

Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and Engineering

By

PAUL SHRUTI KANAILAL

(Reg.No. 40731077)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SCHOOL
OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY JEPPIAAR
NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

APRIL 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of PAUL SHRUTI KANAILAL (40731077) who carried out the project entitled “TODO LIST APPLICATION USING FLUTTER” under my supervision from Jan 2023 to Apr 2023.

Internal Guide

Ms K Ishwarya, M. Tech.(Ph.D)

Head of the Department

Dr.S.Vigneshwari, M.E., Ph.D.

Submitted for Viva voce Examination held on

17-04-2023

Internal Examiner

External Examiner

DECLARATION

I PAUL SHRUTI KANAILAL hereby declare that the Project Report entitled **TODO LIST APPLICATION USING FLUTTER** done by me under the guidance of **Ms K Ishwarya,M.Tech.(Ph.D)** at **ALGOXFUSION** is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: **17-04-2023**

PLACE: **Chennai**



SIGNATURE OF THECANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D** ,Dean, School of Computing, **Dr. S. Vigneshwari,,M.E.,Ph.D** ,Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms. K Ishwarya, M. Tech(Ph. D)** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science and Engineering who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE



ABSTRACT

The definition of a to do list is simple. This is a list of tasks you need to complete or things you want to do. We have many options to help us stay on top of our daily responsibilities. For some people, a simple "to do" list is enough. Others prefer to use apps. People have preferences about where they check tasks—via desktop utilities ("thick app"), websites ("thin app"), or on the phone ("mobile"). Some software programs are very general and only have a list of task names and due dates. Some are more advanced with nested missions where you have to complete the nested missions for the main mission to be considered complete. Some business software is very advanced and costs a lot - check Microsoft Project and similar offers. Some software programs focus on specific industries, construction, manufacturing, business consulting, website management, wedding planning or even cooking recipes. One of the main reasons you should use a to-do list is that it helps you stay organized. Writing all your tasks down in a list will make them seem more manageable. Having a clear overview of the tasks you need to do and what you have completed will help you focus. At the same time free up space in your mind for other more creative tasks. Once you have completed a task, you can remove it from the list. It gives you a sense of progress and accomplishment that you miss if you're always rushing from one task to the next. When you feel accomplished, it spurs you on and motivates you to keep going.

TABLE OF CONTENTS

Chapter No	TITLE	Page No
	ABSTRACT	6
	LIST OF FIGURES	9
1.	INTRODUCTION	1
1.1	Features of flutter	
1.1.1	Advantages of Flutter	
1.1.2	Disadvantages of Flutter	
1.2	Dart Programming	
1.2.1	Advantages of Dart Programming	
1.2.2	Disadvantages of Dart	
1.3	Widgets	
1.3.1	Types of Widgets	
1.3.1.1	Stateless widget	
1.3.1.2	Stateful widget	
1.3.2	Gestures	
1.4	System Requirements and Installation Procedure	
2.	AIM AND SCOPE OF THE PRESENT INVESTIGATION	17
2.1	Aim	
2.2	Outlook	
3.	ALGORITHM AND METHODOLOGIES USED	18
3.1	Starting Up	
3.2	Including required dependencies at pubspec.yml	
3.2	Flowchart	
3.3	Components and Widgets Used	
3.2.1	Components	
3.2.2	Widgets	
3.2.2.1	In main.dart	
3.2.2.2	In home_page.dart	
4.	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS	24
4.1	Results	
4.1.1	Homepage intro section	
4.1.2	Add To-Do dialogue box intro section	

4.1.3	Edit and delete options implemented using Flutter_	
4.1.4	slidable package	
4.1.4	To-Do list and Completed list	
4.2	Performance Analysis	
5	SUMMARY AND CONCLUSION	27
5.1	Summary	
5.2	Conclusion	
	APPENDIX	28
	REFERENCE	34

LIST OF FIGURES

FIG 1 – Architecture Diagram	24
FIG 2 – Home Page	29
FIG 3 – New Task	30
FIG 4 - Task Updation	31
FIG 5 – TO Do List	32

CHAPTER 1

INTRODUCTION

Creating a mobile application is generally a difficult and demanding undertaking. For the development of mobile applications, there are numerous frameworks available. The native frameworks offered by iOS and Android are based on the Objective-C/Swift programming languages, respectively.

However, we must code in two distinct languages and use two different frameworks in order to create an application that supports both OSs. Mobile frameworks supporting both OS exist to aid in overcoming this complication. These frameworks range from straightforward HTML-based hybrid mobile application frameworks (which employ JavaScript for application logic and HTML for user interface) to intricate language-specific frameworks (which do the heavy lifting of converting code to native code). No matter how simple or complex they are, these frameworks always have a number of drawbacks, with their poor performance being one of the most significant.

In this scenario, Flutter – a simple and high performance framework based on Dart language, provides high performance by rendering the UI directly in the operating system's canvas rather than through native framework. Flutter also offers many ready to use widgets (UI) to create a modern application. These widgets are optimized for mobile environment and designing the application using widgets is as simple as designing HTML.

