is an Engine built in C++. It provides low-level rendering support using Google's Skia graphics library. The high-level part of the diagram is the Framework written in Dart. It provides libraries to handle animation, gestures, rendering, widgets and more.
With all this layer the developer can do more with less code by using elements on the top or go down to customize some behavior of its app.

Everything is a widget

In Flutter, everything is a widget nested inside another widget. It comes with beautiful, customizable widgets and we can control the behavior of each widget and styling becomes easy.
All the of a Flutter app form a hierarchy where a widget is a composition of other widgets, and each widget inherits properties from its parent.

4.1.1 Basic widgets

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

• Text - The Text widget displays a string of text with single style. The string might break across multiple lines or might all be displayed on the same line depending on the layout constraints.

• Row - A widget that displays its children in a horizontal array. The design of these objects is based on the web's flex box layout model.

• Column - A widget that displays its children in a vertical array. The design of these objects is based on the web's flex box layout model.

• Image - A widget that displays an image. • Icon - This widget acts as a container for storing the Icon in the Flutter.

• Scaffold - This widget provides a framework that allows you to add common material design elements like App Bar, Floating Action Buttons, Drawers, etc.

• Button - This widget allows you to perform some action on click. Flutter does not allow you to use the Button widget directly; instead, it uses a type of buttons like a Flat Button and a Raised Button.

4.1.2 What is Material design?

Material is an adaptable design system, backed by open-source code that helps developers easily build high-quality, digital experiences. From design guidelines to developer components, Material can help developers build products faster.

• App structure and navigation – Consists of App Bar, Bottom Navigation Bar, Material App, Scaffold, Tab Bar and Tab bar View.

• Buttons –Consists of Button Bar, Dropdown Button, Flat Button, Floating Action Button, Icon Button and Raised Button.

• Input and Selection – Consists of Checkbox, Radio, Slider, Switch.

• Dialogues, alerts and panels – Consists of Alert Dialog, Bottom Sheet, Expansion Panel.

• Information Display – Consists of Card, Icon, Image and Chip.

• Layout – Consists of Divider, List Tile and Stepper.

4.2 Business Logic Component

Flutter delivers the basic architecture that you can apply to your application and manage its state easily. The architecture that is used in Flutter is called the Business Logic Component (BLOC). Basically, it is an event-state based approach that allows you to trigger events and handle state changes based on them. The BLOC is a good approach that separates your business logic from the user interface and oversees business logic key points by testing. The core ideas that were used for BLOC architecture are simplicity, scalability, and testability, and all these goals were achieved within the BLOC architecture.

Chapter 5

DETAILED DESIGN

5.1 Widget tree

The widget tree is how developers create their user interface; developers position widgets within each other to build simple and complex layouts. Since just about everything in the Flutter framework is a widget, and as they start nesting them, the code can become harder to follow.

