VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018

An Internship Project Report

on

To Do List using Flutter

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

By

Mehul Gupta

1RN18IS066

Under the Guidance of

Dr. R Rajkumar

Associate Professor

Department of ISE

Department of Information Science & 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 ***To Do List*** has been successfully completed by **Mehul Gupta (1RN18IS066)** bonafide students of **RNS Institute of Technology, Bengaluru** in partial fulfillment of the requirements of 8th 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. T S Bhagavath Singh	**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.	**1.**
2.	**2.**

DECLARATION

I, **Mehul Gupta [USN: 1RN18IS066]** students of VIII Semester BE, in Information Science and Engineering, RNS Institute of Technology hereby declare that the Internship work entitled ***To Do List*** 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 **Mehul Gupta**

Date : **(1RN18IS066)**

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.

Also, constantly switching between tasks may give us the false feeling that we are being productive when we are, in fact, not. It’s more important for us to prioritize tasks and work on those that are most important, rather than focusing on deleting small items from our todo list just for the sake of appearances.

The goal of this app is to help us manage all our tasks at one place so that we are able to manage our daily routines more efficiently. We will have a better sense of the estimated time we’ll need to spend on our tasks and will be able to manage our routine according to the various tasks at hand.

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 Mr. T S Bhagavath Singh Associate 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, ENMAZ, 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.

Mehul Gupta

1RN18IS066

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 CanvasKit 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 TaskDo: A Daily Task Recommender System

Many individuals like working professionals, students, and house makers often find lack of time and time management as problems forsuccessful task accomplishment.One of the key reasons for failure in task accomplishment is inefficient planning of the tasks. There are many task management and to-do-list applications, but most of them do not advise on optimal task management and guidance for optimal performance. This problem has driven us to contribute a task recommender system which suggests a specific type of tasks to users based on their history of tasks and various factors at that specific time. This system not only suggests a specific type of task for the user but also collects feedback from the user to make the recommender system learn on how to provide useful recommendations thus making the users time much productive.

2.1.1 Introduction

Striving to be productive remains a challenge for many workers, professionals and students alike. Various researches and surveys state that 70% of employees work beyond scheduled time and on weekends; more than half cited "self-imposed pressure" as the reason [1]. The survey, conducted by Greenfield Online [2], found that nearly half of college students (47 percent) feel their high school did not prepare them with the organizational skills required to do well in college and 54 percent felt they would get better grades if they "got organized and stayed organized."To recover from these sorts of problems, spending time organizing daily tasks would be helpful. Research says thatfor every hour of planning, 3 to 4 hours are saved from redundancy, waiting for information, not being prepared and poorly managed tasks [3]. With new age advancements in information technology, we have many applications which focus on time and task

management. Task and time management tools such as Todoist [4] and Wunderlist [5] allow users to add and track tasks. However, these tools do not have a mechanism for suggesting tasks to be done at a specific time. Hence, users are left on their own to plan their daily schedule. Further, these tools have implemented some visual analytics to help

users to understand their productivity, and how it changes over time. Driven by the importance of time and task management, and lack of the tools which suggest a specific task for a specific time, we are inspired by the research question: “Based on the user’s history of completed tasks, can we recommend certain task types to be done at certain days of The week and times of day to increase a user’s productivity?”. To answer this question, we contribute TaskDo, a task recommender system which suggests a specific task to be done on certain days of the week and/or at certain times of the day. In generating the recommendations, the system relies upon the history of a user’s task completion. For example, since the system has observed that Adam tends to complete his chores at a shorter time on weekends in the morning, the system will recommend to Adam that he does his chores in the morning on weekends. This is a simple case, but it illustrates the point of the system learning about user’s habits of task completion and suggesting what is believed to best the best time for a specific type of task. Our initial thought is that the recommendations of task types will be affected by many variables such as day of the week, time of day, whether the task is done indoors or outdoors, whether the task is intellectual or physical, and so on.

2.1.2 Related work

2.1.2.1 Task management application

Many task and time management tools allow users to manage their tasks and times. These tools operate based on the simple idea of allowing users to add tasks, set due dates to them, and complete them when they can. However, these tools do not assist users in planning their daily or weekly schedule with tasks that users may be able to complete to increase productivity. There are some popular task management applications such as Wunderlist [5] which focuses on the organization of tasks into folders, hosting communication for tasks which involve multiple people, adding remainders, setting due dates and related notifications and so on. There is one more popular application called Todoist [4] which has most of the functions of Wunderlist, and it has additional features like organizing tasks into separate projects, and tracking user’s productivity with help of various types of visual analytics which demonstrate task progress, taskcompletion and pending tasks with help of bar graphs, pie charts and so on. However, these tools do not recommend users suitable type of tasks for a specific day and time. Timeful [6] was an application that aimed at understanding users’ habits and schedules by asking users how

often and when they want to do things. The system then assists users with planning their schedule. The idea seems promising, but the application is currently unavailable, which makes it hard to evaluate the accuracy of the application

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
Backend Tool	:	SQLite

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, Swift, SQLite
Backend Tool	:	SQLite

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.