In particular, the Flutter application itself is a widget. Gestures and animations are also supported by Flutter widgets. Reactive programming is the foundation for the application logic. A state for the widget might be an option. Flutter will automatically (reactive programming) compare the widget's old and new states by changing the widget's state, rendering the widget with only the necessary modifications rather than re-rendering the widget as a whole.

1.1 FEATURES OF FLUTTER

Flutter framework offers the following features to developers –

- Modern and reactive framework.
- Uses Dart programming language and it is very easy to learn.

- Fast development.
- Beautiful and fluid user interfaces.
- Huge widget catalog.
- Runs same UI for multiple platforms
- High performance application

1.1.1 Advantages of flutter

Flutter comes with beautiful and customizable widgets for high performance and outstanding mobile application. It fulfils all the custom needs and requirements. Besides these, Flutter offers many more advantages as mentioned below –

- Dart has a large repository of software packages which lets you to extend the capabilities of your application.
- Developers need to write just a single code base for both applications (both Android and iOS platforms). Flutter may to be extended to other platform as well in the future.
- Flutter needs lesser testing. Because of its single code base, it is sufficient if we write automated tests once for both the platforms.
- Flutter's simplicity makes it a good candidate for fast development. Its customization capability and extendibility makes it even more powerful.
- With Flutter, developers has full control over the widgets and its layout.
- Flutter offers great developer tools, with amazing hot reload.

1.1.2 Disadvantages of flutter

Despite its many benefits, flutter has the following drawbacks:

- A developer must learn a new language because it is coded in Dart (though it is easy to learn).
-

- While a modern framework tries to keep logic and UI as separate as possible, Flutter mixes logic and user interface. This can be overcome by using high-level modules to separate the user interface and logic and clever coding.
- Flutter is yet another mobile application development framework. In a highly competitive market, choosing the right development tools can be challenging for developers.

1.2 DART PROGRAMMING

Dart is a general-purpose programming language that is open-source. Google was the one who developed it initially. Dart is an object-oriented programming language with syntax similar to C. Dart, in contrast to other programming languages, does not support arrays and supports programming concepts like interfaces and classes. Replicating data structures like arrays, generics, and optional typing is possible with Dart collections.

The Dart programming language supports the following data types:

- Numbers, which are used to represent integer and double literals.
- Strings—It represents a character sequence. Single or double quotes can be used to specify string values.
- Booleans Dart represents true and false Boolean values with the bool keyword.
- Maps and lists: It's used to show a collection of things.

1.2.1 Advantages of dart

- The ease of learning is the first benefit. Dart programming can be quickly learned by any JavaScript programmer. They only need to learn the fundamentals of this programming language to accomplish this.
- The availability of documentation is the second aspect. Since Google is working on the Dart interpreter, every aspect of the language is explained in detail. This lets you quickly get answers to almost any question you might have while writing code or during the training process.
- Its high performance is the third benefit. Dart-based programs typically run faster than JavaScript-based ones.
- Dart is very stable and can be used to create real-time applications of production quality. The inheritance, interface, and optional typing features of this object-oriented programming language are all supported.

- It employs AOT and JIT compilation; Dart is unique in that it can manage both ahead-of-time and just-in-time compilation. The Dart code can be converted directly into native machine code in AOT. It can be compiled for gamechanging workflow and extremely quick development cycles when in the JIT mode.
- DartPad is the tool for you if you want to start writing your first Dart program without installing or configuring anything. The DartPad user interface is straightforward. To run your code, write it down and click "Run." Dart programming is introduced to users through DartPad. DartPad's library support is limited to the most fundamental level.

1.2.2 Disadvantages of dart

- The first disadvantage is that Dart has few online resources, making it difficult to solve issues. The lack of larger, more cohesive developer communities that will support you is the cause.

