VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018

An Internship Project Report

on

Alarm Clock App

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

Rachit Tawani

1RN18IS082

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

Dr. Vishnuvaradhan Road, Rajarajeshwari Nagar post,

Channasandra, Bengaluru - 560098

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the Internship work entitled ***Alarm Clock App*** has been successfully completed by **Rachit Tawani (1RN18IS082)** bonafide students 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.

Mrs. Kavya TC	**Dr. Suresh L**	**Dr. M K Venkatesha**
Internship Guide	Professor and HoD	Principal
Assistant Professor	Department of ISE	RNSIT
Department of ISE	RNSIT	

External Viva

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

DECLARATION

I, **Rachit Tawani [USN: 1RN18IS082]** students of VIII Semester BE, in Information Science and Engineering, RNS Institute of Technology hereby declare that the Internship work entitled ***Alarm Clock App*** has been carried out by us and submitted in partial fulfillment of the requirements for the *VIII Semester degree of* ***Bachelor of Engineering in Information Science and Engineering*** *of Visvesvaraya Technological University, Belagavi* during academic year 2021-2022.

Place: Bengaluru **Rachit Tawani**

Date: **(1RN18IS082)**

ABSTRACT

Social media and other easily accessible online distractions make it hard for us to stay focused on our tasks and make it difficult for us to do our work efficiently.

It is more important for us to get a reminder for our tasks and so we can use this app to get those reminders by setting up alarm. It also helps us to do the most basic task of waking us up.

Such an app is helpful in coordinating our daily schedules. This app with impressive layout makes it easy for us to set alarm. It also makes it easy for us to know the current time, day and date using the Analog Clock.

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

I am 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.

I place our heartfelt thanks to **Mrs. Kavya TC** Assistant Professor, Department of Information Science and Engineering for having guided internship and all the staff members of the department of Information Science and Engineering for always helping.

I thank **Mr. Akshay D R, ENMAZZ**, 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.

Rachit Tawani

1RN18IS082

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

UI	:	User Interface
FK	:	Flutter Kick
IoT	:	Internet of Things
FCL	:	Flutter Cycle Length
AOT	:	Ahead of Time
SDK	:	Software Development Kit

Chapter 1

INTRODUCTION

1.1 Introduction to Flutter

Flutter is Google's Mobile SDK to build native iOS and Android, Desktop (Windows, Linux, macOS), 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.

1.2 History

Flutter launched as a project called Sky which at the beginning worked only on Android. Flutter's goal is enabling 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 [6] with the stated intent of being able to render consistently at 120 frames per second.[7] 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.[8]

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.

On March 3, 2021, Google released Flutter 2 during an online Flutter Engage event. This major update brought official support for web-based applications with new Canvas Kit renderer and web specific widgets, early-access desktop application support for Windows, macOS, and Linux and improved Add-to-App APIs.[9] This release included sound null-safety, which caused many breaking changes and issues with many external packages, but the Flutter team included instructions to mitigate these changes as well.

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 called Material You. Dart received two new updates, the newest lint conditions have been standardized and preset as the default conditions as well Dart for Apple Silicon is now stable.

1.3 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[11] 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.

Flutter engine

Flutter's engine, written primarily in C++, provides low-level rendering support using Google's Skia graphics library. Additionally, it interfaces with platform-specific SDKs such as those provided by Android and iOS.[10] 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 interact with Flutter via the Flutter Framework, which provides a reactive framework and a set of platform, layout, and foundation widgets.

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.

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 SURVEY

2.1 BEDRUNNER: AN INTELLIGENT RUNNING ALARM CLOCK

The conventional alarm clocks have been serving their purpose to mankind since their inception, to wake people up every single morning. However, the rates of oversleeping are still on the rise and people are having trouble waking up in the morning even with the use of alarm clocks. The snooze button that is available on all conventional alarm clocks provide user with more sleep but at a cost of deteriorating their quality of health and exacerbating sleep inertia at the same time. The objective of this paper is to study the problems associated with sleep that are often faced by people and to develop an intelligent moving alarm clock that implements the application of Artificial Intelligence. The Hypothetico-Deductive method will be used as the research methodology while the development will be following the Rapid Application Development (RAD) model. The developed prototype is tested against conventional alarm clocks and has shown significant improvements in the percentages of oversleeping and snoozing. Recommendations for continuation and future work of this paper are also included.