SQLite

SQLite is a self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine. It is the most used database engine in the world. It is an in-process library and its code is publicly available. It is free for use for any purpose, commercial or private. It is basically an embedded SQL database engine. The SQLite database file format is cross-platform so that anyone can easily copy a database between 32-bit and 64- bit systems. Due to all these features, It is a popular choice as an Application File Format.

Chapter 4

SYSTEM DESIGN

4.1 Home page widget tree:

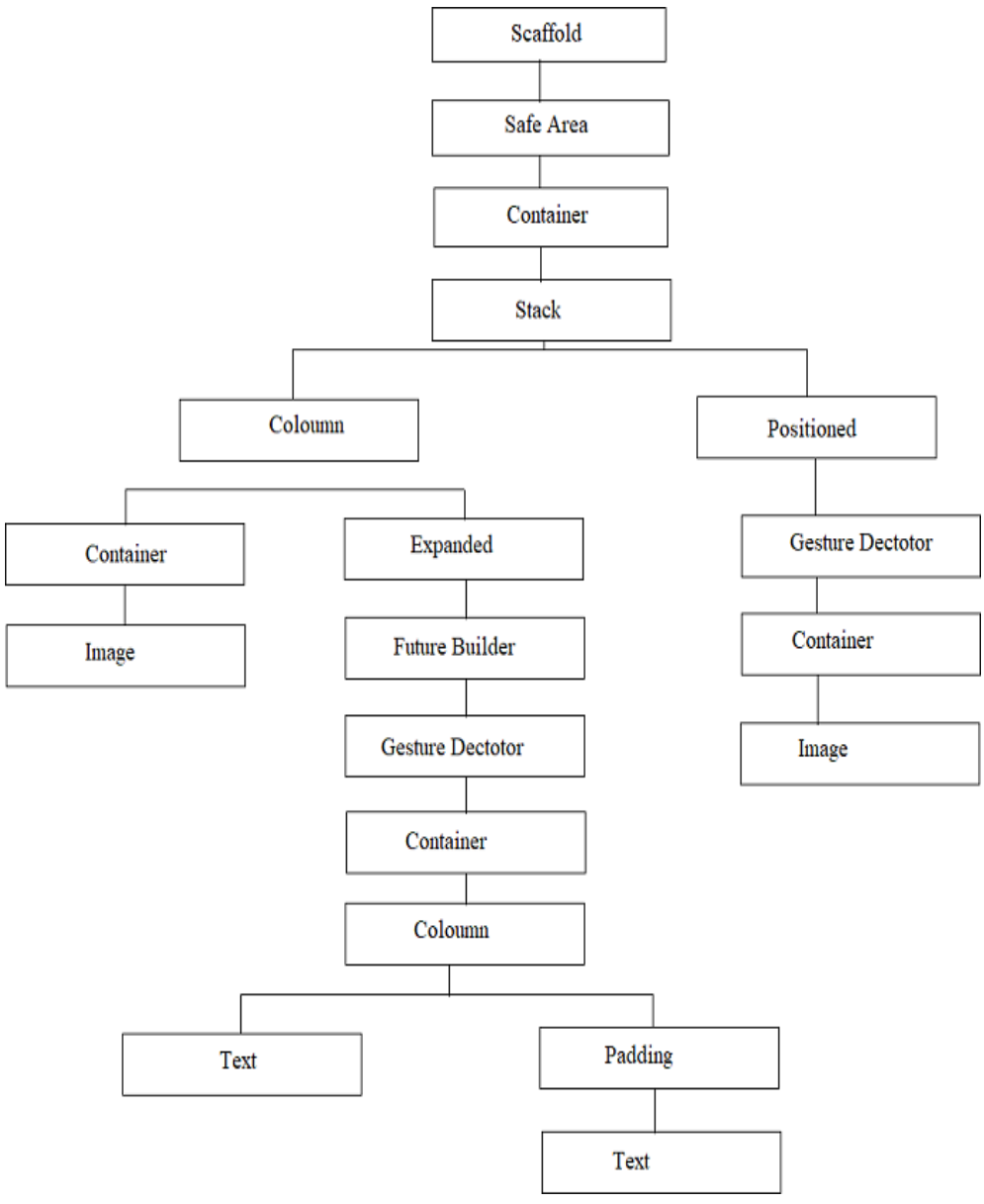

