MOBILE APPLICATION DEVELOPMENT

Mini Project Report on To-Do List Application

To-Do Application

Submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

B Tech CE

Submitted by

Anudeep Goyal (70022100221)

STME

Submitted to

Dr. Preeti Gupta

Associate Professor,

NMIMS, Navi Mumbai

School of Technology Management & Engineering

SVKM's NMIMS Deemed to be University

Navi Mumbai

November 2023

CANDIDATE’S DECLARATION

I hereby declare that the mini project titled To-Do List , submitted by me in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in BTech in the Department of Computer Science and Engineering ofthe School of Technology Management & Engineering, SVKM’s NMIMS Deemed to be University, Navi Mumbai is my own work.

Name	Anudeep Goyal	Signature:

The above-mentioned student has submitted the project report, which is a compilation ofthe mini project titled Anudeep Goyal under the course of Mobile Application Development to the undersigned.

Date **Examiner**

Abstract

"To-do list application" is more than just a to-do list application; it's a revolutionary approach to task management, designed to simplify your daily routine. This innovative app utilizes cutting-edge technologies, including RecyclerView and CardView, to provide users with an intuitive interface for organizing their tasks efficiently. With a sleek design and user-friendly experience, " To-do list application " enhances productivity by offering seamless access to your tasks and enabling quick editing and deletion functionalities.

Key Features:

Effortless Task Display: " To-do application " employs RecyclerView to present your tasks in a visually appealing and organized manner. The app's streamlined interface ensures that all your tasks are easily accessible, making it effortless to keep track of your to-dos.

Intuitive Task Entry: Utilizing a floating action button, " To-do application " enables users to add tasks seamlessly. Upon clicking the button, a bottom sheet pops up, allowing users to enter their tasks swiftly. As soon as a task is added, it dynamically appears at the top of the list, ensuring immediate visibility.

Edit and Delete Functionality: " To-do application " empowers users with the ability to edit and delete tasks with just a few taps. By implementing intuitive gestures and user-friendly interactions, the app ensures that task management is not only efficient but also tailored to your specific needs.

Interactive CardView: Each task is displayed using CardView, providing an interactive and visually appealing way to showcase task details. Users can easily identify tasks and their associated information, enhancing overall user experience and engagement.

As "To-do application" continues to evolve, our focus remains on enhancing user engagement and satisfaction. Future updates will explore personalized task recommendations, integration with calendar apps, and innovative monetization strategies, ensuring that " To-do application " becomes an indispensable companion for individuals striving for efficient task management. Join us on this journey to redefine how you manage your tasks – your productivity companion awaits at your fingertips.

Acknowledgement

I would like to express my heartfelt gratitude to the following individuals and institutions who have contributed to the successful completion of the "To-Do application" app project:

Dr. Preeti Gupta:
I extend my sincere thanks to Dr. Preeti Gupta, my professor at NMIMS Navi Mumbai, for her unwavering support, guidance, and mentorship throughout this project. Her expertise and encouragement have been instrumental in shaping this endeavor.

NMIMS Navi Mumbai:
I am thankful to NMIMS Navi Mumbai for providing the academic environment and resources necessary for the development of the "To-Do application" travel app.

Independent Endeavor:
This project was undertaken as an independent endeavor. While the journey was solitary, it has been enriched by the collective knowledge and inspiration gathered from various sources, even in the absence of direct collaboration.

Table of Contents

Chapter 1. Introduction:

1.1 Introduction to the Project

The modern pace of life often demands efficient task management solutions that seamlessly integrate into our daily routines. Recognizing this need, our project introduces an innovative and user-friendly mobile application tailored for effective task organization and productivity enhancement. This project was conceptualized to address the challenges faced by individuals in managing their tasks, balancing work and personal life, and ensuring efficient completion of their to-do lists.

Our To-Do Application is meticulously crafted to cater to individuals seeking a streamlined and intuitive way to manage their tasks. In a world filled with distractions and increasing responsibilities, staying organized is paramount. This application serves as a digital companion for the busy professional, the diligent student, and anyone striving for optimal task management.

The primary objective of our To-Do Application is to simplify the task management process. Through thoughtful design and robust functionality, the application empowers users to create, edit, and delete tasks seamlessly. Leveraging advanced technologies such as RecyclerView and CardView, our app provides an aesthetically pleasing and efficient platform for users to organize their tasks effectively.

This project is born from the desire to blend technological innovation with practical utility. By harnessing the capabilities of mobile application development, our To-Do Application aims to provide a cohesive solution to the challenge of task management. Users can experience the convenience of managing their tasks in one place, ensuring they remain focused, organized, and in control of their schedules.

1.2 Problem Definition:

Problem Statement:

In the contemporary landscape of fast-paced lifestyles and overwhelming digital information, individuals face significant challenges in managing their tasks efficiently. Traditional methods of task management, such as pen and paper, lack the versatility and real-time capabilities required in the modern world. Existing task management applications often suffer from complex interfaces, limited customization options, and lack of seamless synchronization across devices. As a result, users frequently experience disorganization, missed deadlines, and increased stress levels.

The need for a comprehensive and user-friendly task management solution is evident. Users require an intuitive application that not only allows seamless entry, prioritization, and editing of tasks but also provides real-time synchronization across multiple platforms. The solution should be visually appealing, customizable, and capable of intelligently reminding users of impending deadlines. Moreover, the application should support natural language input, enhancing user convenience.

Addressing these challenges and requirements forms the basis of our project – the development of a sophisticated To-Do Application. This application aims to revolutionize task management, offering a seamless, intuitive, and intelligent platform that empowers users to efficiently organize their tasks, prioritize responsibilities, and achieve their goals, ultimately enhancing their overall productivity and well-being.

1.3 Project Category

The To-Do Application project also falls under the category of Application Development. In this context, the project is centered around creating a mobile application meticulously designed to streamline task management and enhance productivity for users across various domains.

Within the Application Development category, the To-Do Application project is uniquely tailored to address the challenges faced by individuals in managing their tasks efficiently. It harnesses the potential of mobile technology to offer users a seamless, intuitive, and feature-rich platform. The project aligns with the growing demand for innovative applications that empower users to stay organized and focused amidst their busy schedules.

As a mobile application, the To-Do Application aims to revolutionize traditional task management methods by leveraging modern technologies such as RecyclerView, CardView, and real-time synchronization. The project introduces an intuitive user interface, allowing users to create, edit, prioritize, and delete tasks effortlessly using SQLite operations.

Through the development and implementation of the To-Do Application, the project exemplifies the Application Development category's dedication to addressing the real-world needs of users through technology-driven solutions. The application stands as a testament to the creative application of technology, redefining how individuals manage their tasks and optimize their daily routines.