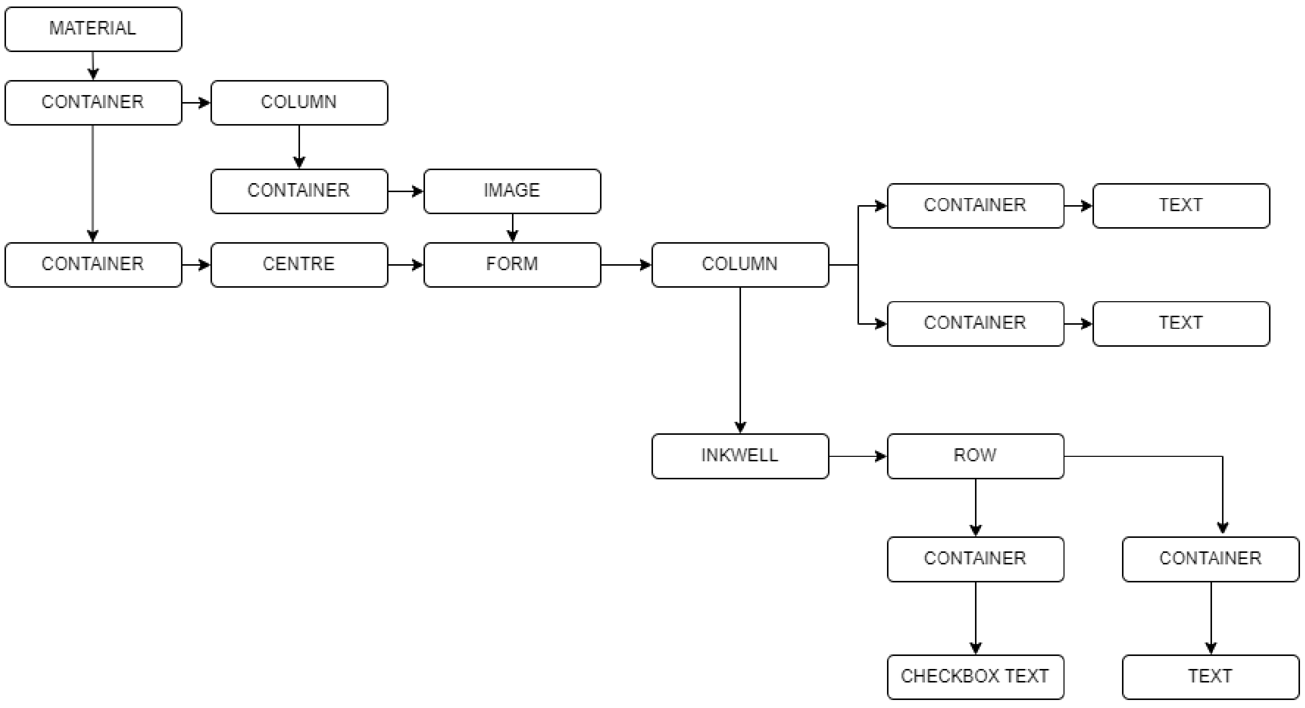

Figure 5.1 Widget tree for Login page

The above figure is an illustration of the widget tree for the login page of the app. It shows all the widgets that the screen contains and a flow of how they are placed. The more the elements/widgets, the bigger the widget tree. As the widget tree increases in size, the complexity of the design for the application increases.

Chapter 6
IMPLEMENTATION

Implementation is the process of defining how the system should be built, ensuring that it is operational and meets quality standards. It is a systematic and structured approach for effectively integrating software-based service or component into the requirements of end users.

6.1 About Visual Studio Code

Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion code refactoring and embedded Git.

6.1.1 Features

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 Visual Studio Code employs the same editor component (codenamed "Monaco") used in Azure DevOps . Visual Studio Code can be extended via extensions, available through a central repository. This includes additions to the editor and language support. 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**.**

6.1.2 Language Support

Out of the box, Visual Studio Code includes basic support for most common programming languages. This basic support includes syntax highlighting, bracket matching, code folding, and configurable snippets. Visual Studio Code also ships with IntelliSense for JavaScript, TypeScript, JSON, CSS, and HTML, as well as debugging support for Node.js. Support for additional languages can be provided by freely available extensions on the VS Code Marketplace.

6.1.3 Version Control

Source control is a built-in feature of Visual Studio Code. It has a dedicated tab inside of the menu bar where you can access version control settings and view changes made to the current project. To use the feature you must link Visual Studio Code to any supported version control system (Git, Apache Subversion, Perforce, etc.). This allows you to create repositories as well as make push and pull requests directly from the Visual Studio Code program.

6.2 About Flutter

Flutter is an open-source UI software development kit created by Google. It is used to develop cross platform applications for Android, IoS, Linus, Mac, Windows, Fushia and web from a single code base.

6.2.1 History

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[6] with the stated intent of being able to render consistently at 120 frames per second. 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 December 11, 2019, Flutter 1.12 was released at the Flutter interactive event.

On May 6, 2020, the Dart software development kit (SDK) in version 2.8 and the Flutter in version 1.17.0 were released, where support was added to the Metal API, improving performance on iOS devices (approximately 50%), new Material widgets, and new network tracking.

6.2.2 Widgets

Widgets are generally defined in three basic types: Stateful widgets, Stateless widgets, and Inherited widgets. Being the central class hierarchy in the Flutter framework the three basic types of widgets are used in the construction of every Flutter application. Although all the instances of a widget are immutable, the Stateful widget allows the interaction between user and application. By giving access to the method setState, the state can be maintained in separate state objects. Alternatively, the Stateless widget acts as a constant, and before anything displayed can be changed, the widget must be recreated

6.3 Pseudo Code

6.3.1 Main.dart

```
import 'package:challenge_infracea/pages/login/login_page.dart';
import 'package:challenge_infracea/utils/consts.dart';
import 'package:flutter/material.dart';
void main() {
 runApp(VeppoApp());}
class VeppoApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        fontFamily: 'Montserrat',
        scaffoldBackgroundColor: Colors.white,
        primarySwatch: white,
         hintColor: Colors.transparent,
        focusColor: Colors.transparent,
        hoverColor: Colors.transparent,
        indicatorColor: Colors.transparent,
        splashColor: Colors.transparent,
        highlightColor: Colors.transparent,),
      home: LoginPage(),
    );
  }
}
```