Fig 4.1 HomePage Widget Tree

4.2 To Do list page widget tree:

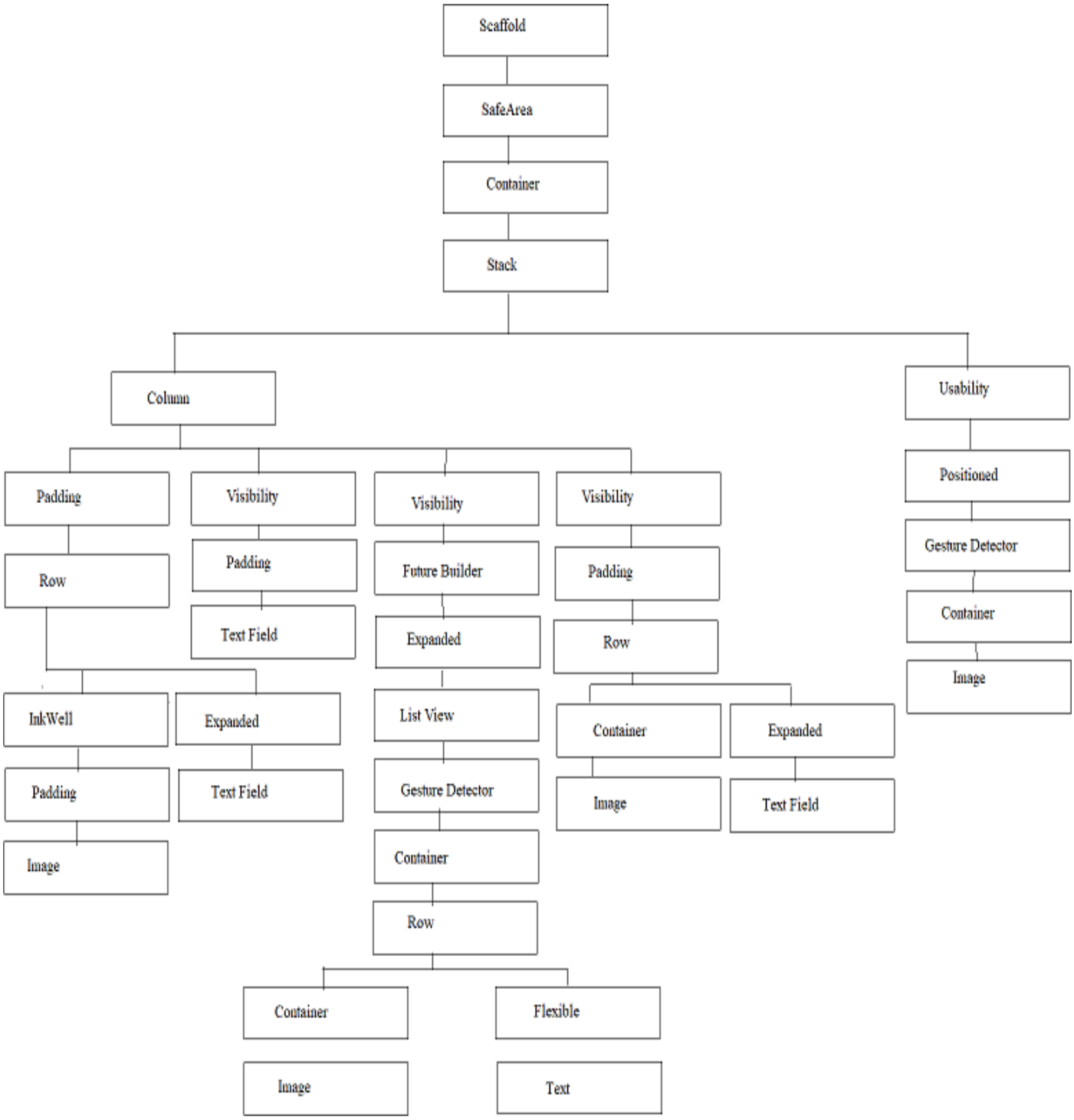

Fig 4.2 To Do List Page Widget Tree

4.3 Database Schema design:

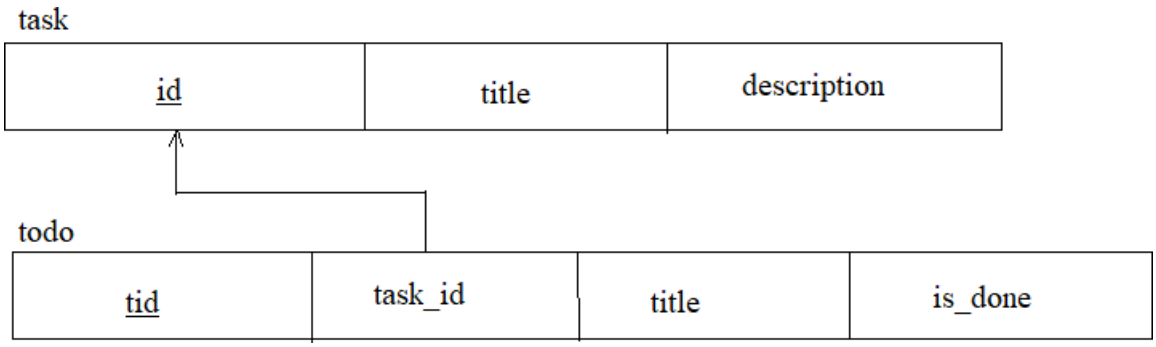

Fig 4.3 Database schema

Chapter 5

IMPLEMENTATION DETAILS

5.1 main.dart

```dart
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:what_todo/screens/homepage.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        textTheme: GoogleFonts.nunitoSansTextTheme(
          Theme.of(context).textTheme,
        ),
      ),
      home: Homepage(),
    );
  }
}
```