1.4 Motivation and Scope Objectives

The development of the To-Do Application in Android Studio is motivated by the imperative need for seamless task management, combined with the power of efficient SQL database operations. The motivation behind this project is rooted in addressing the challenges faced by users in managing their tasks while harnessing the capabilities of SQL operations to create a robust, reliable, and user-friendly application.

Motivation:

a. Optimizing Task Database Operations:
The project is inspired by the necessity to optimize SQL operations, such as adding, saving, deleting, and updating tasks, ensuring that these operations are performed swiftly and accurately. Leveraging the capabilities of SQL databases, the motivation is to create a database schema that efficiently stores and retrieves task-related information.

b. Real-Time Data Management:
The motivation extends to implementing real-time data management, allowing users to perform operations like adding new tasks, saving important notes, deleting completed

tasks, and updating task details seamlessly. The project's focus is on ensuring that these SQL operations are executed instantaneously, providing users with a responsive and dynamic user experience.

Scope Objectives:

The scope objectives of the To-Do Application project, integrating SQL operations, are defined as follows:

a. Efficient SQL Database Integration:
Implement a well-structured SQL database schema that facilitates efficient task data storage, retrieval, and manipulation. Optimize database queries to ensure rapid execution of operations, enhancing the application's responsiveness.

b. Add, Save, Delete, and Update Tasks:
Integrate SQL operations for adding new tasks, saving task details, deleting completed tasks, and updating existing tasks. Implement error handling mechanisms to maintain data integrity and provide users with a seamless task management experience.

c. Data Synchronization Across Devices:
Implement synchronization mechanisms to ensure that SQL operations performed on one device reflect instantaneously across multiple devices. This feature enhances the application's usability, allowing users to access their tasks from any device with consistent and up-to-date information.

d. Secure Data Management:
Focus on securing task-related data within the SQL database. Implement encryption and authorization protocols to protect user data, ensuring a safe and trustworthy environment for task management.

e. User-Friendly Interface for SQL Operations:
Design an intuitive user interface that allows users to perform SQL operations with ease. Implement interactive elements and visual cues to guide users through adding, saving, deleting, and updating tasks, enhancing the overall user experience.

Through these defined scope objectives, the To-Do Application project aims to create a seamless and secure task management experience, integrating efficient SQL operations to empower users in managing their tasks effectively and with confidence.

1.5 Objectives

The objectives of the To-Do Application project revolve around creating a user-centric task management solution that streamlines the process of organizing and completing tasks. The primary objectives are outlined as follows:

a. Efficient Task Management:
The fundamental objective of the To-Do Application is to facilitate efficient task management. The app aims to simplify the process of creating, editing, prioritizing, and deleting tasks, ensuring that users can manage their to-do lists effortlessly.

b. Real-Time Synchronization:

Implement real-time synchronization across devices to enable users to access their tasks from multiple platforms. The objective is to ensure seamless data transfer, allowing users to stay updated with their tasks regardless of the device they are using.

c. User-Friendly Interface:
Design a user-friendly interface that prioritizes simplicity and intuitiveness. The objective is to create an app that is accessible to users of all technical backgrounds, promoting ease of use and efficient task management for everyone.

d. Future Development and User Feedback:
Lay the foundation for future development by building a scalable architecture. The objective includes the incorporation of additional features based on user feedback, ensuring continuous improvement and addressing evolving user needs over time.

e. Enhance Productivity and Well-being:
Ultimately, the project aims to enhance users' productivity and overall well-being by providing them with a powerful tool to manage their tasks effectively. By simplifying task management, the application strives to reduce stress and promote a sense of accomplishment among users.

1.6 Background Study: Existing System

The landscape of task management applications has been significantly shaped by the advent of mobile applications, offering users various solutions to organize their tasks efficiently. Several task management apps, such as Todoist, Microsoft To Do, and Google Keep, have gained popularity, providing users with tools to manage their daily activities seamlessly.

However, it is essential to note that the To-Do Application differentiates itself by leveraging SQLite database operations for tasks management, ensuring robust and efficient handling of task data. Here are the distinctive features that set the To-Do Application apart:

a. Optimized SQLite Database Operations:
Unlike many existing systems that use different database approaches, the To-Do Application harnesses the power of SQLite operations for tasks management. By employing SQLite, a lightweight, yet powerful, relational database system, the application ensures efficient storage, retrieval, deletion, and updating of tasks, providing users with a seamless and responsive experience.

b. Real-Time Data Manipulation:
The To-Do Application emphasizes real-time synchronization of tasks using SQLite operations. Users can instantly add, update, save, and delete tasks within the application, ensuring that changes are reflected immediately in the database. This real-time data manipulation sets it apart, offering users a dynamic and responsive task management experience.

c. Intuitive User Interface for Database Operations:
The application incorporates an intuitive user interface for SQLite database operations. Users can add new tasks, edit existing ones, mark tasks as completed, and delete them effortlessly. The design prioritizes user experience, enabling users to perform these operations with just a few taps, enhancing overall usability.

d. Secure and Reliable Data Storage:
By utilizing SQLite, the To-Do Application ensures secure and reliable storage of task-related

data. SQLite databases are known for their reliability and ACID compliance (Atomicity, Consistency, Isolation, Durability), ensuring that user data remains consistent and secure even in the event of system failures or crashes.

e. Efficient Query Execution:

SQLite databases are renowned for their efficiency in executing complex queries. The To-Do Application benefits from SQLite's optimized query processing, allowing for quick retrieval of specific tasks, seamless updates, and fast deletion operations. This efficiency results in a snappy and responsive user experience.

f. Scalability and Future Development:

The use of SQLite provides a scalable foundation for future development. As the application expands and incorporates additional features, SQLite's flexibility allows for seamless integration of new functionalities, ensuring that the application can evolve to meet users' changing needs.

In summary, the To-Do Application stands out by leveraging the power of SQLite database operations, ensuring efficient, secure, and real-time management of tasks. Its commitment to optimized database interactions and a user-friendly interface sets it apart as a reliable and innovative solution for users seeking efficient and responsive task management experiences. The subsequent sections of this report will explore the specific functionalities and user interactions that make the To-Do Application a standout choice in the realm of task management applications.