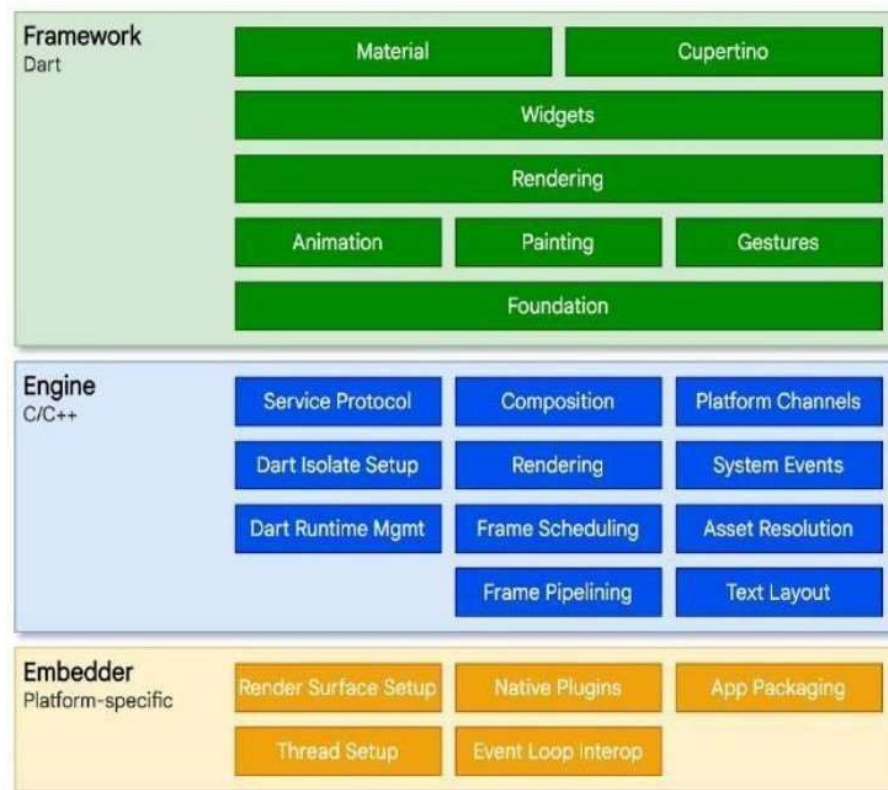


Fig.1.2. Flutter architecture

- Dart is relatively new to programmers and is rarely utilized on the market. Because of this, it will be harder for you to find a Dart programmer position at a company that hasn't yet advertised widely.
- Dart is being developed at the moment. That's exciting, and it has nothing to do with Dart. However, if you start developing in Dart right now, there is a chance that the API will change or that things won't be fully or correctly documented. Additionally, the amount of information available online is less than what you can find about jQuery, for example.

- There is no native support available. Developers must implement Dart in users' browsers in order to provide a stunning demonstration of its capabilities. It turns out that Dart developers must provide native support for it in all popular browsers in order to achieve at least 80% success.
- In conclusion, Dart is a fascinating programming language that includes tools for making Web, mobile, and command-line applications easier. Its stability and ease of learning are its primary advantages. On account of Web applications, the mix with AngularDart makes it an extremely incredible asset. In general, we believe that, despite its novelty, you should pay attention to this programming language.

1.3 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 11 minimal changes needed in the underlying render tree to transition from one state to the next.

Each element on a screen of the Flutter app is a widget. The view of the screen completely depends upon the choice and sequence of the widgets used to build the app. And the structure of the code of an app is a tree of widgets. Widgets are the central class hierarchy in the Flutter framework. A widget is an immutable description of part of a user interface. Widgets can be inflated into elements, which manage the underlying render tree.

Widgets themselves have no mutable state (all their fields must be final). If you wish to associate mutable state with a widget, consider using a `StatefulWidget`, which creates a `State` object (via `StatefulWidget.createState`) whenever it is inflated into an element and incorporated into the tree.

1.3.1 Types of Widgets

There are broadly two types of widgets in the flutter:

1. Stateless Widget
2. StatefulWidget

The State is the information that can be read synchronously when the widget is built and might change during the lifetime of the widget

1.3.1.1 Stateless Widget

The widgets whose state can not be altered once they are built are called stateless widgets. These widgets are immutable once they are built i.e any amount of change in the variables, icons, buttons, or retrieving data can not change the state of the app. Stateless widget overrides the build() method and returns a widget.

1.3.1.2 Stateful Widget

The widgets whose state can be altered once they are built are called stateful Widgets. These states are mutable and can be changed multiple times in their lifetime. This simply means the state of an app can change multiple times with different sets of variables, inputs, data. Stateful widget overrides the createState() method and returns a State. It is used when the UI can change dynamically. Some examples can be CheckBox, RadioButton, Form, TextField.

1.3.2 Gestures

Flutter widgets support interaction through a special widget, GestureDetector. GestureDetector is an invisible widget having the ability to capture user interactions such as tapping, dragging, etc., of its child widget. Many native widgets of Flutter support interaction through the use of GestureDetector. We can also incorporate interactive feature into the existing widget by composing it with the GestureDetector widget. We will learn the gestures separately in the upcoming chapters.

1.4 System Requirements and Installation Procedure

System requirements To install and run Flutter, your development environment must meet these minimum requirements:

- Operating Systems: Windows 10 or later (64-bit), x86-64 based.
- Disk Space: 1.64 GB (does not include disk space for IDE/tools).
- Tools: Flutter depends on these tools being available in your environment.
 - o [Windows PowerShell 5.0](#) or newer (this is pre-installed with Windows 10) o [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.

Installation Procedure

Step 1 – Go to URL, <https://flutter.dev/docs/get-started/install/windows> and download the latest Flutter SDK. As of April 2019, the version is 1.2.1 and the file is flutter_windows_v1.2.1-stable.zip.

Step 2 – Unzip the zip archive in a folder, say C:\flutter\

Step 3 – Update the system path to include flutter bin directory.

Step 4 – Flutter provides a tool, flutter doctor to check that all the requirement of flutter development is met.

```
flutter doctor
```

Step 5 – Running the above command will analyze the system and show its report as shown below –

Doctor summary (to see all details, run flutter doctor -v):

```
[√] Flutter (Channel stable, v1.2.1, on Microsoft Windows [Version  
10.0.17134.706], locale en-US) [√] Android toolchain - develop for Android  
devices (Android SDK version 28.0.3)  
[√] Android Studio (version 3.2)  
[√] VS Code, 64-bit edition (version 1.29.1)  
[!] Connected device  
! No devices available  
! Doctor found issues in 1 category.
```

The report says that all development tools are available but the device is not connected. We can fix this by connecting an android device through USB. Step 6 – Install the latest Android SDK, if reported by flutter doctor

Step 7 – Install the latest Android Studio, if reported by flutter doctor

Step 8 – Start an android emulator or connect a real android device to the system.