2.1.1 Introduction

Every living person on this globe share one common first thing that they all have to do every single day; waking up [1]. Even though it may sound simple, some people find it hard to wake up on time in the morning, especially for heavy sleepers. They oversleep and end up getting to work, classes or lectures late or even worse, missing out on important events in their life. Although alarm clocks have been developed to help combat this problem, they are still unable to stop people from oversleeping [2]. Therefore, one question exists; how can current engineering and technology assist to curb this problem? By applying my knowledge in Artificial Intelligence and robotics, I hope to be able to design an innovative and exciting way to develop a smart alarm clock as a solution to this situation.

2.1.2 Related work

2.1.2.1 Alarm Clock application

Oversleeping refers to an intentional or unintentional act of sleeping beyond one's intended time for waking or intended time for getting up. Regardless of age, race, background and gender, everyone is subjected to the risk of oversleeping in the morning, especially if they do not have a proper sleeping pattern or do not have enough sleep regularly. This statement is supported by a research, stating that in Japan, about half of the students in senior high schools are sleeping less than 6 hours during weekdays. Their sleeping pattern have been altered as a result of the habit of staying up late to do work and to study for tests. Another study also shows that by not having a regular sleep pattern, students are subjected to higher risks of oversleeping in the morning. According to the results of the CDC's Youth Risk Behavior Survey in 2011 and 2013, high school students in the United States are not having a regular and ample amount of sleep, with 69% of the students sleeping less than 8 hours a day and 40% sleeping less than 6 hours. Another poll by National Sleep Foundation also recorded that 59% of 6th to 8th graders and 87% of U.S. high school students were getting less than the recommended 8.5 to 9.5 hours of sleep on weekdays.

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
Front-end	:	Flutter

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

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.

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.