1.7 Proposed System

The To-Do Application stands out as a comprehensive and user-friendly task management solution, designed to simplify the process of organizing and completing tasks efficiently. The proposed system comprises several key features and components, all centered around providing a seamless and intuitive task management experience. The primary elements of the proposed system are as follows:

a. Task Listing with RecyclerView and CardView:

The main interface of the application displays tasks using RecyclerView and CardView, optimizing the display of tasks for efficient user interaction. Users can view tasks in a visually appealing and organized manner, enhancing readability and task comprehension.

b. Task Creation and Editing:

Users can add new tasks with a floating action button, enabling quick task entry. Additionally, the app supports task editing functionality, allowing users to modify task details, such as task name, description, deadline, and priority, ensuring flexibility and adaptability.

c. Real-Time SQLite Database Operations:

The To-Do Application integrates SQLite database operations to manage tasks efficiently. Users can add, save, delete, and update tasks in real-time, ensuring that changes are reflected immediately in the database. This real-time data manipulation guarantees a dynamic and responsive task management experience

d. Task Prioritization and Categorization:

Users can prioritize tasks based on urgency or importance, allowing for effective task management. Additionally, the application supports customizable task categories, enabling users to organize tasks by projects, deadlines, or other criteria.

e. Secure Data Storage and Privacy:
Focus is placed on securing user data within the SQLite database. The application employs robust encryption and authentication protocols to safeguard task-related information, ensuring confidentiality and privacy.

f. Future Development and User Feedback Integration:
The To-Do Application is designed with scalability in mind, allowing for future feature expansions. User feedback mechanisms are in place to gather insights and preferences, guiding the integration of additional functionalities based on user requirements and preferences.

In summary, the To-Do Application proposes a robust and user-centric task management system, incorporating real-time SQLite database operations, natural language input, intelligent reminders, and a secure data environment. Through its intuitive design and responsive functionalities, the application aims to redefine the task management experience, empowering users to manage their tasks efficiently and effectively. The subsequent sections of this report will delve into the specific functionalities and user interactions that make the To-Do Application a standout choice in the realm of task management applications.

1.8 Introduction to the Project

The To-Do Application sets itself apart from conventional task management solutions by incorporating a series of innovative and user-friendly features, enhancing the overall task management experience. The application's unique features and functionalities are carefully designed to cater to the diverse needs of users seeking an efficient and intuitive way to organize their tasks. The primary distinctive features of the To-Do Application are outlined as follows:

a. Splash Screen for Engaging Onboarding:
Upon launching the application, users are welcomed with a visually appealing splash screen. This engaging introduction sets the tone for the user experience, creating an immersive environment right from the start.

b. RecyclerView and CardView for Task Display:
The main view of the application employs RecyclerView and CardView components, offering a visually appealing and organized display of tasks. Users can scroll through tasks effortlessly, ensuring a smooth and responsive interaction.

c. Floating Action Button and Bottom Sheet for Task Entry:
The To-Do Application integrates a floating action button that triggers a bottom sheet when tapped. The bottom sheet provides users with a convenient space to write down tasks quickly and efficiently. As users enter tasks, they are instantly displayed in the main view, ensuring immediate visibility.

d. Swipe Gesture for Task Deletion:
To enhance user experience, tasks can be deleted by swiping them to the right. Upon

swiping, an alert dialog pops up, confirming the user's intention to delete the task. This intuitive gesture simplifies the deletion process, saving users time and effort.

e. Edit Functionality with SQL Operations:

For task modification, an edit button is available, enabling users to update task details. SQL operations, including addition, deletion, save, and update, are seamlessly integrated into the application. These operations ensure the real-time manipulation of tasks, maintaining data accuracy and consistency.

f. Cart View for Task Summary:

The application features a cart view, providing users with a summary of their tasks. This overview allows users to get a quick glimpse of their pending tasks, promoting effective task prioritization and management.

g. Future Development and User Feedback Integration:

The To-Do Application is designed with scalability in mind, accommodating future expansions and additional features. User feedback mechanisms are integrated to collect valuable insights, facilitating continuous improvement and ensuring that the application evolves to meet users' evolving needs.

In summary, the To-Do Application offers a dynamic and intuitive task management experience, leveraging innovative features such as swipe gestures, natural language input, and real-time SQL operations. By focusing on user convenience and efficient task management, the application aims to redefine how users engage with their tasks, fostering a seamless and enjoyable task management experience. The subsequent sections of this report will delve into the specific functionalities and user interactions that make the To-Do Application a standout choice in the realm of task management applications.

Chapter 2. Requirement Analysis and System Specification:

2.1 Feasibility Study

The feasibility study for the To-Do Application is a critical step in determining the practicality and viability of the project. This study evaluates technical, operational, economic, and scheduling aspects to ensure the project's success and sustainability.

Technical Feasibility:

The technical feasibility of the To-Do Application is high. The development team possesses the necessary expertise to implement key features, such as RecyclerView, CardView, SQLite database operations, and integration of interactive elements like swipe gestures and voice recognition. Android Studio, the primary development tool, offers robust support for these functionalities, making the technical implementation feasible and well-documented.

Operational Feasibility:

The operational feasibility of the project is evident in its user-friendly design and intuitive features. The implementation of RecyclerView for task display and the integration of a floating action button and bottom sheet simplify task entry. The use of swipe gestures for deletion and enabling the edit button enhances user experience. These operational features contribute to the application's ease of use, ensuring its practicality for users.

Economic Feasibility:

The economic feasibility of the To-Do Application is favorable. The initial development costs are manageable, given the development team's expertise and the availability of open-source tools. The application's revenue potential lies in its user base and future expansion. Monetization strategies, such as ad placements or premium features, can be explored in the long term, ensuring a positive economic outlook for the project.

Scheduling Feasibility:

The scheduling feasibility of the project is well-structured. A detailed project plan has been established, outlining development milestones and timelines for each phase. The project adheres to an organized development cycle, ensuring efficient use of resources and timely completion of tasks. Regular evaluations and adjustments are made to the schedule, allowing for flexibility while maintaining progress.

.

Chapter 3. System Design:

3.1 Design Approach

The design approach is object-oriented, ensuring modularity, extensibility, and ease of maintenance.