Step 9 – Install Flutter and Dart plugin for Android Studio. It provides startup template to create new Flutter application, an option to run and debug Flutter application in the Android studio itself, etc.,

- Open Android Studio.
- Click File → Settings → Plugins.
- Select the Flutter plugin and click Install.
- Click Yes when prompted to install the Dart plugin.
- Restart Android studio.

CHAPTER 2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 AIM

The definition is a simple one. It's a list of tasks you need to complete or things that you want to do. Most typically, they're organised in order of priority. Traditionally, they're written on a piece of paper or post it notes and act as a memory aid. As technology has evolved we have been able to create a todo lists with excel spreadsheets, word documents, email,lists etc. Here we are using todo list app for this process We can use a todo list in our home and personal life, or in the workplace.

2.2 OUTLOOK

To start with, it is better to mention that a typical to-do list (also called as a list of things, a list of errands, things-to-do list, etc.) is just a list of tasks for a certain period of time, a day or a week.

Initially, to-do list was created for the purpose to be a well-structured work plan helps every person to see his current load and execute all the necessary tasks. Now it becomes a common practice almost for everyone, and it solves many problems concerning the high workload.

To-do list has its following functions:

- Organization of a proper work with an [accurate planning](#) of every task with its short description
- Simplify the workflow and a timely execution of necessary work scope
- Improvement of every employee's [productivity](#) and time management skills.

CHAPTER 3

ALGORITHM AND METHODOLOGIES USED

3.1 Starting up

- Open and setup Visual Studio Code or Android Studio Code according to your convince and install the Flutter and Dart plugins/extensions in order to create a new flutter project
- A default basic flutter application will created and the necessary files and package are included by VS code its as default.

3.2. Including required dependencies at pubspec.yml

- Add `animated_splash_screen: ^1.1.0`
For create and work with SplashScreen page.
- Add `flutter_slidable: ^0.5.7`

A Flutter implementation of slidable list item with directional slide actions that can be dismissed.

- Add assets and paste the relative path of the images folder which will be having the images necessary for our project.
- `flutter_launcher_name: “^0.0.1”`

A command-line tool which simplifies the task of updating yor flutter app’s launcher name. If you use this package, you don’t need modify AndroidManifest.xml (for android) or info.plist(ios), by adding,

`Flutter_launche_name`
Name: “ToDo List”

After setting up the configuration, Run the package. `firebase_core: ^0.5.2+1`

After adding dependencies run this command `flutter pub`
`add firebase_core`

A flutter plugin to use the Firebase Core API, which enables connecting to multiple Firebase apps.

- cloud_firestore: ^0.14.3+1 After adding it run

this command

Flutter pub add cloud_firestore

A flutter plugin to use the Cloud Firestore API.

Import all necessary package and create necessary pages, widgets and providers for our app