Chapter 4

SYSTEM DESIGN

4.1 Alarm Page Widget Tree:

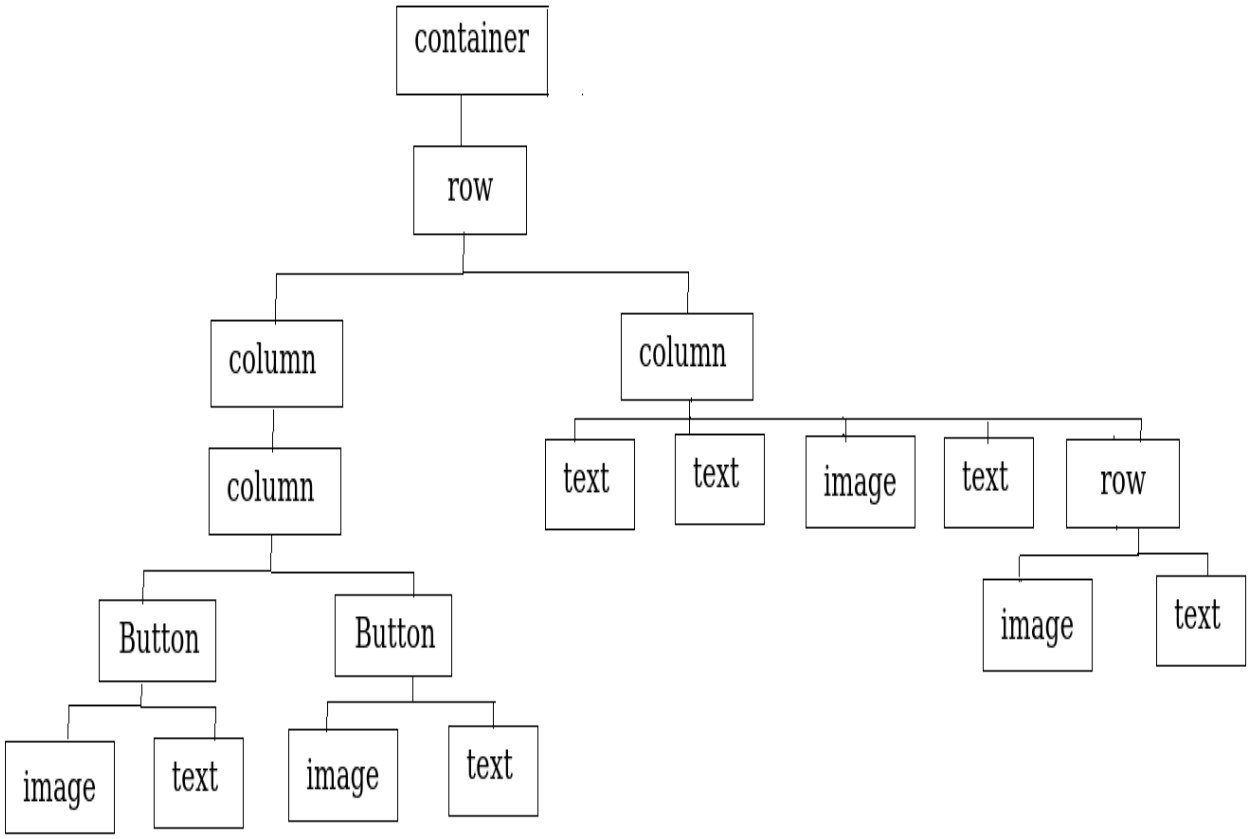

Fig 4.1 Clock Page Widget Tree

4.2 Alarm Page Widget Tree:

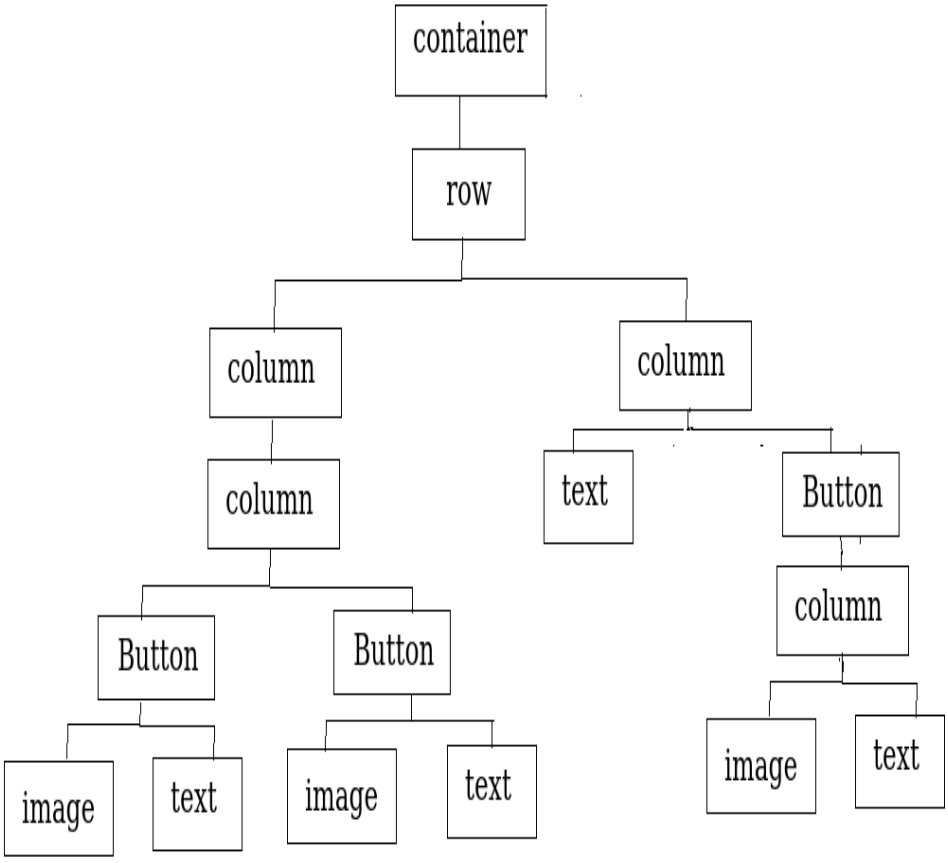

Fig 4.2 Alarm Page Widget Tree

Chapter 5

IMPLEMENTATION DETAILS

5.1 alarm.dart

```
import 'package:clock_app/alarm_helper.dart';
import 'package:clock_app/constants/theme_data.dart';
import 'package:clock_app/data.dart';
import 'package:clock_app/models/alarm_info.dart';
import 'package:dotted_border/dotted_border.dart';
import 'package:flutter/material.dart';
import 'package:flutter_local_notifications/flutter_local_notifications.dart';
import 'package:intl/intl.dart';

import '../main.dart';

class AlarmPage extends StatefulWidget {
  @override
  _AlarmPageState createState() => _AlarmPageState();
}

class _AlarmPageState extends State<AlarmPage> {
  DateTime _alarmTime;
  String _alarmTimeString;
  AlarmHelper _alarmHelper = AlarmHelper();
  Future<List<AlarmInfo>> _alarms;
  List<AlarmInfo> _currentAlarms;

  @override
  void initState() {
    _alarmTime = DateTime.now();
    _alarmHelper.initializeDatabase().then((value) {
      print('------database intialized');
      loadAlarms();
    });
    super.initState();
  }

  void loadAlarms() {
    _alarms = _alarmHelper.getAlarms();
    if (mounted) setState(() {});
  }
```

```
Text(
          'Alarm',
          style: TextStyle(
              fontFamily: 'avenir',
              fontWeight: FontWeight.w700,
              color: CustomColors.primaryTextColor,
              fontSize: 24),
        ),
        Expanded(
          child: FutureBuilder<List<AlarmInfo>>(
            future: _alarms,
            builder: (context, snapshot) {
              if (snapshot.hasData) {
                _currentAlarms = snapshot.data;
                return ListView(
                  children: snapshot.data.map<Widget>((alarm) {
                    var alarmTime =
                        DateFormat('hh:mm aa').format(alarm.alarmDateTime);
                    var gradientColor = GradientTemplate
                        .gradientTemplate[alarm.gradientColorIndex].colors;
                    return Container(
                      margin: const EdgeInsets.only(bottom: 32),
                      padding: const EdgeInsets.symmetric(
                          horizontal: 16, vertical: 8),
                      decoration: BoxDecoration(
                        gradient: LinearGradient(
                          colors: gradientColor,
                          begin: Alignment.centerLeft,
                          end: Alignment.centerRight,
                        ),
                        boxShadow: [
                          BoxShadow(
                            color: gradientColor.last.withOpacity(0.4),
                            blurRadius: 8,
                            spreadRadius: 2,
                            offset: Offset(4, 4),
                          ),
                        ],
                        borderRadius: BorderRadius.all(Radius.circular(24)),
                      ),
