TODO LIST APPLICATION

BY

MISS THANAWAN LHIMPATHANAWAT 6288027

MISS SUCHANUN SATANUNTIPHAT 6288154

ADVISOR

LECT. SNIT SANGHLAO

A Project Submitted in Partial Fulfillment of

the Requirements for

THE DEGREE OF BACHELOR OF SCIENCE

(INFORMATION AND COMMUNICATION TECHNOLOGY)

Faculty of Information and Communication Technology

Mahidol University

2021

CHAPTER 1

INTRODUCTION

The introduction part will introduce the motivation, problem, objectives, scope of the project as well as benefits from our application.

1.1 Motivation

Starting to develop the application nowadays was not as challenging as it was in the past. For someone who wishes to begin their programming experience, Flutter and Android Studio are a beneficial combination of free learning resources with many examples and projects. We chose to create the to-do list application to learn not only fundamental programming but also the UI/UX design that goes along with it.

1.2 Problem Statement

More than 80% of the human population have a smartphone of themselves, but how many people can remember the tasks that they have to do when the day is almost ending? A To-do list is one of the most reliable solutions to solve this problem for someone who doesn't have a decent memory whether you want the tasks to be done or memorize the tasks that you did.

1.3 Objectives of the Project

To make human life become easier and lighten the user's burden of remembering their tasks. It will help users become more productive and responsible for their tasks, also help boost the memory so that users will no longer forget the tasks that they have to do anymore.

1.4 Scope of the Project

The scope of our project comes from our focus on the current situation. Nowadays people live in urgent situations, so if there is a "Todo list Application" it will be a good help to users.

1.5 Expected Benefits

- This project will lead us to know how to implement flutter to create many useful applications.
- This application will help the user organize their task.
- This application will remind the user about lists that they need to do each day.
- This application will help users be more productive in their work.

1.6 Organization of the Document

This document consists of 6 chapters including:

1. **Introduction –** The introduction part will introduce the motivation, problem, objectives, scope of the project as well as benefits from our application.
2. **Background –** The background part will describe the reason for intending to do this project and the understanding of the project and flutter.
3. **Analysis and Design –** The analysis and design will show our structure and architecture then will explain the detail of the system in our application which can be able to visualize and understand in our application easily.
4. **Implementation –** The implementation will state our hardware and system environment, also explain the implementation guide and techniques of the system.
5. **Testing and Evaluation – TBA**
6. **Conclusion – TBA**

CHAPTER 2

BACKGROUND

The background part will describe the reason for intending to do this project and the understanding of the project and flutter.

2.1 Literature Review

The main idea of a To-Do list is to write the lists of tasks or things that need to be done and then sort them efficiently. This way of sorting and managing your tasks helps reduce the brain's work because your brain will remember and keep track of the tasks that are marked as undone. Once the task has been completed, it will free up your mind.

Nowadays, there are many applications that help you manage your tasks by using a To-Do list. It doesn't always require to be written by pen and paper, the To-Do list application allows you to tick and untick the tasks on your lists whether it is a chore, shopping list, tasks, or something else. It is one of the most effective productivity tools that people use nowadays.

Ref.

https://sea.pcmag.com/productivity/16954/the-best-to-do-list-apps-for-2020
https://fs.blog/the-psychology-of-the-to-do-list/