6.3.2 List_flights.dart

```
import 'package:challenge_infracea/pages/search/seats_grid_page.dart';
import 'package:challenge_infracea/providers/flight_provider.dart';
import 'package:challenge_infracea/utils/consts.dart';
import 'package:flutter/material.dart';
```

```
 import 'package:delayed_display/delayed_display.dart';
class ListFlights extends StatefulWidget {
  @override
_ListFlightsState createState() =>
_ListFlightsState();
}
class _ListFlightsState extends State<ListFlights>
    with TickerProviderStateMixin {
  PageController _cardController = PageController(initialPage: 0);
List<int> passengers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20];
  int currentPassengers = 1;
  int currentIndex = 0;

  @override
  dispose() {
    super.dispose();
    _cardController.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return ListView.builder(
      itemCount: flightsAvailable.length,
      itemBuilder: (context, index) {
        return DelayedDisplay(
          child: GestureDetector(
            onTap: () {
              setState(() {
                currentIndex = index;
              });
            },
            child: AnimatedContainer(
              height: currentIndex == index ? 134 : 66,
              duration: Duration(milliseconds: 200),
              margin: EdgeInsets.fromLTRB(
                  16, 0, 16, index == flightsAvailable.length - 1 ? 80 : 16),
              padding: EdgeInsets.all(16),
  decoration: BoxDecoration(
                color: Colors.white,
                borderRadius: BorderRadius.all(
Radius.circular(4),
                boxShadow: [
                  BoxShadow(
                    color: Colors.black.withOpacity(0.1),
                    spreadRadius: 2,
                    blurRadius: 4,
                  )
                ],
              ),
              child: PageView(
                controller: _cardController,
                children: [
                  Column(
                    children: [
                      Row(
                        crossAxisAlignment: CrossAxisAlignment.start,
                        children: [
                          Image.asset(
                            flightsAvailable[index].logo!,
                            width: 64,
                          ),
                          SizedBox(width: 16),
                      Container(
                            color: veppoLightGrey,
                            height: 32,
                            width: 1,
                          ),
                          SizedBox(width: 16),
                          Column(
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: [
                              Text(
                                'Select',
                                style: TextStyle(
              color: veppoLightGrey,
            ),
          ),
          Text(
            'Travel Date',
            style: TextStyle(
              color: veppoLightGrey,
            ),
          ),
        ],
      )
    ],
  ),
  Spacer(),
  AnimatedContainer(
    duration: Duration(milliseconds: 200),
    height: currentIndex == index ? 32 : 0,
    width: MediaQuery.of(context).size.width,
    child: ElevatedButton(
      style: ButtonStyle(
        backgroundColor:
            MaterialStateProperty.resolveWith<Color>(
          (Set<MaterialState> states) {
            if (states.contains(MaterialState.pressed))
              return veppoLightGrey;
            return veppoBlue;
          },
        ),
      ),
      onPressed: () {},
      child: Text(
        'Select Travel date',
        style: TextStyle(
          color: Colors.white,
        ),
      ),
    ),
  ),
          ],
         ),
         Column(
          children: [

  Row(
                mainAxisAlignment: MainAxisAlignment.start,
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                 Icon(
                  Icons.supervised_user_circle,
                  size: 32,
                 ),
                 SizedBox(width: 16),
                 Container(
                  color: veppoLightGrey,
                  height: 32,
                  width: 1,
                 ),
                 SizedBox(width: 16),
                 Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                   Text(
                    'Passengers',
                    style: TextStyle(
                     color: veppoLightGrey,
                    ),
                   ),
                   Text(
                    'Adults (12+)',
                    style: TextStyle(
                     color: veppoLightGrey,
                    ),
                   ),
                  ],
                 ),
           Spacer(),
                 Container(
                  height: 34,
                  child: ElevatedButton(
                   style: ButtonStyle(
                    backgroundColor:
                      MaterialStateProperty.resolveWith<Color>(
                     (Set<MaterialState> states) {
                      if (states.contains(MaterialState.pressed))
                       return veppoLightGrey;
                      return veppoBlue;
                     }),),
                   onPressed: () {
                    Navigator.push(
                     context,
                     MaterialPageRoute(
                      builder: (context) => SeatsGridPage(
                        flight: flightsAvailable[index]),
                     ),);},
                   child: Text(
                    'next step',
                    style: TextStyle(
                     color: Colors.white,
                    ),),),],)
               Spacer(),
               Expanded(
                child: ListView(
                 scrollDirection: Axis.horizontal,
                 children: passengers.map((e) {
                  return Container(
                   margin: EdgeInsets.all(4),
                   width: 32,
                   height: 32,
                   decoration: BoxDecoration(
                    shape: BoxShape.circle,
                    border: Border.all(
                     color: currentPassengers == e
                       ? veppoBlue
```

```
                                      : veppoLightGrey.withOpacity(0.4),
                                ),
                              ),
                              child: InkWell(
                                onTap: () {
                                  setState(() {
                                    currentPassengers = e;
                                  });
                                },
                                child: Center(
                                  child: Text(
                                    e.toString(),
                                    style: TextStyle(
                                      color: currentPassengers == e
                                          ? veppoBlue
                                          : veppoLightGrey.withOpacity(0.4),
                                      fontSize: 12,
                                    ),
                                  ),
                                ),
                              ),
                            );
                          }).toList(),
                        ),
                      ),
                    ],
                  ),
                ],
              ),
            ),
          ),
        );
      },
    );
  }
}
```