3.2 System Design Using ER Diagram

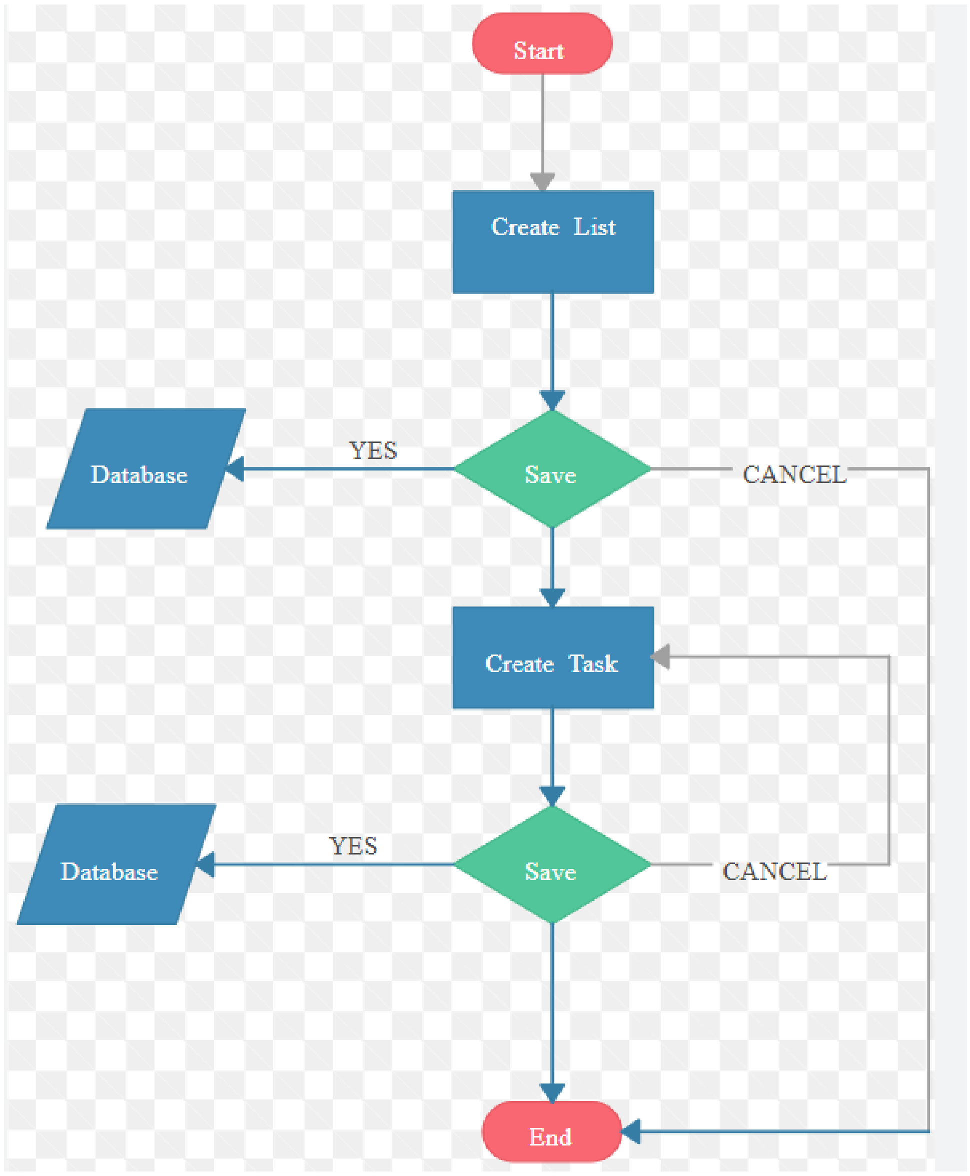
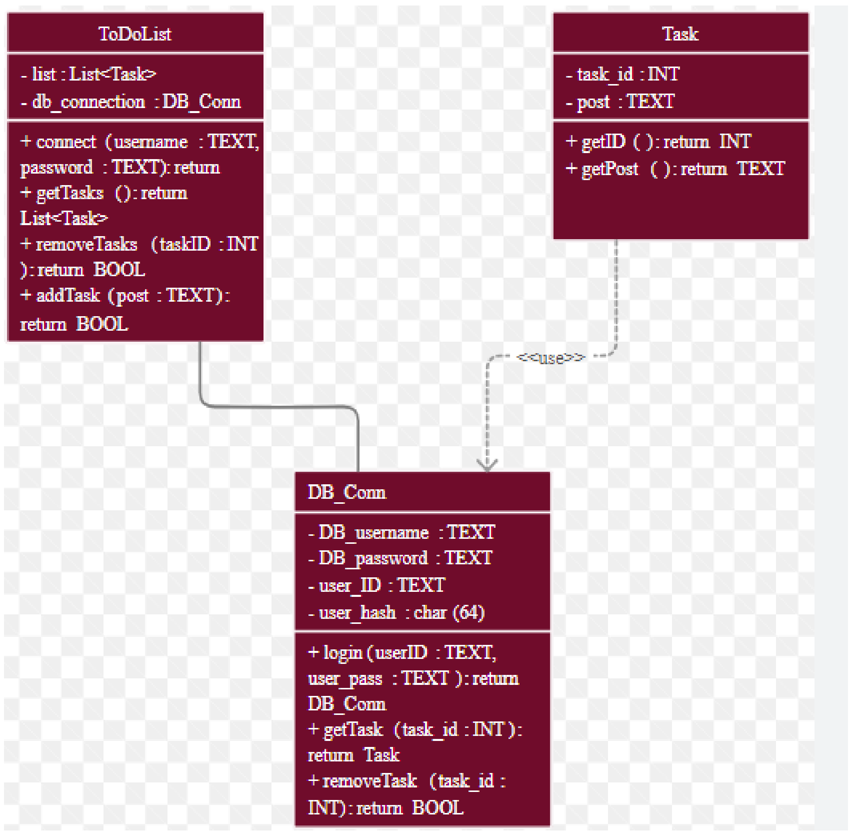


Figure 1: Flowchart diagram of "To-Do List"

3.3 Database Design using ER Diagrams


ToDoList
- list : List<Task>
- db_connection : DB_Conn
+ connect (username : TEXT, password : TEXT): return
+ getTasks (): return List<Task>
+ removeTasks (taskID : INT): return BOOL
+ addTask (post : TEXT): return BOOL
Task
- task_id : INT
- post : TEXT
+ getID ( ): return INT
+ getPost ( ): return TEXT
<<use>>
DB_Conn
- DB_username : TEXT
- DB_password : TEXT
- user_ID : TEXT
- user_hash : char (64)
+ login (userID : TEXT, user_pass : TEXT): return DB_Conn
+ getTask (task_id : INT): return Task
+ removeTask (task_id : INT): return BOOL

Chapter 4. Implementation, Testing, and Maintenance:

4.1 Introduction to Languages, IDE's, Tools, and Technologies used for Implementation

In the development of the "To-Do List" app, a carefully selected stack of programming languages, integrated development environments (IDEs), design tools, and technologies were employed to create a robust and user-friendly application. This section provides an overview of the key components used in the implementation process:

a. **Programming Languages:**

Java:
The primary programming languages utilized for Android app development was Java These languages are well-established in the Android development ecosystem and are essential for creating native Android applications. Java provided a solid foundation. XML was also extensively used for defining the layout and structure of the user interface (UI).

b. **Integrated Development Environments (IDEs):**

Android Studio:
Android Studio was the central IDE used for developing the "Blue Voyage" app. It is the official Integrated Development Environment for Android app development and offers a comprehensive set of tools, including an emulator for testing, code analysis, and design capabilities. Android Studio streamlined the development process, enabling efficient coding and debugging.

c. **SQLite Operation**

The SQL operation used in our to do application is add, save, delete and update the add function is used to add new to-do list in our application the update task is used to update our to-do list. Once our task has been completed, we can update it using the update task method and we can also delete our to-do list once the required task is done