3.3 FLOWCHART

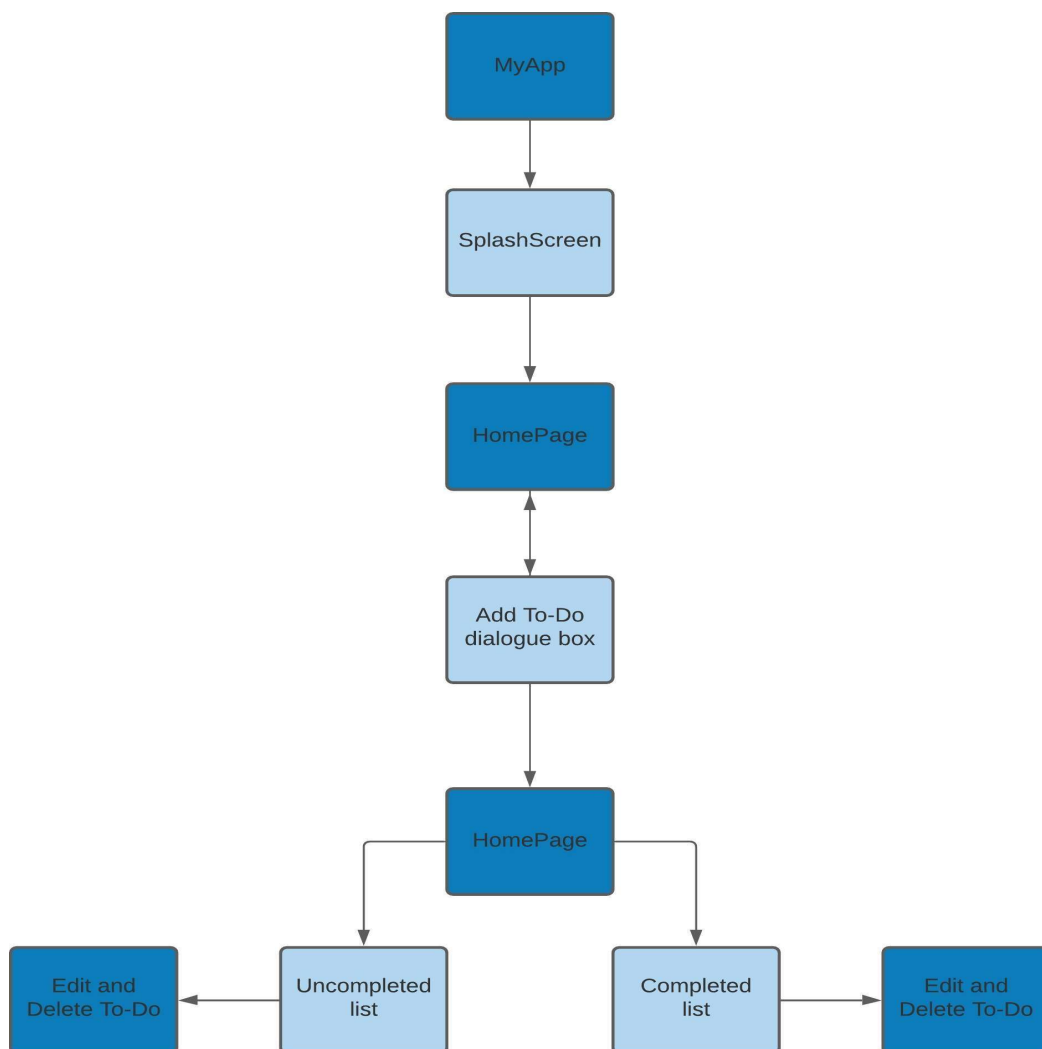


Fig 1:Architecture diagram

3.4 COMPONENTS AND WIDGETS USED

3.4.1 Components

Creating a Flutter project named `ToDo List Application`, in our application `Splash Screen` will be the first screen after splash screen it navigates to the `Home Page`.

`Home` consists of an `AppBar`, `bottomNavigationBar` with two icons (`todos`, `completed`) on pressing it navigates accordingly, and a `floatingActionButton` with an `add` icon on it. On pressing `add` a `ToDo` dialog box appears to create a new task.

- `android` - Auto generated source code to create android application
- `assets` – contains image folder for the splash screen
- `build` – When you run a flutter project, it builds depending on which emulator or device it is running on, doing a `Gradle` or `XCode` build using the folders inside it. In short, those folders are entire apps which set the stage for the flutter code to run
- `ios` - Auto generated source code to create ios application
- `lib` - Main folder containing Dart code written using flutter framework
- `lib/main.dart` - Entry point of the Flutter application
- `lib/pages` – contains dart code for `Splash Screen`, `Home Page` and `Edit to-do Page`
- `lib/widgets` – contains dart files for `AddToDoDialogWidget`
`ToDoFormWidget`
`ToDoWidget`
`ToDoListWidget`
`CompletedListWidget`
- `lib/provider` contains `todo.dart` file
- `lib/api` – contains dart file for `firebase`
- `gitignore` - Git version control file
- `metadata` - auto generated by the flutter tools
- `packages` - auto generated to track the flutter packages
- `pubspec.yaml` - Used by `Pub`, Flutter package manager
- `pubspec.lock` - Auto generated by the Flutter package manager, `Pub` • `README.md` - Project description file written in `Markdown` format.

3.4.2 Widgets

Stateless and stateful widgets:

To make our app look more like a todo app, let's show some tasks. You may have noticed that our simple app above is a `StatelessWidget`. This means that it cannot be changed dynamically. For our todo app, this is no good, because todo items get added and removed all the time. However, a `StatelessWidget` can have children which are dynamic, which are

StatefulWidgets. Let's break out our stateful functionality the todo list container from the overall app container.

To make a stateful widget, we need two classes — one for the widget itself, and a second which creates the state. This setup allows the state to be easily saved, and enables the use of features such as hot reloading.

- Widgets are high level objects used to describe any part of an application. It can be but is not limited to UI elements such as buttons ; layout alignment, padding;data a theme, configurations
- Widgets can be everything. In flutter, almost everything you'll code will be inside a Widget. Even things like "redux reducers" are widgets in flutter.
- The core concept of the Flutter framework is In Flutter, Everything is a widget. Widgets
- are basically user interface components used to create the user interface of the application.
- In Flutter, the application is itself a widget. The application is the top- level widget and its UI is build using one or more children widgets, which again build using its children
- widgets. This composability feature helps us to create a user interface of any complexity.

3.4.2.1. In main.dart

- runApp() Function takes the given Widget and makes it to the widget tree in main.drat the widget tree consists of a MaterialApp widget which describes theme ,primarySwatch and scaffoldBackgroundColor.

3.4.2.2. In home_page.dart

- Scaffold() – implements the baic material design visual layout structure.
This class provides APIs for showing deawers and bottom sheets.
- AppBar() – AppBar and Scaffold widgets, and from material.dart, the app is starting to look a bit more Material. For example, the app bar has a shadow and the title text inherits the correct styling automatically. AppBar widget let you pass in widgets for the leading widget, and the actions of the title widget.