5.2 homepage.dart

```
import 'package:flutter/material.dart';
import 'package:what_todo/database_helper.dart';
import 'package:what_todo/screens/taskpage.dart';
import 'package:what_todo/widgets.dart';

class Homepage extends StatefulWidget {
  @override
  _HomepageState createState() => _HomepageState();
}

class _HomepageState extends State<Homepage> {
  DatabaseHelper _dbHelper = DatabaseHelper();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Container(
          width: double.infinity,
          padding: EdgeInsets.symmetric(horizontal: 24.0),
          color: Color(0xFFF6F6F6),
          child: Stack(
            children: [
              Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
```

```
                Container(
                  margin: EdgeInsets.only(
                    top: 32.0,
                    bottom: 32.0,
                  ),
                  child: Image(
                    image: AssetImage("assets/images/logo.png"),
                  ),
                ),
                Expanded(
                  child: FutureBuilder(
                    initialData: [],
                    future: _dbHelper.getTasks(),
                    builder: (context, snapshot) {
                      return ScrollConfiguration(
                        behavior: NoGlowBehaviour(),
                        child: ListView.builder(
                          itemCount: snapshot.data.length,
                          itemBuilder: (context, index) {
                            return GestureDetector(
                              onTap: () {
                                Navigator.push(
                                  context,
                                  MaterialPageRoute(
                                    builder: (context) => Taskpage(
                                      task: snapshot.data[index],
                                    ),
                                  ),
                                ).then(
```

```
                        (value) {
                          setState(() {});
                        },
                      );
                    },
                    child: TaskCardWidget(
                      title: snapshot.data[index].title,
                      desc: snapshot.data[index].description,
                    ),
                  );
                },
              ),
            );
          },
        ),
      )
    ],
  ),
  Positioned(
    bottom: 24.0,
    right: 0.0,
    child: GestureDetector(
      onTap: () {
        Navigator.push(
          context,
          MaterialPageRoute(
              builder: (context) => Taskpage(
                  task: null,
                )),
```

```
                ).then((value) {
                  setState(() {});
                });
              },
              child: Container(
                width: 60.0,
                height: 60.0,
                decoration: BoxDecoration(
                  gradient: LinearGradient(
                      colors: [Color(0xFF7349FE), Color(0xFF643FDB)],
                      begin: Alignment(0.0, -1.0),
                      end: Alignment(0.0, 1.0)),
                  borderRadius: BorderRadius.circular(20.0),
                ),
                child: Image(
                  image: AssetImage(
                    "assets/images/add_icon.png",
                  ),
                ),
              ),
            ),
          )
        ],
      ),
    ),
  ),
);
  }
}
```