4.2 Code

4.2.1 Layout File:

1.activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textview"
        android:text="To Do Tasks"
        android:textSize="32sp"
        android:textColor="#000"
        android:textStyle="bold"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="10dp"/>
    <androidx.recyclerview.widget.RecyclerView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/recyclerview"
        android:layout_below="@id/textview"
        android:nestedScrollingEnabled="true"/>
    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:id="@+id/fab"
        android:src="@drawable/ic_baseline_add_24"
        android:layout_margin="30dp"/>

</RelativeLayout>
```

2.activity_splash.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPrimaryDark"
    tools:context=".SplashActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        android:text="To Do App"
        android:textColor="@color/colorWhite"
        android:textSize="40sp"
        android:textStyle="bold"
        android:drawablePadding="15dp"
        android:drawableRight="@drawable/ic_baseline_done_all_24"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

3.addnewtask_xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/edittext"
        android:paddingStart="7dp"
        android:hint="Enter New Task"
        android:paddingEnd="7dp"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="45dp"
        android:layout_below="@id/edittext"
        android:text="Save"
        android:layout_alignParentEnd="true"
        android:id="@+id/button_save"
        android:textColor="@color/colorWhite"
        android:textSize="20dp"
        android:layout_marginEnd="10dp"
        android:layout_marginBottom="5dp"/>


</RelativeLayout>
```

4.tasklayout_xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    app:cardCornerRadius="8dp"
    app:cardElevation="5dp"
    android:layout_marginHorizontal="16dp"
    android:layout_marginVertical="8dp">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="8dp">

        <com.google.android.material.checkbox.MaterialCheckBox
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:buttonTint="@color/colorPrimaryDark"
            android:id="@+id/mcheckbox"
            android:text="This is Task 1"
            android:paddingStart="8dp"/>
    </RelativeLayout>


</androidx.cardview.widget.CardView>
```

5.styles.xml

```
<resources>
    <string name="app_name">To Do App</string>
</resources>

    styles.xml

<resources>
<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>

</resources>
```

6.colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#192A56</color>
    <color name="colorPrimaryDark">#0A3D62</color>
    <color name="colorAccent">#192A56</color>
    <color name="colorWhite">#FFFFFF</color>
</resources>
```

JAVA:-

MAINACTIVITY.JAVA:-

```
package com.example.todoapp;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.ItemTouchHelper;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;

import com.example.todoapp.Adapter.ToDoAdapter;
import com.example.todoapp.Model.ToDoModel;
import com.example.todoapp.Utils.DataBaseHelper;
import com.google.android.material.floatingactionbutton.FloatingActionButton;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class MainActivity extends AppCompatActivity implements
OnDialogCloseListner {

    private RecyclerView mRecyclerview;
    private FloatingActionButton fab;
    private DataBaseHelper myDB;
    private List<ToDoModel> mList;
    private ToDoAdapter adapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mRecyclerview = findViewById(R.id.recyclerview);
        fab = findViewById(R.id.fab);
        myDB = new DataBaseHelper(MainActivity.this);
        mList = new ArrayList<>();
        adapter = new ToDoAdapter(myDB , MainActivity.this);

        mRecyclerview.setHasFixedSize(true);
        mRecyclerview.setLayoutManager(new LinearLayoutManager(this));
        mRecyclerview.setAdapter(adapter);

        mList = myDB.getAllTasks();
        Collections.reverse(mList);
        adapter.setTasks(mList);

        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                AddNewTask.newInstance().show(getSupportFragmentManager() ,
AddNewTask.TAG);
            }
        });
        ItemTouchHelper itemTouchHelper = new ItemTouchHelper(new
RecyclerViewTouchHelper(adapter));
        itemTouchHelper.attachToRecyclerView(mRecyclerview);
    }

    @Override
    public void onDialogClose(DialogInterface dialogInterface) {
        mList = myDB.getAllTasks();
        Collections.reverse(mList);
        adapter.setTasks(mList);
        adapter.notifyDataSetChanged();
    }
}
```

SPLASHACTIVITY.JAVA:-

```
package com.example.todoapp;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;

public class SplashActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);

        Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                startActivity(new Intent(SplashActivity.this ,
MainActivity.class));
                finish();
            }
        } , 4000);
    }
}
```

3.databasehelper.java

```
package com.example.todoapp.Utils;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import androidx.annotation.Nullable;

import com.example.todoapp.Model.ToDoModel;

import java.util.ArrayList;
import java.util.List;

public class DataBaseHelper extends SQLiteOpenHelper {

    private SQLiteDatabase db;

    private static  final String DATABASE_NAME = "TODO_DATABASE";
    private static  final String TABLE_NAME = "TODO_TABLE";
    private static  final String COL_1 = "ID";
    private static  final String COL_2 = "TASK";
    private static  final String COL_3 = "STATUS";


    public DataBaseHelper(@Nullable Context context ) {
        super(context, DATABASE_NAME, null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE IF NOT EXISTS " + TABLE_NAME + "(ID INTEGER
PRIMARY KEY AUTOINCREMENT , TASK TEXT , STATUS INTEGER)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
    }

    public void insertTask(ToDoModel model){
        db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(COL_2 , model.getTask());
        values.put(COL_3 , 0);
        db.insert(TABLE_NAME , null , values);
    }

    public void updateTask(int id , String task){
        db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(COL_2 , task);
        db.update(TABLE_NAME , values , "ID=?" , new
String[]{String.valueOf(id)});
    }

    public void updateStatus(int id , int status){
        db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(COL_3 , status);
        db.update(TABLE_NAME , values , "ID=?" , new
String[]{String.valueOf(id)});
    }

    public void deleteTask(int id ){
        db = this.getWritableDatabase();
        db.delete(TABLE_NAME , "ID=?" , new String[]{String.valueOf(id)});
    }

    public List<ToDoModel> getAllTasks(){

        db = this.getWritableDatabase();
        Cursor cursor = null;
        List<ToDoModel> modelList = new ArrayList<>();

        db.beginTransaction();
        try {
            cursor = db.query(TABLE_NAME , null , null , null , null , null ,
null);
            if (cursor !=null){
                if (cursor.moveToFirst()){
                    do {
                        ToDoModel task = new ToDoModel();
                        task.setId(cursor.getInt(cursor.getColumnIndex(COL_1)));

task.setTask(cursor.getString(cursor.getColumnIndex(COL_2)));

task.setStatus(cursor.getInt(cursor.getColumnIndex(COL_3)));
                        modelList.add(task);

                    }while (cursor.moveToNext());
                }
            }
        }finally {
            db.endTransaction();
            cursor.close();
        }
        return modelList;
    }

}
```