- BottomNavigationBar class – A material widget that's displayed at the bottom of an app for selecting among a small number of views, typically between three and five.

The bottom navigation bar consists of multiple items in the form of text labels, icons, or both, laid out on top of a piece of material. It provides quick navigation may be a better fit.

A bottom navigation bar is usually used in conjunction with a scaffold, where it is provided as the Scaffold.bottomNavigationBar argument.

- FloatingActionButton class – A material design floating action button.

A floating action button is a circular icon button that hovers over content to promote a primary action in the application. Floating action buttons are most commonly used in the Scaffold.

onPressed callbacks to AddToDoDialogWidget()

CHAPTER – 4 RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

4.1 RESULTS

4.1.1. Homepage intro section

This consists of a ‘+’ icon to add todo tasks and a delete icon to delete the added tasks. It also displays the added tasks which are completed or yet to be completed.

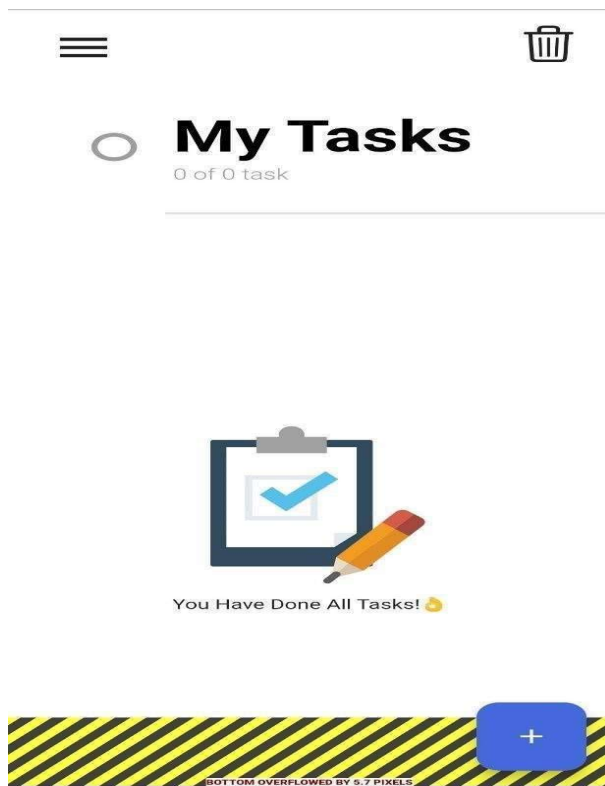


Fig 2:Home page

4.1.2. Add To-Do dialogue box intro section

This screen pops up when we press the add icon(+) floating action button on the homepage.

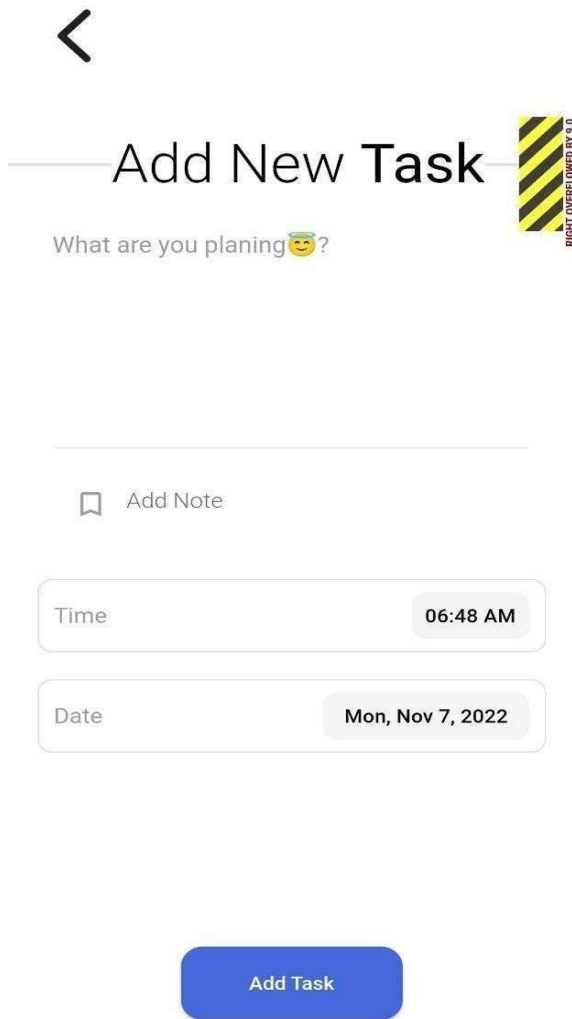


Fig 3: New task

4.1.3. Edit and delete options implemented using flutter_slidable package

To edit the task we need to tap on added task, it will lead to task editing page. Make the necessary changes and update the task.

To delete the task we need to tap on added task, it will lead to the task editing page where a delete option is available.