                      child: Column(
                        crossAxisAlignment: CrossAxisAlignment.start,
                        children: <Widget>[
                          Row(
                            mainAxisAlignment: MainAxisAlignment.spaceBetween,
                            children: <Widget>[
                              Row(
                                children: <Widget>[
                                  Icon(
                                    Icons.label,
                                    color: Colors.white,
                                    size: 24,
```

```
SizedBox(width: 8),
    Text(
              alarm.title,
              style: TextStyle(
              color: Colors.white,
              fontFamily: 'avenir'),
          ),
          ],
    ),
    Switch(
          onChanged: (bool value) {},
              value: true,
              activeColor: Colors.white,
    ),
    ],
    ),
   Text(
              'Mon-Fri',
              style: TextStyle(
              color: Colors.white, fontFamily: 'avenir'),
    ),
     Row(
              mainAxisAlignment:
              MainAxisAlignment.spaceBetween,
              children: <Widget>[
         Text(
               alarmTime,
               style: TextStyle(
               color: Colors.white,
               fontFamily: 'avenir',
               fontSize: 24,
               fontWeight: FontWeight.w700),
            ),
              IconButton(
              icon: Icon(Icons.delete),
              color: Colors.white,
              onPressed: () {
              deleteAlarm(alarm.id);
              }),
              ],
```

```
void scheduleAlarm(
    DateTime scheduledNotificationDateTime, AlarmInfo alarmInfo) async {
  var androidPlatformChannelSpecifics = AndroidNotificationDetails(
    'alarm_notif',
    'alarm_notif',
    'Channel for Alarm notification',
    icon: 'codex_logo',
    sound: RawResourceAndroidNotificationSound('a_long_cold_sting'),
    largeIcon: DrawableResourceAndroidBitmap('codex_logo'),
  );

  var iOSPlatformChannelSpecifics = IOSNotificationDetails(
      sound: 'a_long_cold_sting.wav',
      presentAlert: true,
      presentBadge: true,
      presentSound: true);
  var platformChannelSpecifics = NotificationDetails(
      androidPlatformChannelSpecifics, iOSPlatformChannelSpecifics);

  await flutterLocalNotificationsPlugin.schedule(0, 'Office', alarmInfo.title,
      scheduledNotificationDateTime, platformChannelSpecifics);
 }


void onSaveAlarm() {
  DateTime scheduleAlarmDateTime;
  if (_alarmTime.isAfter(DateTime.now()))
    scheduleAlarmDateTime = _alarmTime;
  else
    scheduleAlarmDateTime = _alarmTime.add(Duration(days: 1));

  var alarmInfo = AlarmInfo(
    alarmDateTime: scheduleAlarmDateTime,
    gradientColorIndex: _currentAlarms.length,
    title: 'alarm',
  );
  _alarmHelper.insertAlarm(alarmInfo);
  scheduleAlarm(scheduleAlarmDateTime, alarmInfo);
  Navigator.pop(context);
  loadAlarms();
 }

 void deleteAlarm(int id) {
  _alarmHelper.delete(id);
  //unsubscribe for notification
  loadAlarms();
 }
}
```