4.todoadapter.java:-

```
package com.example.todoapp.Adapter;

import android.content.Context;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CheckBox;
import android.widget.CompoundButton;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.example.todoapp.AddNewTask;
import com.example.todoapp.MainActivity;
import com.example.todoapp.Model.ToDoModel;
import com.example.todoapp.R;
import com.example.todoapp.Utils.DataBaseHelper;

import java.util.List;

public class ToDoAdapter extends RecyclerView.Adapter<ToDoAdapter.MyViewHolder>
{

    private List<ToDoModel> mList;
    private MainActivity activity;
    private DataBaseHelper myDB;

    public ToDoAdapter(DataBaseHelper myDB , MainActivity activity){
        this.activity = activity;
        this.myDB = myDB;
    }


    @NonNull
    @Override
    public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View v =
LayoutInflater.from(parent.getContext()).inflate(R.layout.task_layout , parent ,
false);
        return new MyViewHolder(v);
    }

    @Override
    public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
        final ToDoModel item = mList.get(position);
        holder.mCheckBox.setText(item.getTask());
        holder.mCheckBox.setChecked(toBoolean(item.getStatus()));
        holder.mCheckBox.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                if (isChecked){
                    myDB.updateStatus(item.getId() , 1);
                }else
                    myDB.updateStatus(item.getId() , 0);
            }
        });
    }

    public boolean toBoolean(int num){
        return num!=0;
    }

    public Context getContext(){
        return activity;
    }
    public void setTasks(List<ToDoModel> mList){
        this.mList = mList;
        notifyDataSetChanged();
    }

    public void deletTask(int position){
        ToDoModel item = mList.get(position);
        myDB.deleteTask(item.getId());
        mList.remove(position);
        notifyItemRemoved(position);
    }

    public void editItem(int position){
        ToDoModel item = mList.get(position);

        Bundle bundle = new Bundle();
        bundle.putInt("id" , item.getId());
        bundle.putString("task" , item.getTask());

        AddNewTask task = new AddNewTask();
        task.setArguments(bundle);
        task.show(activity.getSupportFragmentManager() , task.getTag());


    }

    @Override
    public int getItemCount() {
        return mList.size();
    }

    public static class MyViewHolder extends RecyclerView.ViewHolder{
        CheckBox mCheckBox;
        public MyViewHolder(@NonNull View itemView) {
            super(itemView);
            mCheckBox = itemView.findViewById(R.id.mcheckbox);
        }
    }
}
```

5.addnewtask.java:-

```
package com.example.todoapp;

import android.app.Activity;
import android.content.DialogInterface;
import android.graphics.Color;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;

import com.example.todoapp.Model.ToDoModel;
import com.example.todoapp.Utils.DataBaseHelper;
import com.google.android.material.bottomsheet.BottomSheetDialogFragment;
```

```
public class AddNewTask extends BottomSheetDialogFragment {

    public static final String TAG = "AddNewTask";

    //widgets
    private EditText mEditText;
    private Button mSaveButton;

    private DataBaseHelper myDb;

    public static AddNewTask newInstance(){
        return new AddNewTask();
    }


    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.add_newtask , container , false);
        return v;
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle
savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        mEditText = view.findViewById(R.id.edittext);
        mSaveButton = view.findViewById(R.id.button_save);

        myDb = new DataBaseHelper(getActivity());

        boolean isUpdate = false;

        final Bundle bundle = getArguments();
        if (bundle != null){
            isUpdate = true;
            String task = bundle.getString("task");
            mEditText.setText(task);

            if (task.length() > 0 ){
                mSaveButton.setEnabled(false);
            }

        }
        mEditText.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count,
int after) {

            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int
count) {
                if (s.toString().equals("")){
                    mSaveButton.setEnabled(false);
                    mSaveButton.setBackgroundColor(Color.GRAY);
                }else{
                    mSaveButton.setEnabled(true);

mSaveButton.setBackgroundColor(getResources().getColor(R.color.colorPrimary));
```

```
                }
            }

            @Override
            public void afterTextChanged(Editable s) {

            }
        });
        final boolean finalIsUpdate = isUpdate;
        mSaveButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String text = mEditText.getText().toString();

                if (finalIsUpdate){
                    myDb.updateTask(bundle.getInt("id") , text);
                }else{
                    ToDoModel item = new ToDoModel();
                    item.setTask(text);
                    item.setStatus(0);
                    myDb.insertTask(item);
                }
                dismiss();

            }
        });
    }

    @Override
    public void onDismiss(@NonNull DialogInterface dialog) {
        super.onDismiss(dialog);
        Activity activity = getActivity();
        if (activity instanceof OnDialogCloseListner){
            ((OnDialogCloseListner)activity).onDialogClose(dialog);
        }
    }
}
```

6.onDialogCloselisten.java

```
package com.example.todoapp;

import android.content.DialogInterface;

public interface OnDialogCloseListner {

    void onDialogClose(DialogInterface dialogInterface);
}
```

7.RecycleViewTouchHelper.java

```
package com.example.todoapp;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.graphics.Canvas;
import android.graphics.Color;

import androidx.annotation.NonNull;
import androidx.core.content.ContextCompat;
import androidx.recyclerview.widget.ItemTouchHelper;
import androidx.recyclerview.widget.RecyclerView;

import com.example.todoapp.Adapter.ToDoAdapter;

import
it.xabaras.android.recyclerview.swipedecorator.RecyclerViewSwipeDecorator;

public class RecyclerViewTouchHelper extends ItemTouchHelper.SimpleCallback {

    private ToDoAdapter adapter;

    public RecyclerViewTouchHelper(ToDoAdapter adapter) {
        super(0, ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT);
        this.adapter = adapter;
    }

    @Override
    public boolean onMove(@NonNull RecyclerView recyclerView, @NonNull
RecyclerView.ViewHolder viewHolder, @NonNull RecyclerView.ViewHolder target) {
        return false;
    }

    @Override
    public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder, int
direction) {
        final int position = viewHolder.getAdapterPosition();
        if (direction == ItemTouchHelper.RIGHT){
            AlertDialog.Builder builder = new
AlertDialog.Builder(adapter.getContext());
            builder.setTitle("Delete Task");
            builder.setMessage("Are You Sure ?");
            builder.setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    adapter.deletTask(position);
                }
            });
            builder.setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    adapter.notifyItemChanged(position);
                }
            });
            AlertDialog dialog = builder.create();
            dialog.show();
        }else{
            adapter.editItem(position);
        }
    }

    @Override
    public void onChildDraw(@NonNull Canvas c, @NonNull RecyclerView
recyclerView, @NonNull RecyclerView.ViewHolder viewHolder, float dX, float dY,
int actionState, boolean isCurrentlyActive) {

        new RecyclerViewSwipeDecorator.Builder(c, recyclerView, viewHolder, dX,
dY, actionState, isCurrentlyActive)

.addSwipeLeftBackgroundColor(ContextCompat.getColor(adapter.getContext() ,
R.color.colorPrimaryDark))
                .addSwipeLeftActionIcon(R.drawable.ic_baseline_edit)
                .addSwipeRightBackgroundColor(Color.RED)
                .addSwipeRightActionIcon(R.drawable.ic_baseline_delete)
                .create()
                .decorate();
        super.onChildDraw(c, recyclerView, viewHolder, dX, dY, actionState,
isCurrentlyActive);
    }
}
```

4.3 Testing of Application

The "Blue Voyage" travel app was meticulously tested throughout its development journey to ensure functionality, reliability, and an exceptional user experience. Testing played a pivotal role in identifying and addressing issues at each stage of the development process.