5.3 taskPage.dart

```dart
import 'package:flutter/material.dart';
import 'package:what_todo/database_helper.dart';
import 'package:what_todo/models/task.dart';
import 'package:what_todo/models/todo.dart';
import 'package:what_todo/widgets.dart';

class Taskpage extends StatefulWidget {
  final Task task;

  Taskpage({@required this.task});

  @override
  _TaskpageState createState() => _TaskpageState();
}

class _TaskpageState extends State<Taskpage> {
  DatabaseHelper _dbHelper = DatabaseHelper();

  int _taskId = 0;
  String _taskTitle = "";
  String _taskDescription = "";

  FocusNode _titleFocus;
  FocusNode _descriptionFocus;
  FocusNode _todoFocus;
```

```
bool _contentVisile = false;


@override
void initState() {
  if (widget.task != null) {
    // Set visibility to true
    _contentVisile = true;


    _taskTitle = widget.task.title;
    _taskDescription = widget.task.description;
    _taskId = widget.task.id;
  }


  _titleFocus = FocusNode();
  _descriptionFocus = FocusNode();
  _todoFocus = FocusNode();


  super.initState();
}


@override
void dispose() {
  _titleFocus.dispose();
  _descriptionFocus.dispose();
  _todoFocus.dispose();


  super.dispose();
}
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: SafeArea(
      child: Container(
        child: Stack(
          children: [
            Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Padding(
                  padding: EdgeInsets.only(
                    top: 24.0,
                    bottom: 6.0,
                  ),
                  child: Row(
                    children: [
                      InkWell(
                        onTap: () {
                          Navigator.pop(context);
                        },
                        child: Padding(
                          padding: const EdgeInsets.all(24.0),
                          child: Image(
                            image: AssetImage(
                                'assets/images/back_arrow_icon.png'),
                          ),
                        ),
                      ),
```

```
                        Expanded(
                          child: TextField(
                            focusNode: _titleFocus,
                            onSubmitted: (value) async {
                              // Check if the field is not empty
                              if (value != "") {
                                // Check if the task is null
                                if (widget.task == null) {
                                  Task _newTask = Task(title: value);
                                  _taskId = await
_dbHelper.insertTask(_newTask);
                                  setState(() {
                                    _contentVisile = true;
                                    _taskTitle = value;
                                  });
                                } else {
                                  await _dbHelper.updateTaskTitle(_taskId,
value);
                                  print("Task Updated");
                                }
                                _descriptionFocus.requestFocus();
                              }
                            },
                            controller: TextEditingController()
                              ..text = _taskTitle,
                            decoration: InputDecoration(
                              hintText: "Enter Task Title",
                              border: InputBorder.none,
                            ),
```

```
                        style: TextStyle(
                          fontSize: 26.0,
                          fontWeight: FontWeight.bold,
                          color: Color(0xFF211551),
                        ),
                      ),
                    )
                  ],
                ),
              ),
              Visibility(
                visible: _contentVisile,
                child: Padding(
                  padding: EdgeInsets.only(
                    bottom: 12.0,
                  ),
                  child: TextField(
                    focusNode: _descriptionFocus,
                    onSubmitted: (value) async {
                      if(value != ""){
                        if(_taskId != 0){
                          await
_dbHelper.updateTaskDescription(_taskId, value);
                          _taskDescription = value;
                        }
                      }
                      _todoFocus.requestFocus();
                    },
```

```
                    controller: TextEditingController()..text =
_taskDescription,

                    decoration: InputDecoration(
                      hintText: "Enter Description for the task...",
                      border: InputBorder.none,
                      contentPadding: EdgeInsets.symmetric(
                        horizontal: 24.0,
                      ),
                    ),
                  ),
                ),
              ),
              Visibility(
                visible: _contentVisile,
                child: FutureBuilder(
                  initialData: [],
                  future: _dbHelper.getTodo(_taskId),
                  builder: (context, snapshot) {
                    return Expanded(
                      child: ListView.builder(
                        itemCount: snapshot.data.length,
                        itemBuilder: (context, index) {
                          return GestureDetector(
                            onTap: () async {
                              if(snapshot.data[index].isDone == 0){
                                await
_dbHelper.updateTodoDone(snapshot.data[index].id, 1);
                              } else {
                                await
_dbHelper.updateTodoDone(snapshot.data[index].id, 0);
```

```
                    }
                    setState(() {});
                  },
                  child: TodoWidget(
                    text: snapshot.data[index].title,
                    isDone: snapshot.data[index].isDone == 0
                        ? false
                        : true,
                  ),
                );
              },
            ),
          );
        },
      ),
    ),
    Visibility(
      visible: _contentVisile,
      child: Padding(
        padding: EdgeInsets.symmetric(
          horizontal: 24.0,
        ),
        child: Row(
          children: [
            Container(
              width: 20.0,
              height: 20.0,
              margin: EdgeInsets.only(
                right: 12.0,
```

```
                            ),
                            decoration: BoxDecoration(
                                color: Colors.transparent,
                                borderRadius: BorderRadius.circular(6.0),
                                border: Border.all(
                                    color: Color(0xFF86829D), width:
1.5)),
                            child: Image(
                              image:
AssetImage('assets/images/check_icon.png'),
                            ),
                          ),
                          Expanded(
                            child: TextField(
                              focusNode: _todoFocus,
                              controller: TextEditingController()..text =
"",
                              onSubmitted: (value) async {
                                // Check if the field is not empty
                                if (value != "") {
                                  if (_taskId != 0) {
                                    DatabaseHelper _dbHelper =
DatabaseHelper();
                                    Todo _newTodo = Todo(
                                      title: value,
                                      isDone: 0,
                                        taskId: _taskId,
                                    );
                                    await _dbHelper.insertTodo(_newTodo);
                                    setState(() {});
```

```
                        _todoFocus.requestFocus();
                      } else {
                        print("Task doesn't exist");
                      }
                    }
                  },
                  decoration: InputDecoration(
                    hintText: "Enter Todo item...",
                    border: InputBorder.none,
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
    ],
  ),
  Visibility(
    visible: _contentVisile,
    child: Positioned(
      bottom: 24.0,
      right: 24.0,
      child: GestureDetector(
        onTap: () async {
          if(_taskId != 0) {
            await _dbHelper.deleteTask(_taskId);
            Navigator.pop(context);
          }
```

```
                },
                child: Container(
                  width: 60.0,
                  height: 60.0,
                  decoration: BoxDecoration(
                    color: Color(0xFFFE3577),
                    borderRadius: BorderRadius.circular(20.0),
                  ),
                  child: Image( image:
                    AssetImage(
                      "assets/images/delete_icon.png",
                    ),
                  ),
                ),
              ),
            )
          ],
        ),
      ),
    ),
  );
```