5.2 clock_page.dart

```
import 'dart:async';

import 'package:clock_app/constants/theme_data.dart';
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';

import 'clockview.dart';

class ClockPage extends StatefulWidget {
  @override
  _ClockPageState createState() => _ClockPageState();
}

class _ClockPageState extends State<ClockPage> {
  @override
  Widget build(BuildContext context) {
    var now = DateTime.now();

    var formattedDate = DateFormat('EEE, d MMM').format(now);
    var timezoneString = now.timeZoneOffset.toString().split('.').first;
    var offsetSign = '';
    if (!timezoneString.startsWith('-')) offsetSign = '+';

    return Container(
      padding: EdgeInsets.symmetric(horizontal: 32, vertical: 64),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: <Widget>[
          Flexible(
            flex: 1,
            fit: FlexFit.tight,
            child: Text(
              'Clock',
              style: TextStyle(
                  fontFamily: 'avenir',
                  fontWeight: FontWeight.w700,
                  color: CustomColors.primaryTextColor,
                  fontSize: 24),
            ),
          ),
          Flexible(
            flex: 2,
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: <Widget>[
                DigitalClockWidget(),
                Text(
                  formattedDate,
                  style: TextStyle(
                      fontFamily: 'avenir',
                      fontWeight: FontWeight.w300,
                      color: CustomColors.primaryTextColor,
                      fontSize: 20),
```

```
        Flexible(
                flex: 4,
                fit: FlexFit.tight,
                child: Align(
                  alignment: Alignment.center,
                  child: ClockView(
                    size: MediaQuery.of(context).size.height / 4,
                  ),
                ),
              ),
              Flexible(
                flex: 2,
                fit: FlexFit.tight,
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: <Widget>[
                    Text(
                      'Timezone',
                      style: TextStyle(
                          fontFamily: 'avenir',
                          fontWeight: FontWeight.w500,
                          color: CustomColors.primaryTextColor,
                          fontSize: 24),
                    ),
                    SizedBox(height: 16),
                    Row(
                      children: <Widget>[
                        Icon(
                          Icons.language,
                          color: CustomColors.primaryTextColor,
                        ),
                        SizedBox(width: 16),
                        Text(
                          'UTC' + offsetSign + timezoneString,
                          style: TextStyle(
                              fontFamily: 'avenir',
                              color: CustomColors.primaryTextColor,
                              fontSize: 14),
                        ),
                      ],
                    ),
                  ],
                ),
              ),
            ],
          ),
        );
      }
    }class DigitalClockWidget extends StatefulWidget {
      const DigitalClockWidget({
        Key key,
      }) : super(key: key);
      @override
      State<StatefulWidget> createState() {
        return DigitalClockWidgetState();
      }
    }
```

```
DigitalClockWidgetState extends State<DigitalClockWidget> {
  var formattedTime =
    DateFormat('HH:mm').format(DateTime.now());
  Timer timer;

  @override
  void initState() {
    this.timer = Timer.periodic(Duration(seconds:
    1), (timer) {
      var perviousMinute =
    DateTime.now().add(Duration(seconds: -
    1)).minute;
      var currentMinute = DateTime.now().minute;
      if (perviousMinute != currentMinute)
        setState(() {
          formattedTime =
    DateFormat('HH:mm').format(DateTime.now());
        });
    });
    super.initState();
  } @override
  void dispose() {
    this.timer.cancel();
    super.dispose();
  }
```