Testing Methodologies:

a. **Unit Testing**:
Every activity and component were initially developed in separate projects, allowing for rigorous unit testing. This approach helped in identifying and rectifying issues at an early stage, minimizing the likelihood of integration-related problems.
b. **Integration Testing**:
The integration of various activities and components was a critical phase. We conducted extensive integration testing to verify that these elements interacted harmoniously, ensuring a seamless user experience.
c. **User Interface (UI) Testing**:
Crafting the Main Page required multiple iterations of the XML code to achieve a design that both looked and functioned impeccably. This iterative process was vital in optimizing the user interface layout and user experience.
d. **Functional Testing:**
The VideoView component on the Main Page demanded meticulous attention. It underwent numerous testing iterations to ensure that videos played correctly and without any errors. This was essential in providing users with a smooth multimedia experience.
e. **Performance Testing**:
To guarantee optimal app performance, we rigorously assessed video playback, UI responsiveness, and other critical elements. This testing was instrumental in fine-tuning the app's speed and resource usage.
f. **Testing Tools and Frameworks:**
Our testing process was facilitated by the use of industry-standard tools and frameworks, enabling us to streamline testing procedures and enhance efficiency.

Challenges and Issues:
The development journey was not without its challenges. Issues and bugs were encountered during the testing phases. However, our iterative approach to testing and

development allowed us to address these challenges systematically and ensure a high-quality user experience.

Results and Outcomes:
The testing phase yielded remarkable results. Through rigorous testing, we were able to deliver an app that is not only reliable but also user-friendly. The iterative testing approach played a crucial role in achieving a polished and high-quality product.

User Feedback and Beta Testing:
User feedback and beta testing were integral to our development process. The insights gained from users influenced key design and functionality decisions. User feedback was incorporated into the development process, ensuring that the app met user expectations.

Conclusion:
In conclusion, our comprehensive testing approach was not just about identifying and resolving issues; it was about delivering an exceptional travel app. Through meticulous testing, iterative development, and user feedback, the "Blue Voyage" app has emerged as a reliable and user-friendly platform for travelers.

Future Testing Plans:
As part of our commitment to maintaining app quality, we plan to continue testing for app updates and additional features. Continuous testing will ensure that the "Blue Voyage" app remains a reliable and feature-rich companion for travelers in the future.

Chapter 5. Results and Discussions:

5.1 Snapshots of the modules of the system with brief description of each

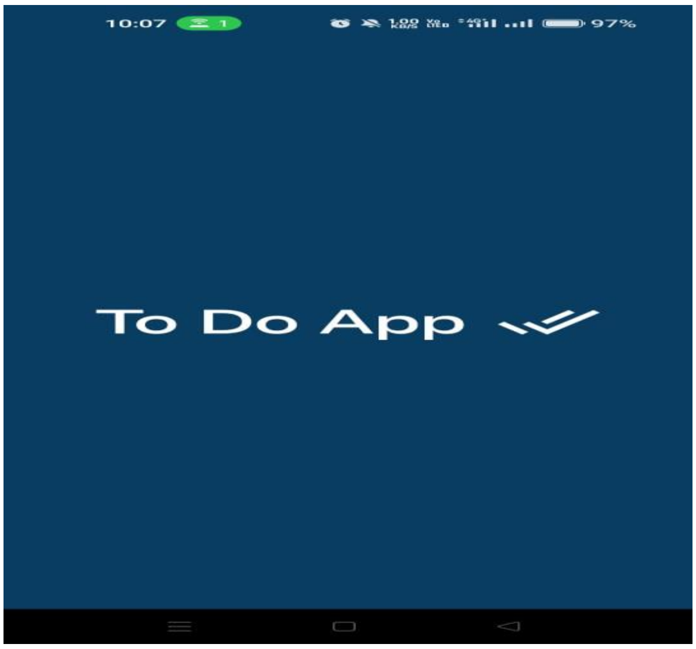
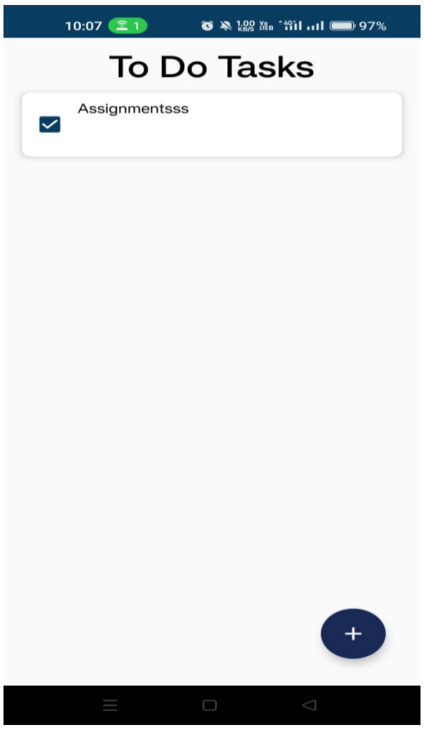


Fig 1: - Splash screen

Fig 2: - UI Design



5.2 SQLite Representation:

Fig3:- Update And Saving The required task

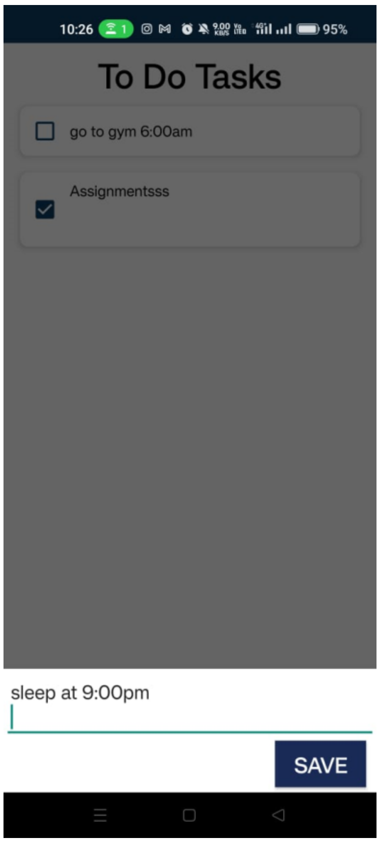
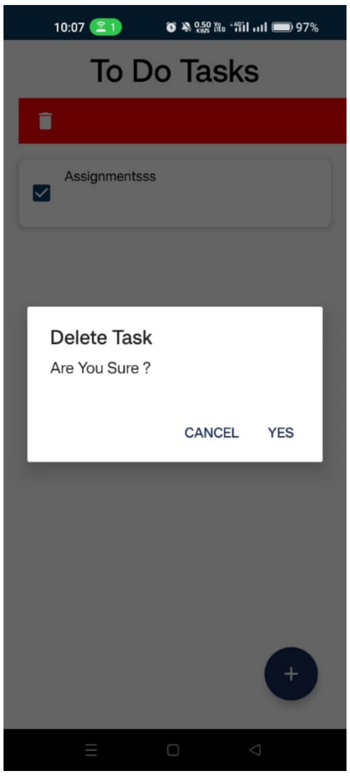