06:52

<

Update Task

What are you planing 😊?

buy milk

📅 today

Time 06:51 AM

Date Mon, Nov 7, 2022

✕ Delete Task Update Task

Fig 4:Task updation

4.1.4. To-Do list and Completed list

The new added tasks will added to 'My Tasks' area.Once the task is completed select the checklist;it will show the task as completed

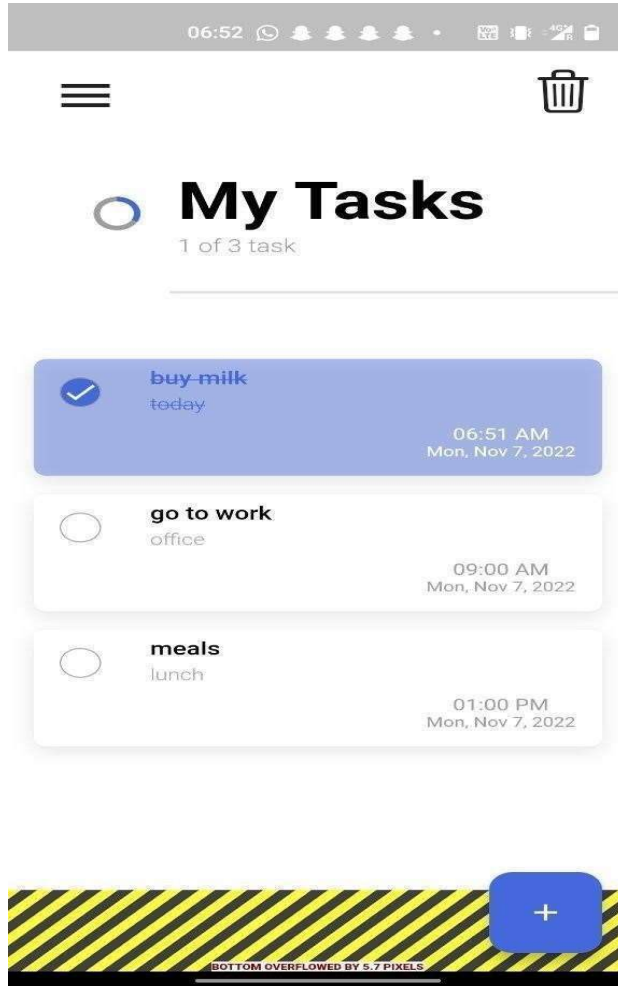


Fig 5:To do list

4.2. PERFORMANCE ANALYSIS

- The performance of the application is very efficient on latest android Os version 12
- The performance of the application is very efficient on latest mac version macos 12 monterey
- Internet connection is recommended
- Minimum RAM required is 2gb
- Due to efficient build strategy the app size is comparatively lower so the performance will be better.

CHAPTER – 5

SUMMARY AND CONCLUSION

5.1. SUMMARY

- This project is done based on flutter frame work using dart programming language with the help of Visual Studio Code IDE and mobile emulator / real time device .
- A to-do list is a form of task management where people compile tasks and determine what requires much attention.
- A to-do list helps us manage our workday
- A to-do list helps us become productive
- A use of firebase_core provides access to the firebase configuration and performs initialization of the native Firebase app
- A use of cloud firestore is a NoSQL document database that lets you easily store, sync, and query data for your mobile and web apps – at global scale

5.2. CONCLUSION

As we are leaving in a Fast growing and competitive world we need our life to be more systematic and disciplined in order to be successful One of the most important reasons you should use a to do list app is that it will help you stay organized. When you write all your tasks in a list, they seem more manageable. When you've got a clear outline of the tasks you've got to do and those you've completed, it helps you stay focused. In future , the application can be developed in schools and colleges and make sure every staffs and students do their duties or tasks on time

APPENDIX

SOURCE CODE main.dart

```
import 'package:flutter/material.dart'; import 'package:hive_flutter/hive_flutter.dart';

///                                import
'../data/hive_data_store.dart'; import '../models/task.dart';      import
'../view/home/home_view.dart';

Future<void> main() async { ///
  Initial Hive DB await
  Hive.initFlutter(); /// Register

  Hive Adapter

  Hive.registerAdapter<Task>(TaskAdapter());

  ///      Open      box      var      box      =      await
  Hive.openBox<Task>("tasksBox");

  /// Delete data from previous day
  // ignore: avoid_function_literals_in_foreach_calls box.values.forEach((task) { if
(task.createdAtTime.day != DateTime.now().day) {
task.delete(); } else {}
});

  runApp(BaseWidget(child: const MyApp()));
}

class BaseWidget extends InheritedWidget {
  BaseWidget({Key? key, required this.child}) : super(key: key, child: child); final HiveDataStore
dataStore = HiveDataStore(); final Widget child;

  static BaseWidget of(BuildContext context) { final base =
context.dependOnInheritedWidgetOfExactType<BaseWidget>(); if (base != null) { return base;
} else { throw StateError('Could not find ancestor widget of type
BaseWidget');
}
}

@override
```

```
bool updateShouldNotify(covariant InheritedWidget oldWidget) { return false;
} }
```

```
class MyApp extends StatelessWidget { const
  MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) { return
    MaterialApp( debugShowCheckedModeBanner:
      false, title: 'Flutter Hive Todo App', theme:
      ThemeData( textTheme: const TextTheme(
        headline1: TextStyle( color: Colors.black,
          fontSize: 45, fontWeight: FontWeight.bold, ),
        subtitle1: TextStyle( color: Colors.grey,
          fontSize:
            16,                      fontWeight:
            FontWeight.w300, ),
        headline2:
          TextStyle( color: Colors.white,
            fontSize: 21,
          ), headline3: TextStyle( color:
            Color.fromARGB(255, 234, 234, 234),
            fontSize: 14, fontWeight: FontWeight.w400,
          ),
        headline4: TextStyle( color:
          Colors.grey, fontSize: 17,
        ), headline5: TextStyle(
          color: Colors.grey, fontSize:
            16,
          ), subtitle2: TextStyle( color:
            Colors.black, fontWeight:
            FontWeight.w500,
          ), headline6:
            TextStyle( fontSize: 40,
```

```

        color: Colors.black,
        fontWeight: FontWeight.w300, ),
    ), ), home: const HomeView(),
  );
}
}