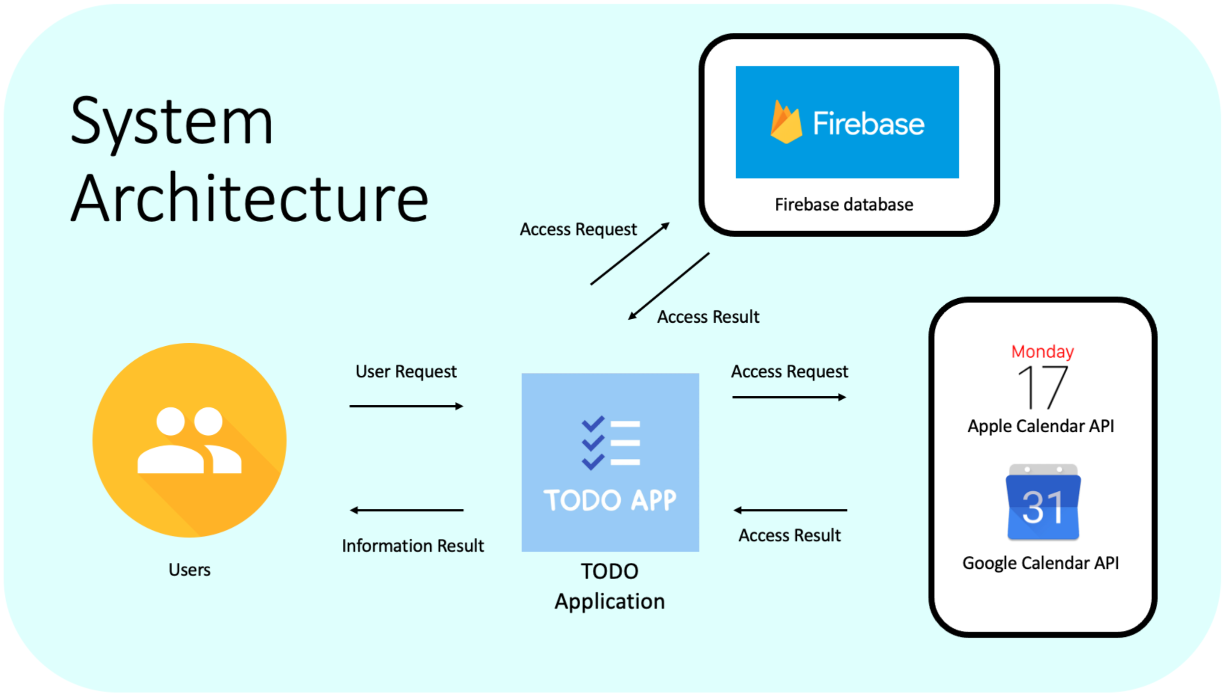
CHAPTER 3

ANALYSIS AND DESIGN

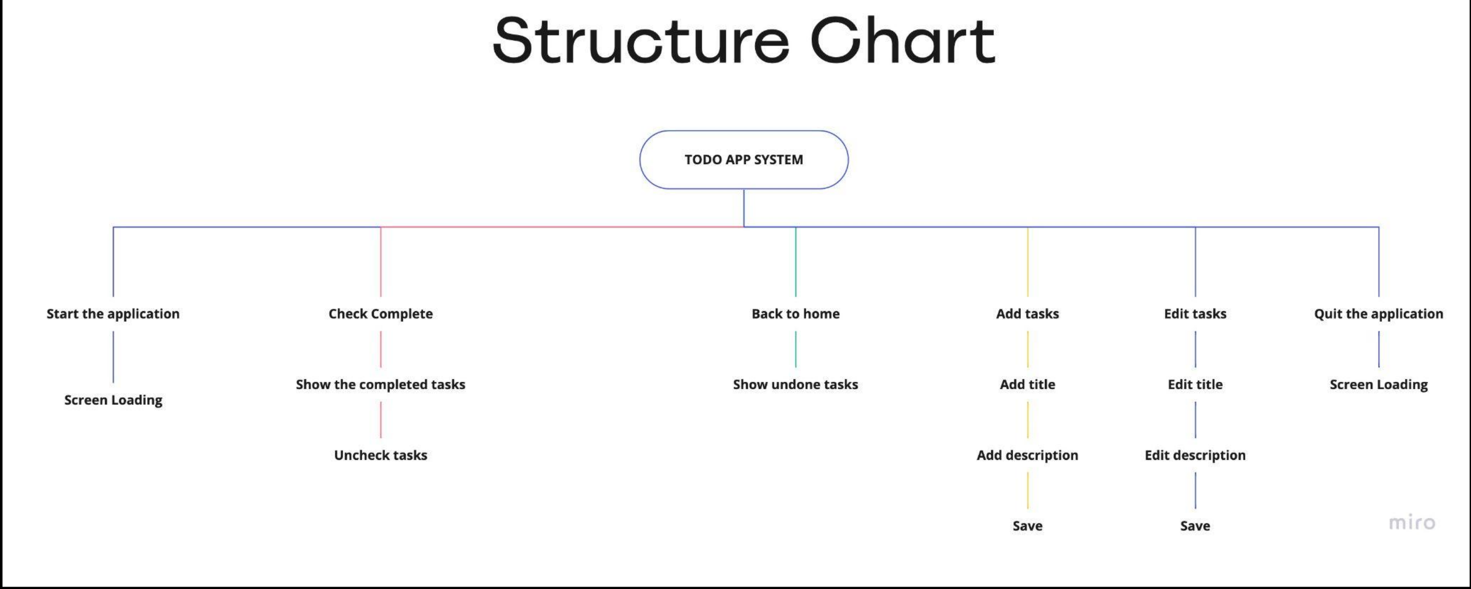
The analysis and design will show our structure and architecture then will explain the details of the system in our application which can be able to visualize and understand in our application easily.