5.4 database_helper.dart

```
import 'package:path/path.dart';
import 'package:sqflite/sqflite.dart';


import 'models/task.dart';
import 'models/todo.dart';
```

```
class DatabaseHelper {


  Future<Database> database() async {

    return openDatabase(

      join(await getDatabasesPath(), 'todo.db'),

      onCreate: (db, version) async {

        await db.execute("CREATE TABLE tasks(id INTEGER PRIMARY KEY, title
TEXT, description TEXT)");

        await db.execute("CREATE TABLE todo(id INTEGER PRIMARY KEY, taskId
INTEGER, title TEXT, isDone INTEGER)");


        return db;

      },

      version: 1,

    );

  }


  Future<int> insertTask(Task task) async {

    int taskId = 0;

    Database _db = await database();

    await _db.insert('tasks', task.toMap(), conflictAlgorithm:
ConflictAlgorithm.replace).then((value) {

      taskId = value;

    });

    return taskId;

  }


  Future<void> updateTaskTitle(int id, String title) async {

    Database _db = await database();
```

```
    await _db.rawUpdate("UPDATE tasks SET title = '$title' WHERE id =
'$id'");

  }


  Future<void> updateTaskDescription(int id, String description) async {

    Database _db = await database();

    await _db.rawUpdate("UPDATE tasks SET description = '$description'
WHERE id = '$id'");

  }


  Future<void> insertTodo(Todo todo) async {

    Database _db = await database();

    await _db.insert('todo', todo.toMap(), conflictAlgorithm:
ConflictAlgorithm.replace);

  }


  Future<List<Task>> getTasks() async {

    Database _db = await database();

    List<Map<String, dynamic>> taskMap = await _db.query('tasks');

    return List.generate(taskMap.length, (index) {

      return Task(id: taskMap[index]['id'], title:
taskMap[index]['title'], description: taskMap[index]['description']);

    });

  }


  Future<List<Todo>> getTodo(int taskId) async {

    Database _db = await database();

    List<Map<String, dynamic>> todoMap = await _db.rawQuery("SELECT * FROM
todo WHERE taskId = $taskId");

    return List.generate(todoMap.length, (index) {
```

```
      return Todo(id: todoMap[index]['id'], title:
todoMap[index]["title"], taskId: todoMap[index]["taskId"], isDone:
todoMap[index]["isDone"]);

    });

  }


  Future<void> updateTodoDone(int id, int isDone) async {

    Database _db = await database();

    await _db.rawUpdate("UPDATE todo SET isDone = "$isDone" WHERE id =
"$id"");

  }


  Future<void> deleteTask(int id) async {

    Database _db = await database();

    await _db.rawDelete("DELETE FROM tasks WHERE id = "$id"");

    await _db.rawDelete("DELETE FROM todo WHERE taskId = "$id"");

  }


}
```

5.5 widgets.dart

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


class TaskCardWidget extends StatelessWidget {

  final String title;

  final String desc;


  TaskCardWidget({this.title, this.desc});
```

```
@override
Widget build(BuildContext context) {
  return Container(
    width: double.infinity,
    padding: EdgeInsets.symmetric(
      vertical: 32.0,
      horizontal: 24.0,
    ),
    margin: EdgeInsets.only(
      bottom: 20.0,
    ),
    decoration: BoxDecoration(
      color: Colors.white,
      borderRadius: BorderRadius.circular(20.0),
    ),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text(
          title ?? "(Unnamed Task)",
          style: TextStyle(
            color: Color(0xFF211551),
            fontSize: 22.0,
            fontWeight: FontWeight.bold,
          ),
        ),
        Padding(
          padding: EdgeInsets.only(
            top: 10.0,
```

```
            ),
            child: Text(
              desc ?? "No Description Added",
              style: TextStyle(
                fontSize: 16.0,
                color: Color(0xFF86829D),
                height: 1.5,
              ),
            ),
          )
        ],
      ),
    );
  }
}

class TodoWidget extends StatelessWidget {
  final String text;
  final bool isDone;

  TodoWidget({this.text, @required this.isDone});

  @override
  Widget build(BuildContext context) {
    return Container(
      padding: EdgeInsets.symmetric(
        horizontal: 24.0,
        vertical: 8.0,
      ),
```

```
      child: Row(
        children: [
          Container(
            width: 20.0,
            height: 20.0,
            margin: EdgeInsets.only(
              right: 12.0,
            ),
            decoration: BoxDecoration(
              color: isDone ? Color(0xFF7349FE) : Colors.transparent,
              borderRadius: BorderRadius.circular(6.0),
              border: isDone ? null : Border.all(
                color: Color(0xFF86829D),
                width: 1.5
              )
            ),
            child: Image(
              image: AssetImage("assets/images/check_icon.png"),
            ),
          ),
          Flexible(
            child: Text(
              text ?? "(Unnamed Todo)",
              style: TextStyle(
                color: isDone ? Color(0xFF211551) : Color(0xFF86829D),
                fontSize: 16.0,
                fontWeight: isDone ? FontWeight.bold : FontWeight.w500,
              ),
            ),
```