5.3 main.dart

```
import 'package:clock_app/enums.dart';
import 'package:clock_app/models/menu_info.dart';
import 'package:flutter/material.dart';
import
'package:flutter_local_notifications/flutter_local_notificat
ions.dart';
import 'package:provider/provider.dart';
import 'views/homepage.dart';

final FlutterLocalNotificationsPlugin
flutterLocalNotificationsPlugin =
    FlutterLocalNotificationsPlugin();
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  var initializationSettingsAndroid =
      AndroidInitializationSettings('codex_logo');



class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity:
VisualDensity.adaptivePlatformDensity,
      ),
      home: ChangeNotifierProvider<MenuInfo>(
        create: (context) => MenuInfo(MenuType.clock),
        child: HomePage(),
      ),
    );
  }
}
```

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 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

6.2.1 Unit Testing

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

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. A manual approach to unit testing may employ a step-by-step instructional document. The unit testing is the process of testing the part of the program to verify whether the program is working corrector not. In this part the main intention is to check the each and every input which we are inserting to 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 and can be used in a bottom-up testing style approach. By testing the parts of a program first and then testing the sum of its parts, integration testing becomes much easier. Unit test cases embody characteristics that are critical to the success of the unit.

6.2.2 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.

6.2.3 System Testing

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

6.2.4 Validation Testing

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

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

6.2.5 Output Testing

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

6.2.6 User Acceptance Testing

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

User Acceptance Testing is done in the final phase of testing.

Chapter 7

DISCUSSION OF RESULTS

7.1 Clock page

This is the Landing Page or the Clock Page of the application where we can see the analog clock which shows Indian Time. It also shows current day, date and time.

Fig 7.1 Clock Page

7.2 Alarm Page

This is the Alarm Page which shows either no alarm or previous alarm. Here we can click “Add Alarm” button to add alarms. Multiple alarms (up to 5) can be added.