3.1 System Architecture Overview

The architecture in TODO Application consists of the Firebase database and Calendar API, all of these will implement in our application to make it more complete.



3.2 System Structure Chart



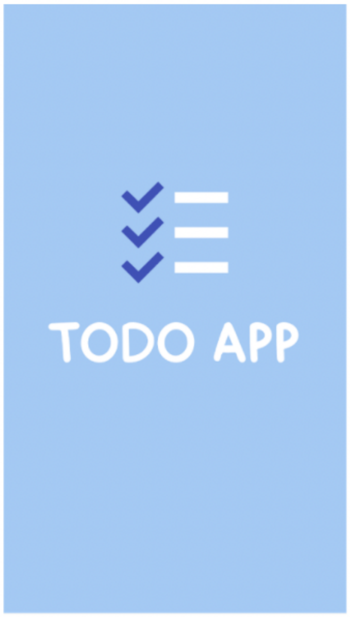
Project : Todo App using Flutter **System :** Flutter - System Structure Chart	**Major Advisor :** SNIT SANGHLAO
Description : The function of the system structure chart in the TODO Application is divided into six subsections: Start, Check, Back, Add, Edit, and Quit.	

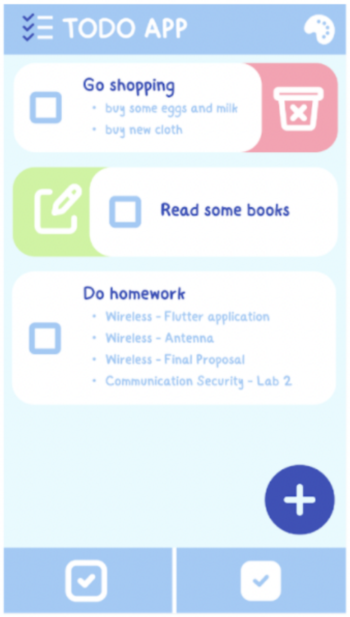
Figure 3.1: <The function of the system structure chart in the TODO Application is divided into six subsections: Start, Check, Back, Add, Edit, and Quit>

3.3 I/O Design

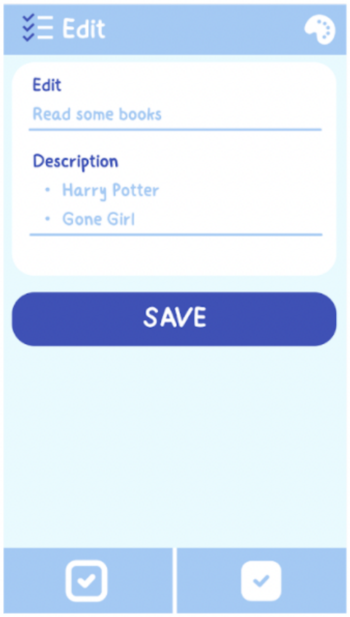
This section explains the design of the Input and Output User Interface. The section consists of two parts, the interface design and the transition diagram showing transition through the system.