Chapter 7

TESTING

Testing is the process used to help identify the correctness, completeness, security, and quality of the developed computer application. Testing is the process of technical investigation and includes the process of executing a program or application with the intent of finding errors. Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test.

7.1 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. The unit testing is the process of testing the part of the program to verify whether the program is working correct or not. 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 in unit testing is to isolate a unit and validate its correctness.

Table 7.1 Unit test case for Flight Input Check

Case_ id	**Description**	**Input data**	**Expected Ouput**	**Actual Ouput**	**Status**
1	Opening the flight application after entering username and password	Username and Password	The flight application opens and displays the home screen	The flight application opens and displays the home screen	Pass
2	Search flight	Any option selected	Checks for available flights	Displays available flights	Pass
3	Show ticket	Ticket selected	Checks for ticket and displays it	Displays the flight ticket	Pass

4	Display trips	Select ‘trips’ button	Shows the list of trips	Shows the list of trips	Pass
5	Display ticket	Select desired trip	Shows complete ticket	Shows complete ticket	Pass
6	Closing application	-	Application should be closed without any error.	Application is closed	Pass

Chapter 8

DISCUSSION OF RESULTS

All the menu options provided in the application and its operations have been presented in as screen shots.

8.1 Login Page

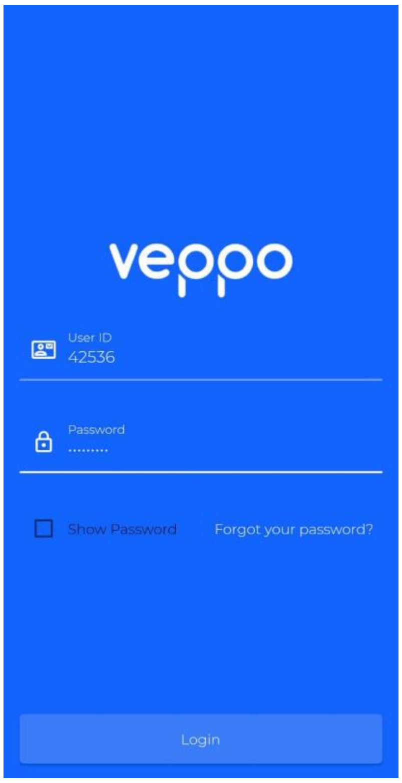

Figure 8.1 Login Page

The Figure 8.1 displays the login page where the user must enter the user id along with the password in order to login to the app.

There is an option to view the password and if selected, the user will be able to view the password. Once this is done, the user clicks on the login button to navigate to the app.

8.2 One way and Round trip Screen

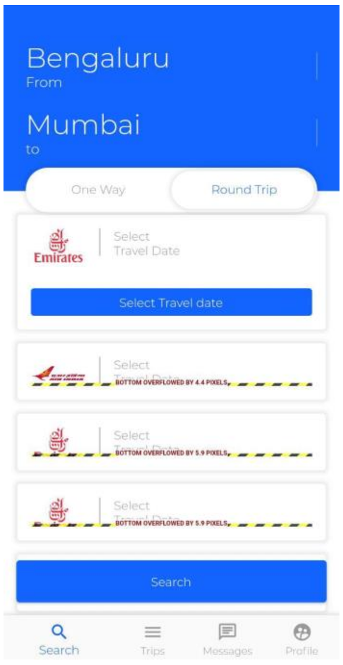

Figure 8.2 One way and Round trip Screen

The Figure 8.2 displays the one way trip and round trip flights that are available for booking. The user can select the flight they wish to book for the available list.