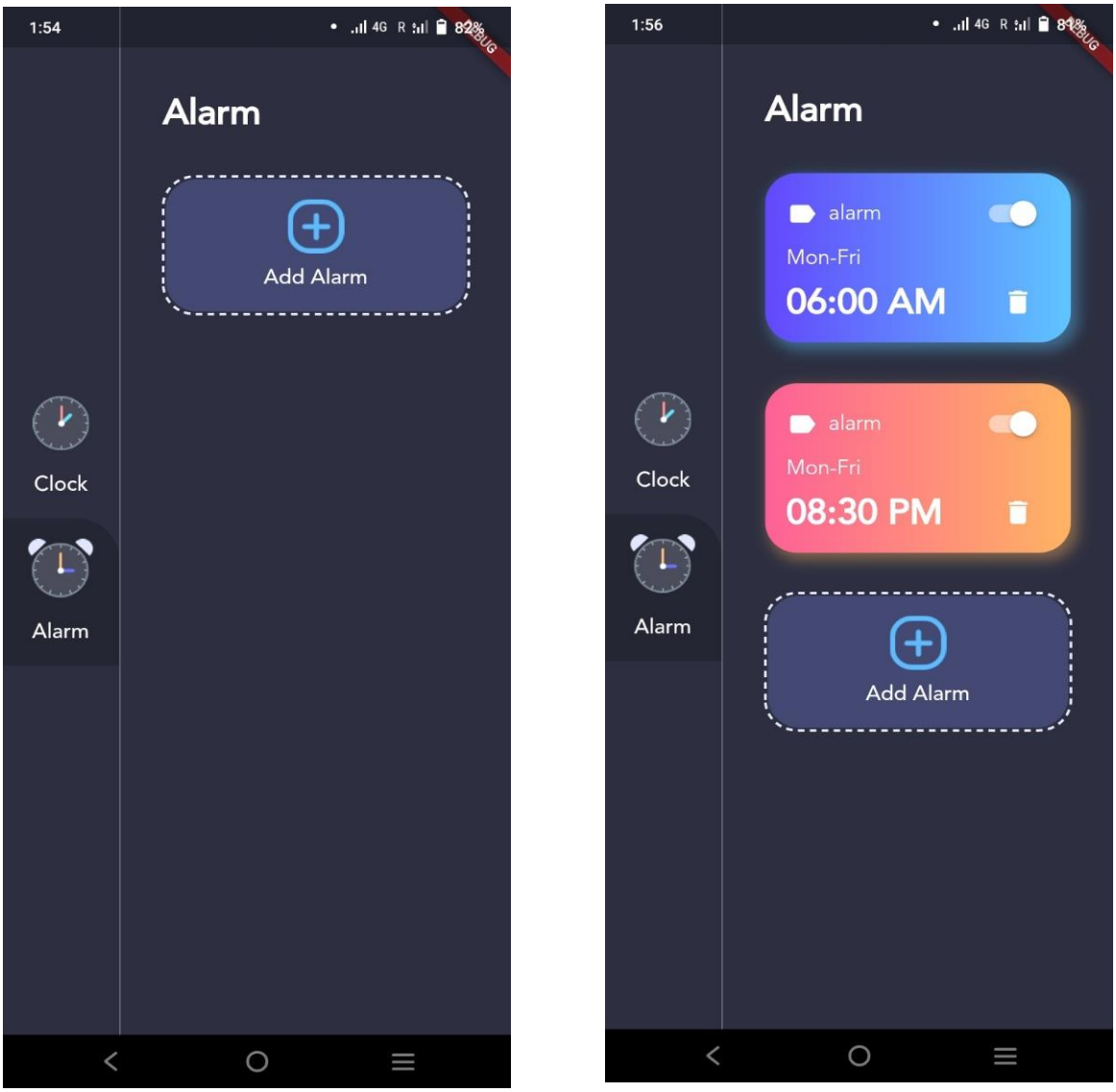

Fig 7.2 Alarm Page

7.3 Creating an Alarm

Creating a new alarm can either be done using pointer or by using keyboard.