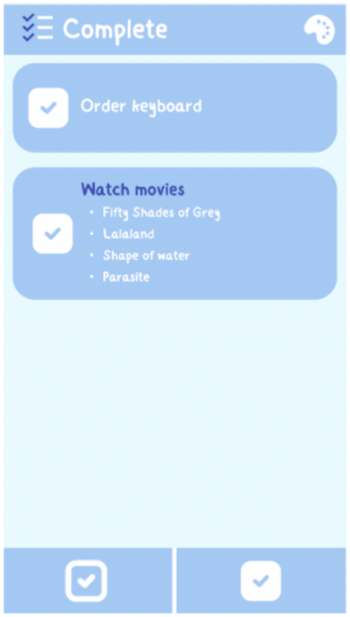
3.3.1 Interface Design

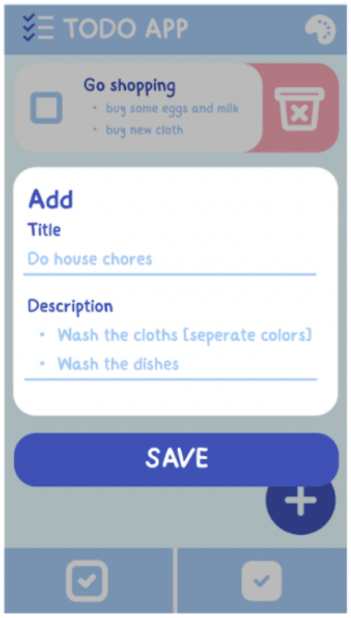




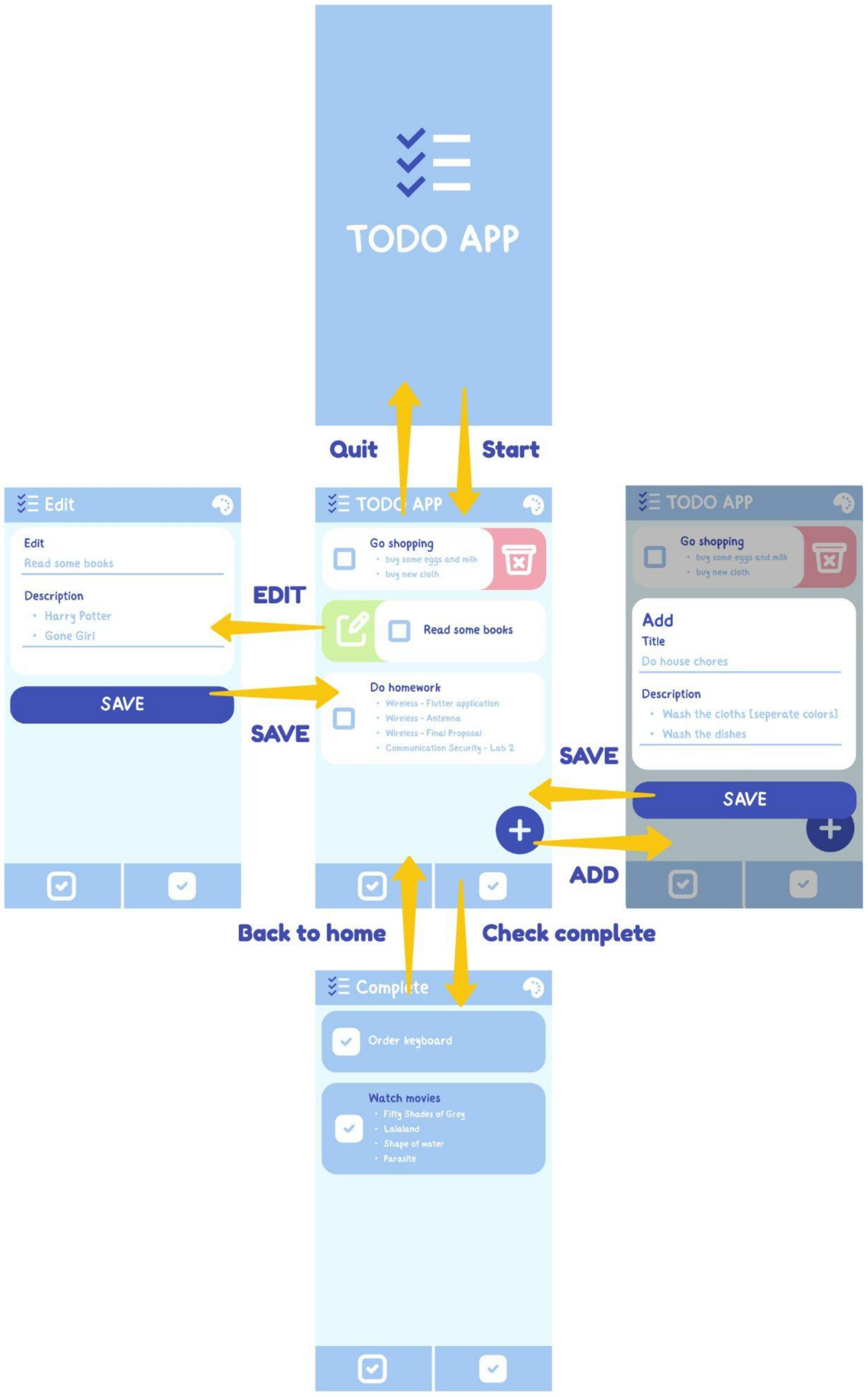
↑ LOGO ↓ EDIT COMPLETE ↓ ↑ HOME ADD ↓







3.5.2 Transition Diagram



CHAPTER 4

IMPLEMENTATION

The implementation will state our hardware and system environment, also explain the implementation guide and techniques of the system.

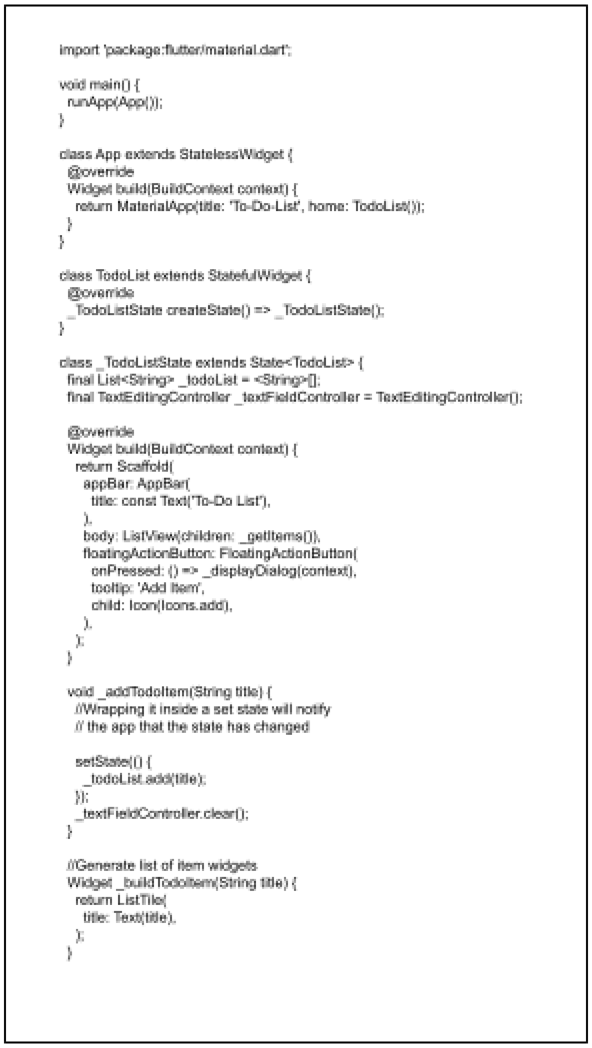
4.1 Hardware and System Environment

- Operating System and Utilities Applications
  - **Application Launchers:** To check the application whether contains bugs or not and how to adjust the UX/UI also associated with the variety of access via other platforms.
  - **Backup Utility:** To backup the files or caches that use to utilized in Todo Application.
- Web Server Software
  - **Apple Calendar API:** To be used to retrieve date information for use in the notification function of the TODO Application for the IOS operating system.
  - **Google Calendar API:** To be used to retrieve date information for use in the notification function of the TODO Application for the Android operating system.
- Editor
  - **Android studio:** Main programming software to create a TODO application structure. being used to create the program, test the bug and check UX/UI.
- Database Management System (DBMS)
  - **Firebase:** To store further information about the Todo Application and keep user data like to-do lists, completed tasks, and others.
- Programming and Scripting Tools
  - **VS Code:** Deeper editing about CSS/ Javascript for the UX/UI.