8.3 Search screen

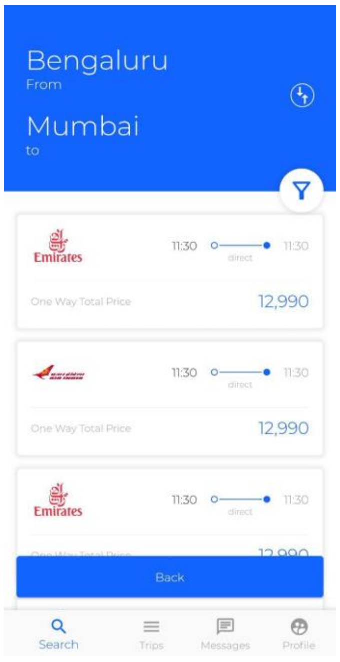

Figure 8.3 Search screen

The Figure 8.3 displays the search option. On clicking the search button, the user is navigated to a list of the flights available from which the user can choose.

8.4 Trips screen

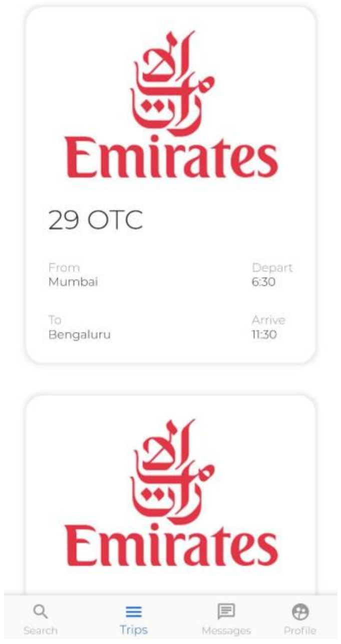

Figure 8.4 Trips screen

The Figure .4 displays the trips option. This option displays the trips taken by the user. It shows the list of journeys of the user. The user can click on these to view the full ticket.

8.5 Ticket screen

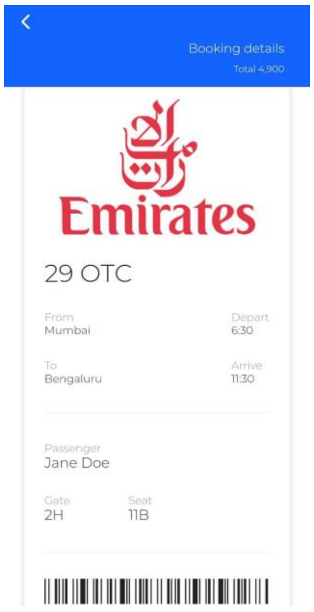

Figure 8.5 Ticket screen

The Figure 8.5 displays the complete ticket. Upon clicking the option in the trips page, the user is navigated to a page that displays the complete ticket.

This includes trip details like departure city and destination city, name, price, barcode, date, flight name and number, etc.

Chapter 9

CONCLUSION AND FUTURE ENHANCEMENTS

This internship project has been implemented with making appropriate effort to overcome the time-consuming problem of manual system. It has been successfully designed and implemented using Flutter. The system has been developed in a way to make it very user friendly. This application provides facility to book tickets anywhere and anytime. It saves time as the user need not wait in long queues to book tickets.

Future Enhancements:

- In future, the system can be implemented with various sources and destinations.
- Discounts and offers can be enabled for users.
- It can be possible to make it more user friendly by adding more variety of functions to it.
- Allowing users to select their choice of seat.
- It can be made fully functional by adding payment options.
- The interface can be connected to a database to facilitate data validation and storage.

REFERENCES

[1] https://flutter.dev/

[2] https://code.visualstudio.com/

Papers:

Paper [1]: https://ieeexplore.ieee.org/document/7845037

Paper [2]: https://ieeexplore.ieee.org/document/7476636

Paper [3]:

https://ijesc.org/upload/b3930ac14331fd1b425af8cd1c341d41.Cross%20Platform%20Development%20using%20Flutter%20(1).pdf

Paper [4]: https://ieeexplore.ieee.org/document/6516325

Paper [5]: https://ieeexplore.ieee.org/document/1450843