```
          ),
        ],
      ),
    );
  }
}

class NoGlowBehaviour extends ScrollBehavior {
  @override
  Widget buildViewportChrome(
    BuildContext context, Widget child, AxisDirection axisDirection) {
    return child;
  }
```

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 provided to test the process

1. A software configuration that includes a software requirement specification, a design specification and source code.
2. 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 softwareare 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 istaking 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 optionsin 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 isnot a pre-requisite and is done by the testing team. System testing is done after integrationtesting is complete. System testing should test functional and non-functional requirementsof 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 Home page

This is the landing page of the application where we see all the previously made to-do lists or tasks and it has the functionality of creating new to-do lists by clicking on the add button.

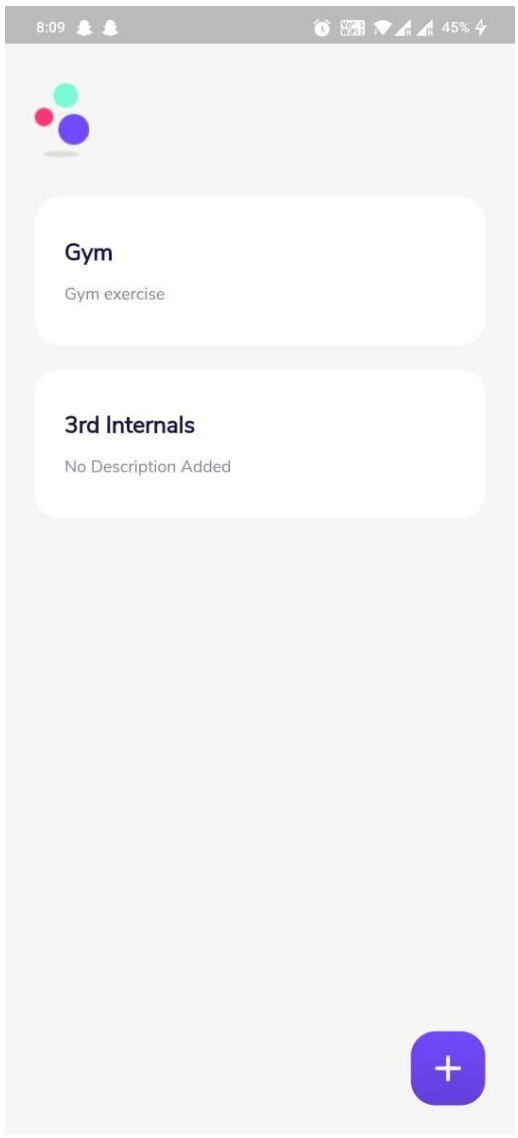

Fig 7.1 Home Page

7.2 Display to-do list