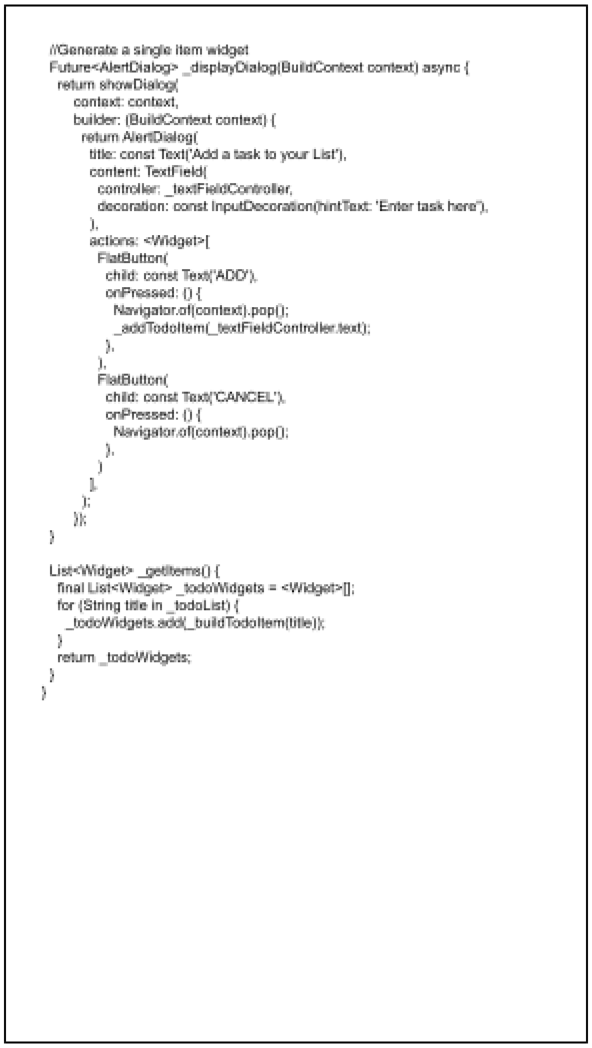
4.2 Implementation Guide and Techniques

4.2.1 <Guide/Technique/Know-how>

How to Build a Flutter ToDo App by section.io. In this tutorial, we'll use Flutter to create a to-do list app. To best comprehend this article, the reader should have a basic understanding of object-oriented programming languages such as Java, C++, and others.



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

void main() {
  runApp(App());
}

class App extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(title: 'To-Do-List', home: TodoList());
  }
}

class TodoList extends StatefulWidget {
  @override
  _TodoListState createState() => _TodoListState();
}

class _TodoListState extends State<TodoList> {
  final List<String> _todoList = <String>[];
  final TextEditingController _textFieldController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('To-Do List'),
      ),
      body: ListView(children: _getItems()),
      floatingActionButton: FloatingActionButton(
        onPressed: () => _displayDialog(context),
        tooltip: 'Add Item',
        child: Icon(Icons.add),
      ),
    );
  }

  void _addTodoItem(String title) {
    //Wrapping it inside a set state will notify
    // the app that the state has changed

    setState(() {
      _todoList.add(title);
    });
    _textFieldController.clear();
  }