```

home_view.dart

```

// ignore_for_file: must_be_immutable

import 'package:animate_do/animate_do.dart'; import
'package:flutter/cupertino.dart'; import 'package:flutter/material.dart';
import
'package:hive_flutter/adapters.dart'; import
'package:lottie/lottie.dart';
import 'package:flutter_slider_drawer/flutter_slider_drawer.dart';
///      import '../main.dart';      import '../models/task.dart';      import '../utils/colors.dart';
      import '../utils/constanst.dart';      import
'../view/home/widgets/task_widget.dart'; import '../view/tasks/task_view.dart';      import
'../utils/strings.dart';

class HomeView extends StatefulWidget { const
HomeView({Key? key}) : super(key: key);
  @override
  // ignore: library_private_types_in_public_api
  _HomeViewState createState() => _HomeViewState(); }
class _HomeViewState extends State<HomeView> {
  GlobalKey<SliderDrawerState> dKey = GlobalKey<SliderDrawerState>();

  /// Checking Done Tasks  int checkDoneTask(List<Task>
task) { int i = 0; for (Task doneTasks in task) { if
(doneTasks.isCompleted) { i++;

```

```

    } } return i;
  }
  /// Checking The Value Of the Circle Indicator dynamic valueOfTheIndicator(List<Task> task) { if
  (task.isNotEmpty) { return task.length;
    } else { return 3;
  }
}

```

```

/// Floating Action Button class FAB extends
StatelessWidget { const FAB({Key?
key}) : super(key: key);
  @override
  Widget build(BuildContext context) { return
    GestureDetector( onTap: ()
    {
      Navigator.of(context).push( CupertinoPageRoute(
        builder: (context) => TaskView(
          taskControllerForSubtitle: null,
          taskControllerForTitle: null, task: null,
        ),
      ),
    ); }, child: Material( borderRadius:
    BorderRadius.circular(15), elevation:
    10, child: Container( width: 70, height:
    70,
    decoration: BoxDecoration( color:
    MyColors.primaryColor, borderRadius:
    BorderRadius.circular(15),
    ), child: const Center( child:
    Icon( Icons.add, color:
    Colors.white,
    )),

```

```

    ),
  ),
);
}
}

```

task_widget.dart

```

import 'package:flutter/cupertino.dart'; import
'package:flutter/material.dart'; import
'package:intl/intl.dart';
/// import
'../../models/task.dart'; import '../../utils/colors.dart';
import '../../view/tasks/task_view.dart';

class TaskWidget extends StatefulWidget { const TaskWidget({Key? key, required
this.task}) : super(key: key); final Task task;

  @override
  // ignore: library_private_types_in_public_api
  _TaskWidgetState createState() => _TaskWidgetState(); }

class _TaskWidgetState extends State<TaskWidget> {
  TextEditingController taskControllerForTitle = TextEditingController();
  TextEditingController taskControllerForSubtitle = TextEditingController();
  @override void initState() { super.initState(); taskControllerForTitle.text =
    widget.task.title; taskControllerForSubtitle.text = widget.task.subtitle; }

  @override void dispose() { taskControllerForTitle.dispose();
    taskControllerForSubtitle.dispose(); super.dispose();
  }

  @override
  Widget build(BuildContext context) { return
    GestureDetector(

```

```
onTap: () {
  Navigator.push( context,
    CupertinoPageRoute( builder: (ctx) => TaskView( taskControllerForTitle: taskControllerForTitle,
      taskControllerForSubtitle:
        taskControllerForSubtitle, task: widget.task, ),
    ),
  );
},
},
```

REFERENCES

- <https://flutter.dev/docs/reference/tutorials> □
- <https://pub.dev/flutter/packages> □
- <https://flutter.dev/docs/development/ui/advanced/splash-screen> □
- <https://www.section.io/engineering-education/how-to-build-a-flutter-todo-app/> □
- <https://daily-dev-tips.com/posts/build-a-todo-list-app-with-flutter/> □
- <https://mobikul.com/how-to-make-the-to-do-list-in-the-flutter/> □