Fig 4: - Deleting the Required task



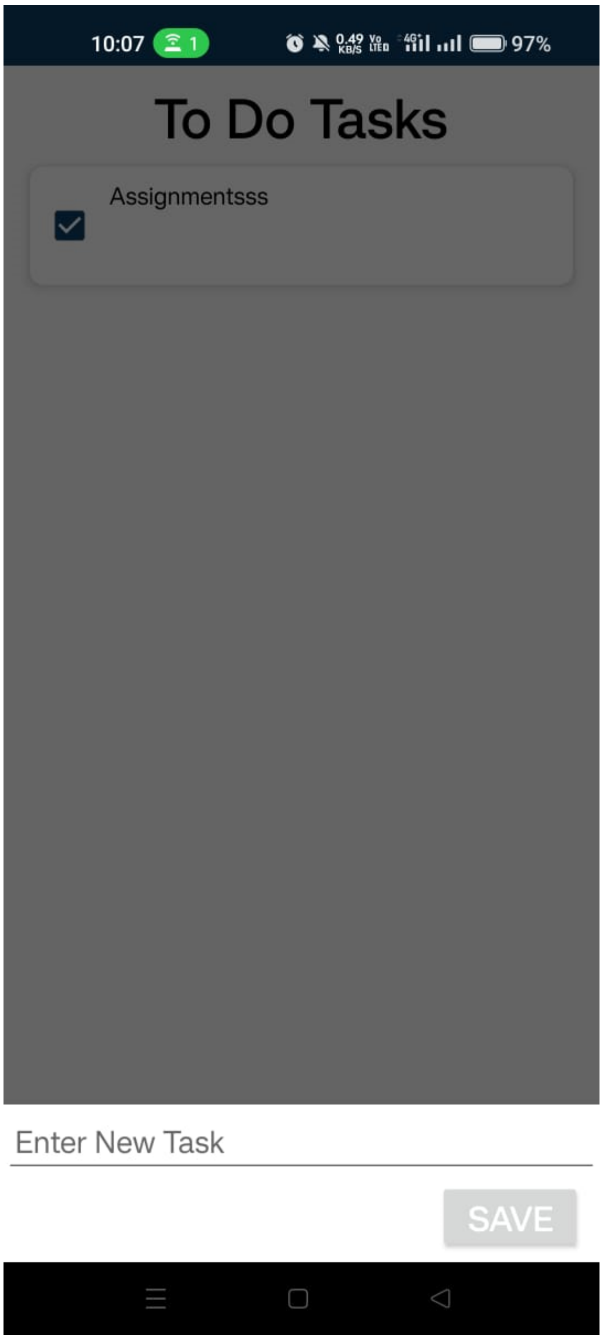


Fig 5:- adding new task

Chapter 6. Conclusion and Future Scope:

6.1 Conclusion:

In the journey of crafting the To-Do Application, we have embarked on a mission to simplify the complexities of daily tasks and enhance productivity. This application, born out of a dedication to efficiency and user-friendly design, has reached its initial stage of realization. Our path has been illuminated by innovation, determination, and the vision of providing users with a seamless task management experience. The To-Do Application, meticulously developed using Android Studio and powered by SQL operations, stands as a testament to our commitment to empowering individuals in their daily lives.

This application is not merely a digital task list; it's a reliable companion for anyone striving for organization and efficiency. Its intuitive interface, coupled with the power of SQL operations for database management, ensures tasks are managed effortlessly. The splash screen welcomes users into a world of streamlined productivity, and the intuitive user interface, featuring a RecyclerView and a Cart view, offers a visual and interactive approach to task management. The floating action button, coupled with the bottom sheet, provides a seamless way to add, edit, and delete tasks. The integration of SQL operations guarantees robust functionalities, allowing users to save, update, and delete tasks effortlessly.

6.2 Future Scope:

As we celebrate the initial success of the To-Do Application, we eagerly anticipate a future filled with innovation and enhanced user experiences. The journey doesn't end here; it's merely the beginning. Here's a glimpse of what lies ahead:

a. Expansion of Features:
We are committed to expanding the application's features to cater to a broader range of user needs. Expect additions such as task prioritization, due date reminders, and task categories, providing users with a comprehensive task management experience.

b. Collaborative Task Management:
Our vision includes the implementation of collaborative features, enabling users to share tasks, create joint to-do lists, and collaborate with colleagues, friends, and family members. This social dimension will foster teamwork and enhance productivity.

c. Intelligent Task Suggestions:
Leveraging the power of machine learning, we plan to introduce intelligent task suggestions. By analyzing user behavior and patterns, the application will provide proactive task recommendations, making task creation even more effortless.

d. Enhanced User Interface:
We are dedicated to refining the user interface further. Expect visually appealing themes, customizable widgets, and intuitive interactions, ensuring a delightful user experience and personalization options.

e. Cloud Sync and Cross-Device Compatibility:
We aim to introduce cloud synchronization, allowing users to seamlessly sync their tasks across devices. Whether on a smartphone, tablet, or desktop, users will have access to their tasks anytime,

anywhere, ensuring continuity and convenience.

f. Monetization and Premium Features:
To support continuous development, we will explore monetization avenues such as premium features, ad-free experiences, and subscription plans. These options will not only sustain the application's growth but also enhance the user experience for our valued users.

g. Accessibility and Inclusivity:
We are committed to making the To-Do Application accessible to all users, including those with disabilities. Expect features such as voice commands, screen reader compatibility, and intuitive gestures, ensuring inclusivity for every user.

References/Bibliography

1. https://bignerdranch.com/books/android-programming-the-big-nerd-ranch-guide-5th-edition/
2. https://abhiandroid.com/database/sqlite
3. https://www.digitalocean.com/community/tutorials/android-sqlite-database-example-tutorial
4. https://developer.android.com/training/wearables/apps/splash-screen
5. https://guides.codepath.com/android/using-the-recyclerview