  //Generate list of item widgets
  Widget _buildTodoItem(String title) {
    return ListTile(
      title: Text(title),
    );
  }
```



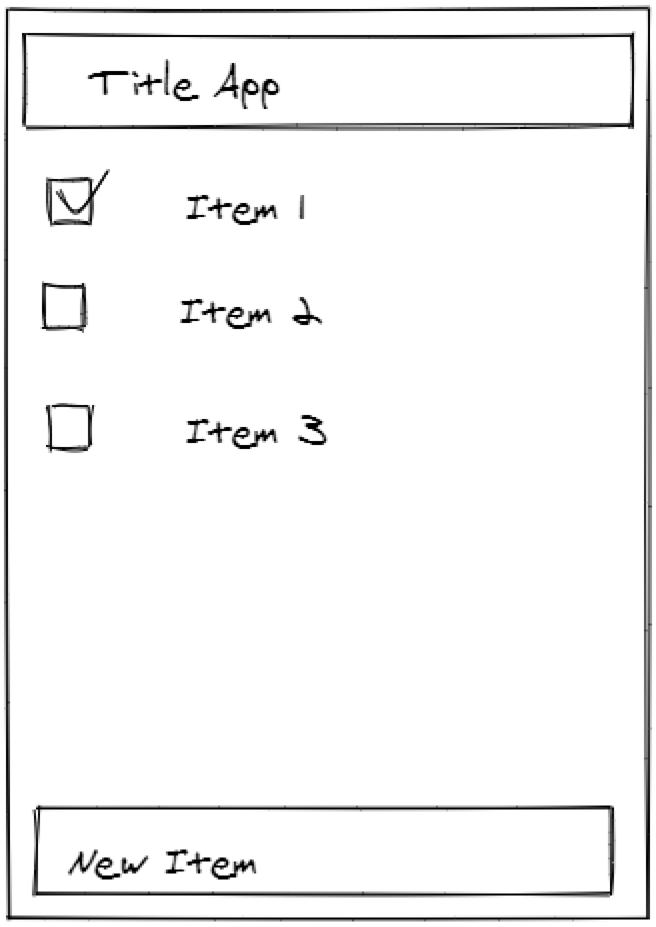
```
  //Generate a single item widget
  Future<AlertDialog> _displayDialog(BuildContext context) async {
    return showDialog(
        context: context,
        builder: (BuildContext context) {
          return AlertDialog(
            title: const Text('Add a task to your List'),
            content: TextField(
              controller: _textFieldController,
              decoration: const InputDecoration(hintText: 'Enter task here'),
            ),
            actions: <Widget>[
              FlatButton(
                child: const Text('ADD'),
                onPressed: () {
                  Navigator.of(context).pop();
                  _addTodoItem(_textFieldController.text);
                },
              ),
              FlatButton(
                child: const Text('CANCEL'),
                onPressed: () {
                  Navigator.of(context).pop();
                },
              )
            ],
          );
        });
  }

  List<Widget> _getItems() {
    final List<Widget> _todoWidgets = <Widget>[];
    for (String title in _todoList) {
      _todoWidgets.add(_buildTodoItem(title));
    }
    return _todoWidgets;
  }
}
```

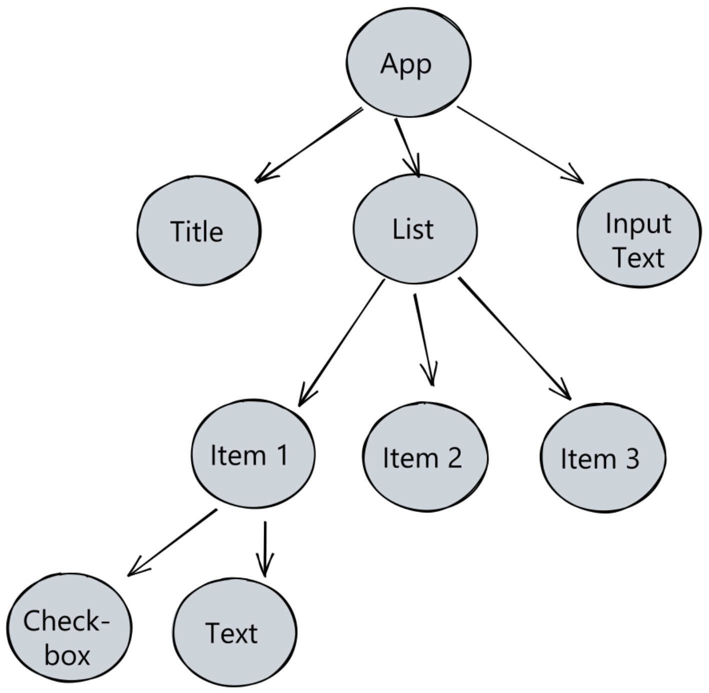
https://github.com/wobin1/Todo-List-app/blob/master/lib/main.dart

4.2.2 < Guide/Technique/Know-how>

The Provider class is used to manage the application's states. When a state change happens, we may utilize providers to notify the application's user interface. Flutter generates the user interface using widgets. Widgets allow you to control how and what appears on the screen.

Wireframe:

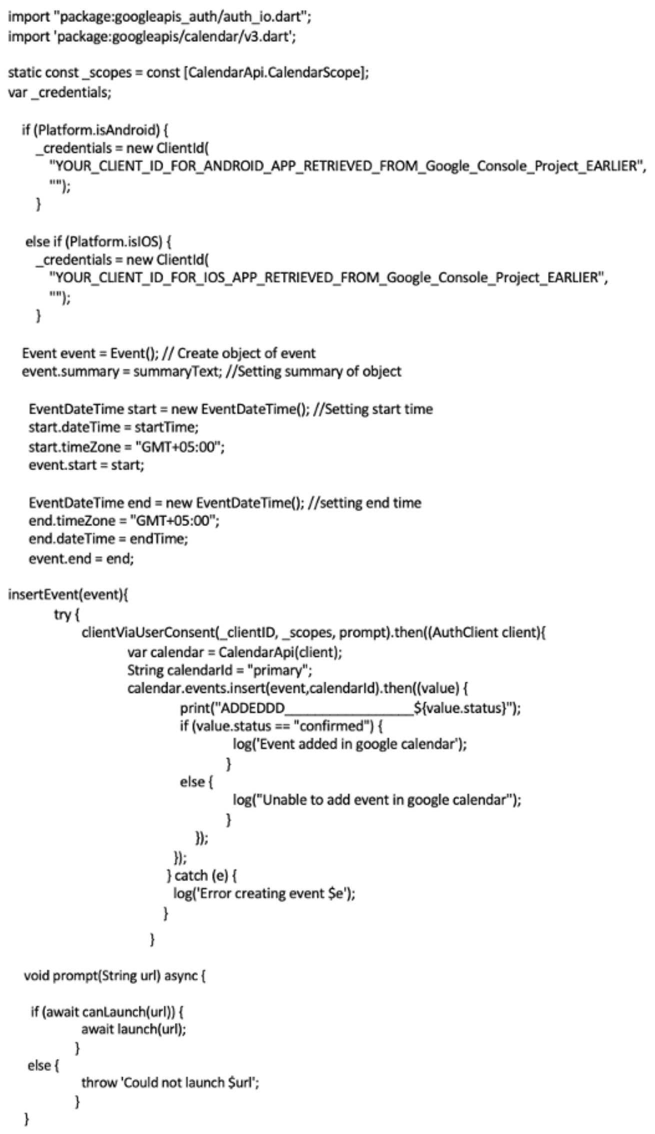




https://github.com/dwyl/flutter-todo-list-tutorial

4.2.3 < Guide/Technique/Know-how>

To create events in Google Calendar through the Flutter app, to implement Google Calendar API to our TODO application, the code has four parts are Create Scopes, Creating ClientID Object, Event Object, and Inserting the Event.