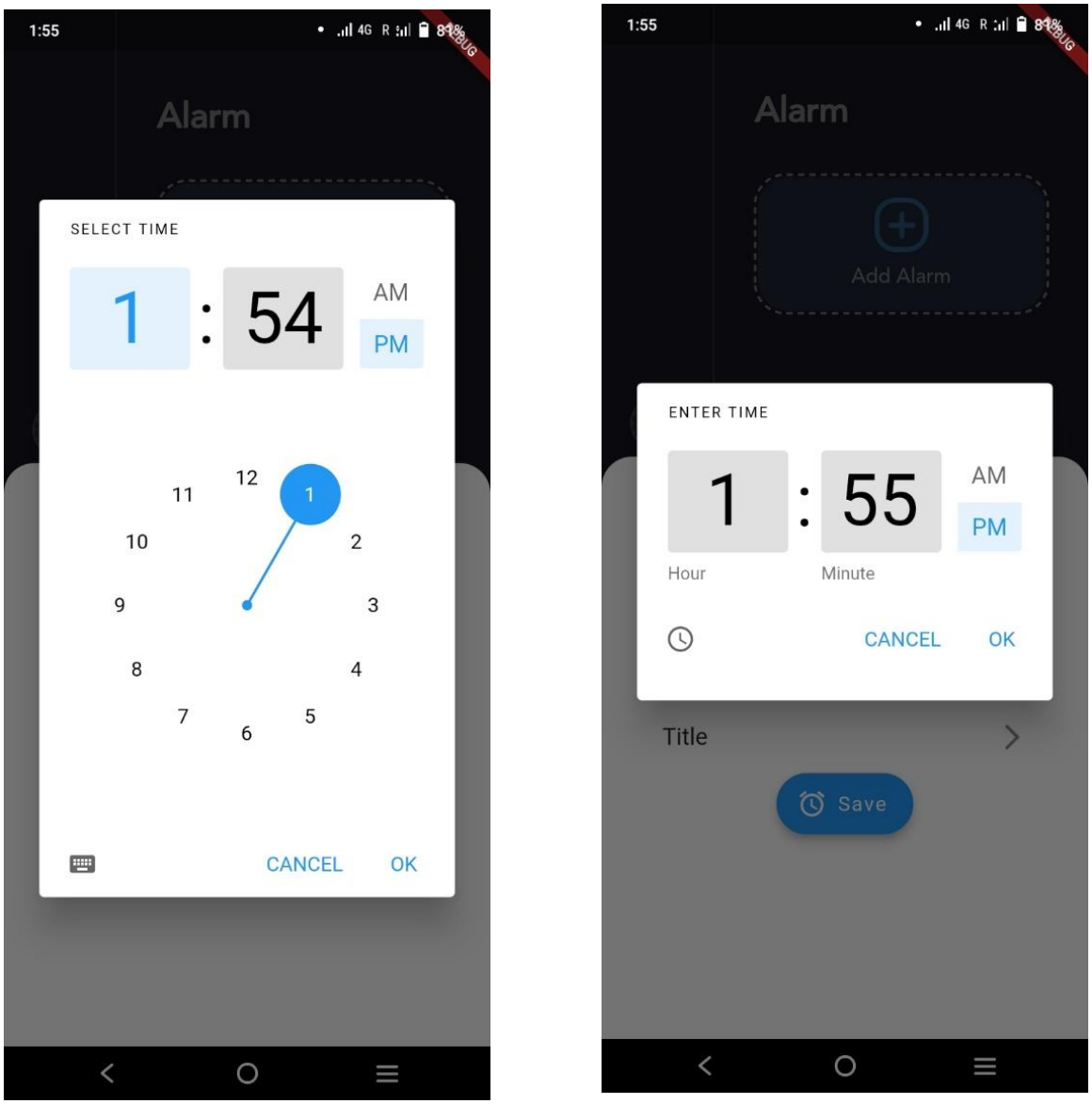

Fig 7.3 Creating an Alarm

Chapter 8

CONCLUSION AND FUTURE WORK

8.1 Conclusion

- This application is beneficial for the us as we can get a reminder for our tasks.
- It is helpful in planning our daily schedules. We can add and delete the alarms.
- It also helps us to know the current day, date and time.

8.2 Future work

- We can add some extra buttons for stop and snooze.
- We can further add features like changing different songs for the alarm.
- We can add the ability to change the time zones to see the time of different countries in Analog Clock.
- We can add pages like Stopwatch and Timer.

Chapter 9

REFERENCES

- https://flutter.dev/
- https://developers.google.com/learn/pathways/intro-to-flutter
- Beginning Flutter: A Hands On Guide to App Development by Marco L Napoli
- https://stackoverflow.com/
- https://www.geeksforgeeks.org/