This page shows the list heading, description and the to-do items along with the check-box beside the to-do item so that user can tick-mark the tasks completed. User can also delete the list using the delete button.

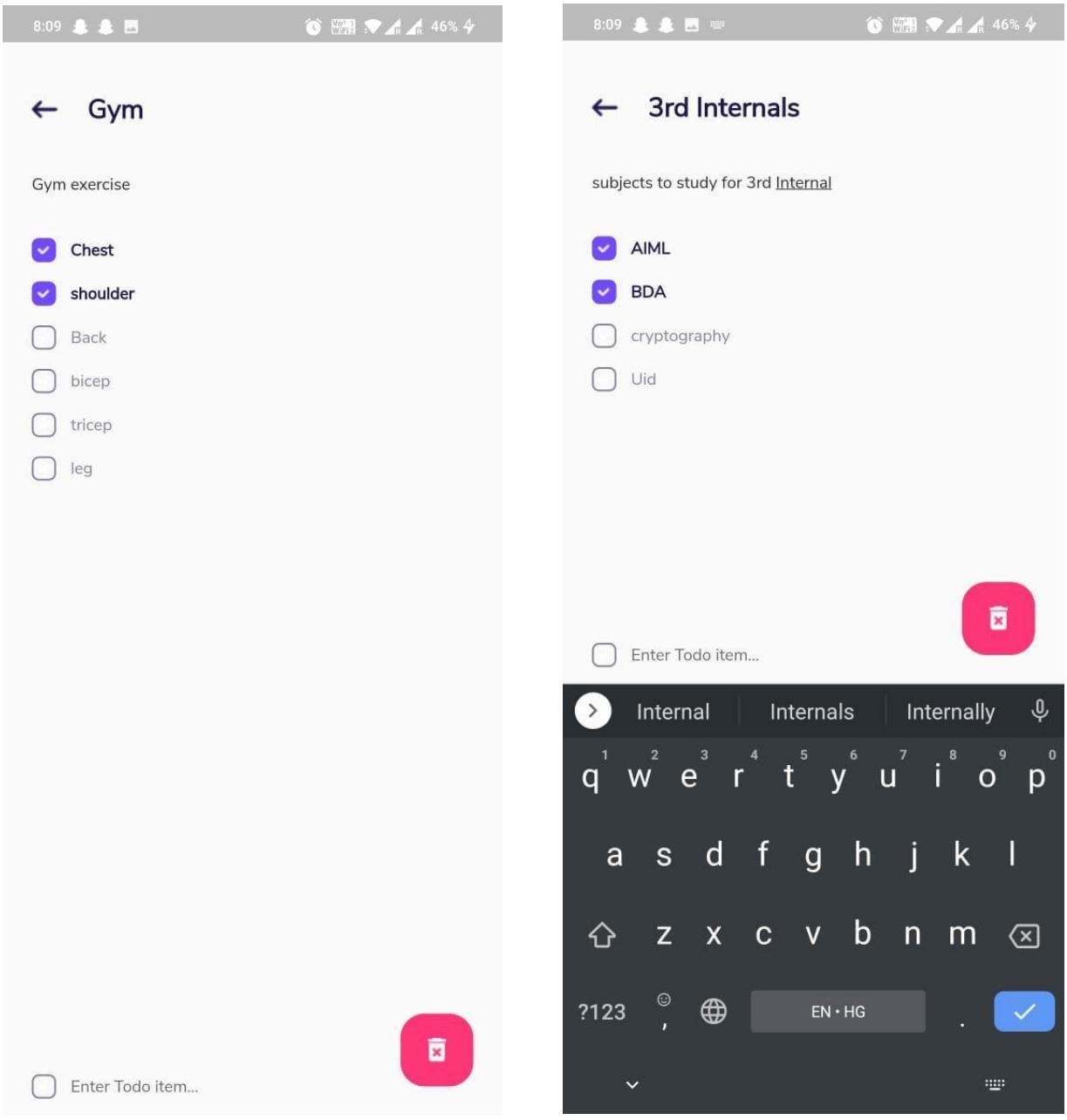

Fig 7.2 Display To-Do List

7.3 Creating a new list

While creating a new list, user can enter the list name, description and can add items by typing in the input area.

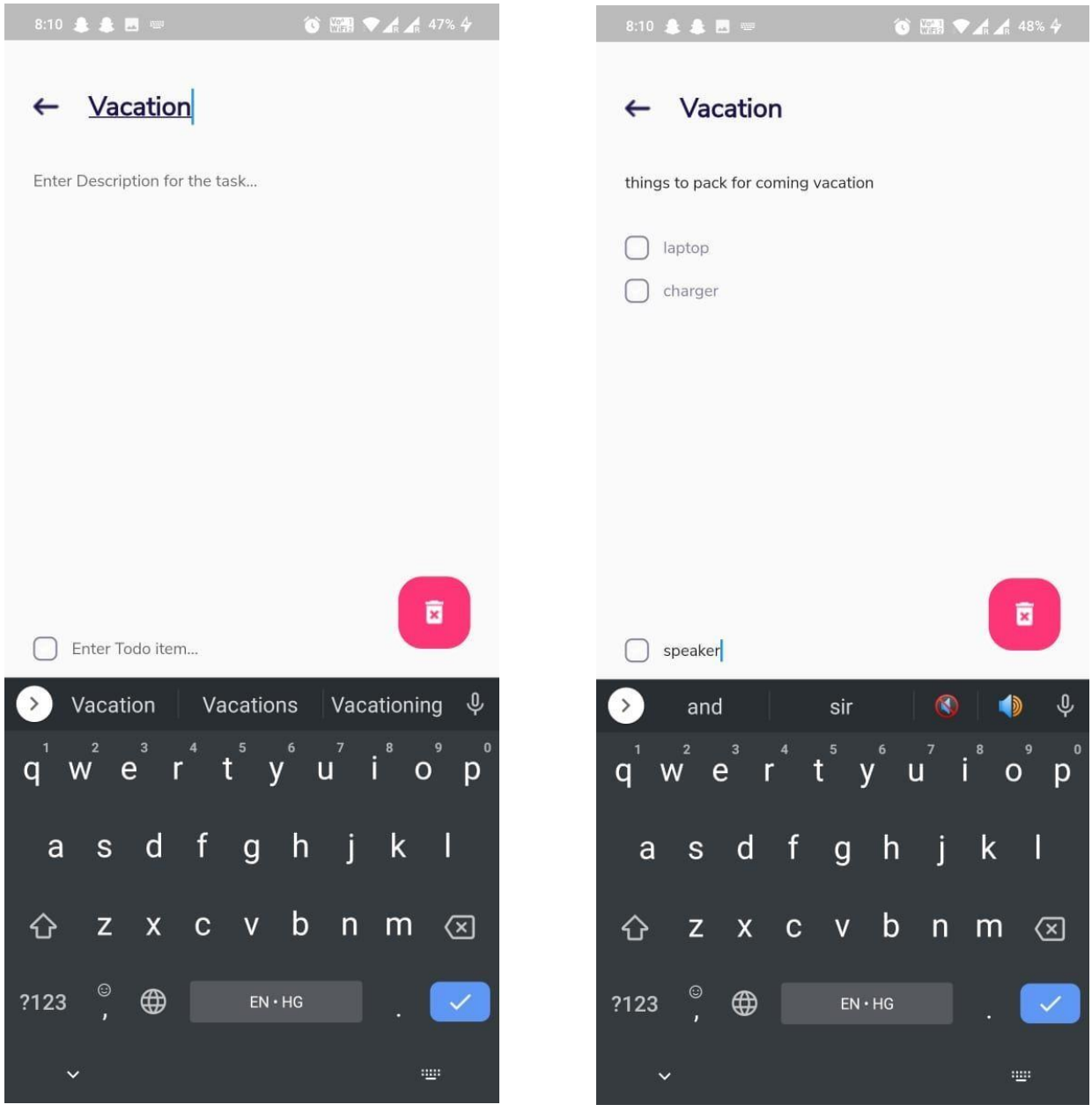

Fig 7.3 Creating a New List

Chapter 8

CONCLUSION AND FUTURE WORK

8.1 Conclusion

- This application is beneficial for the users who want to manage all their tasks at a single place.
- It is helpful in planning our daily schedules. We can add more tasks at any time and delete a task that is completed
- It helps to maintain our day-to-day tasks or list everything that we have to do wherein different tasks are divided into various categories.

8.2 Future work

- We can add the functionality to track the time we spend working on the to-do items so that we have a better sense of the estimated time we'll need to spend on our tasks.
- We can further generate a report showing where we spend our time on tasks at the end of each day.
- We can add the ability to prioritize tasks via deadline.

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/