```
import "package:googleapis_auth/auth_io.dart";
import 'package:googleapis/calendar/v3.dart';

static const _scopes = const [CalendarApi.CalendarScope];
var _credentials;

  if (Platform.isAndroid) {
    _credentials = new ClientId(
      "YOUR_CLIENT_ID_FOR_ANDROID_APP_RETRIEVED_FROM_Google_Console_Project_EARLIER",
      "");
  }

  else if (Platform.isIOS) {
    _credentials = new ClientId(
      "YOUR_CLIENT_ID_FOR_IOS_APP_RETRIEVED_FROM_Google_Console_Project_EARLIER",
      "");
  }

  Event event = Event(); // Create object of event
  event.summary = summaryText; //Setting summary of object

   EventDateTime start = new EventDateTime(); //Setting start time
   start.dateTime = startTime;
   start.timeZone = "GMT+05:00";
   event.start = start;

   EventDateTime end = new EventDateTime(); //setting end time
   end.timeZone = "GMT+05:00";
   end.dateTime = endTime;
   event.end = end;

insertEvent(event){
      try {
          clientViaUserConsent(_clientID, _scopes, prompt).then((AuthClient client){
                var calendar = CalendarApi(client);
                String calendarId = "primary";
                calendar.events.insert(event,calendarId).then((value) {
                      print("ADDEDDD_________________${value.status}");
                      if (value.status == "confirmed") {
                            log('Event added in google calendar');
                           }
                      else {
                            log("Unable to add event in google calendar");
                           }
                        });
                     });
                   } catch (e) {
                    log('Error creating event $e');
                   }

                 }

  void prompt(String url) async {

   if (await canLaunch(url)) {
          await launch(url);
        }
   else {
          throw 'Could not launch $url';
        }
  }
```

https://medium.com/flutter-community/flutter-use-google-calendar-api-adding-the-events-to-calendar-3d8fcb008493