

PROFESSIONAL TRAINING REPORT

at

Sathyabama Institute of Science and Technology

(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award
of Bachelor of Technology Degree in Information Technology

By

HAMID HUSSAIN

(Reg. No. 38120028)



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND

TECHNOLOGY

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI – 600119. TAMILNADU

JANUARY 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" Grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119.

Phone: 044 - 2450 3150 / 51 / 52 / 54 / 55 Fax: 044 - 2450 2344

www.sathyabama.ac.in



SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **HAMID HUSSAIN (Reg. No. 38120028)** who carried out the project entitled "MINIMALISTIC TO-DO LISTING APP" under our supervision from August 2020 to December 2020.

Internal Guide

Mr. Saravanan P. M.E.,

Head of the Department

Dr. R. Subhashini M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **HAMID HUSSAIN** (Reg. No. 38120028) hereby declare that the Project Report entitled "**MINIMALISTIC TO-DO LISTING APP** " done by me under the guidance of Mr.Saravanan P. M.E., at Sathyabama Institute of Science and Technology (Deemed to be University), Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600119 is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 22/01/2021

Place: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

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

I convey our thanks to **Dr. T. SASIKALA M.E., Ph.D., Dean, School of Computing** and **Dr. R. SUBHASHINI M.E., Ph.D., Head of the Department, Department of Information Technology** for providing us the necessary support and details at the right time during the progressive reviews.

I would like to express our sincere and deep sense of gratitude to our Project **Mr. Saravanan P. M.E.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express our thanks to all Teaching and Non-teaching staff members of the Department of **INFORMATION TECHNOLOGY** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE



HAMID HUSSAIN

has successfully completed the requirements to be recognized as a Microsoft Technology Associate for

Introduction to Programming using Java

Date of achievement: December 27, 2019
verify.certipoint.com 2wmC-XMhM


Satya Nadella
Chief Executive Officer

Microsoft
Technology Associate

ABSTRACT

MINIMALISTIC TO-DO LISTING APP

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 to do list just for the sake of appearances. The goal of this app is to help us become more aware of how we spend time in the process of doing those tasks and how productive that time is. The user interface is kept too simple and handy so that the distraction level would further minimize and user become more productive.

TABLE OF CONTENTS

CHAPTER NO	TITLE NAME	PAGE NO
	ABSTRACT	6
	LIST OF FIGURES	8
	LIST OF ABBREVIATION	9
1	INTRODUCTION	10
	1.1 GENERAL	10
	1.2 OUTLINE OF THE PROJECT	10
2	AIM AND SCOPE	11
	2.1 AIM	11
	2.2 PROBLEM STATEMENT	11
	2.3 SCOPE	11
3	SYSTEM ANALYSIS AND DESIGN	11
	3.1 GENERAL	11
	3.2 SOFTWARES USED	12
	3.3 ALGORITHM	13
	3.4 PROJECT DESCRIPTION	13
4	RESULT AND DISCUSSION	14
	4.1 SYSTEM REQUIREMENTS	14
	4.2 RESULT	14
5	CONCLUSION AND FUTURE WORK	14
	5.1 CONCLUSION	14
	5.2 FUTURE WORK	14
	5.3 REFERENCE	15
	APPENDIX	15
	A.SCREENSHOTS	15
	B.SOURCE CODE	17

LIST OF ABBREVIATIONS

ABBREVIATIONS

XML

SE

ME

EXPANSION

EXTENSIBLE MARKUP LANGUAGE

STANDARD EDITION

MOBILE EDITION

CHAPTER 1

INTRODUCTION

1.1 GENERAL

A ToDo Task app is where we can save schedules of some tasks to get reminded on the scheduled time. The tasks we save in the app can be saved multiple ways, online or offline. This project is designed to those people who are struggling with various complexities in their student or work life. It is based upon minimalistic approach which helps to avoid distraction while using the app and helps to use high attention on work productively.

1.2 OUTLINE OF THE PROJECT

The implementation of To Do List App would help in better productivity of users. It keeps away distractions and let you set your task easily and quickly. It uses modern tools of android development like components and Jetpack Compose which makes it more scalable in future.

CHAPTER 2

AIM AND SCOPE

2.1 AIM

The main objective of this project is to spend useful time to be more productive as a student, employee or other user.

2.2 PROBLEM STATEMENT

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.

2.3 SCOPE

The goal of this app is to help us become more aware of how we spend time in the process of doing those tasks and how productive that time is. To reduce distraction and track the time we spend working on the to-do items. When we have a better sense of the estimated time we'll need to spend on our tasks, along with the validated time spent on the items for reference or personal/team reviews, we are able to manage our daily routines more efficiently.

CHAPTER 3

SYSTEM ANALYSIS

3.1 GENERAL

3.1.1 IDE (INTEGRATED DEVELOPMENT ENVIRONMENT)

IDE stands for Integrated Development Environment. It's a coding tool which allows you to write, test, and debug your code in an easier way, as they typically offer code completion or code insight by highlighting, resource management, debugging tools.

3.1.2 USER INTERFACE

The user interface (UI) is the point of human-computer interaction and communication in a device. This can include display screens, keyboards, a mouse and the appearance of a desktop. It is also the way through which a user interacts with an application or a website.

3.2 SOFTWARES USED

3.2.1 CODE EDITORS USED

ANDROID STUDIO

Android Studio provides a unified environment where you can build apps for Android phones, tablets, Android Wear, Android TV, and Android Auto. Structured code modules allow you to divide your project into units of functionality that you can independently build, test, and debug.

3.2.2 PROGRAMMING LANGUAGES USED

JAVA

The mobile edition of Java is called Java ME (Mobile Edition). Java ME (Mobile Edition) is based on Java SE (Standard Edition) and is supported by most smartphones and tablets. The Java Platform Micro Edition (Java ME) provides a flexible, secure environment for building and executing applications that are targeted at embedded and mobile devices.

XML

XML stands for Extensible Mark-up Language. XML is a very popular format and commonly used for sharing data on the internet. This chapter explains how to parse the XML file and extract necessary information from it. Android provides three types of XML parsers which are DOM, SAX and XMLPullParser. Building an Android app comes down to two major skills/languages: Java and Android. Java is the language used in Android, but the Android part encompasses learning XML for the design of the app, learning the concepts of Android, and using the concepts programmatically with Java.

3.2.3 LIBRARIES AND FRAMEWORKS USED

JETPACK COMPOSE

Jetpack Compose is a modern toolkit designed to simplify UI development. It combines a reactive programming model with the conciseness. By making small reusable composable it's easy to build up a library of UI elements used in any app.

3.3 ALGORITHM

3.3.1 SQLIGHT DATABASE

Android SQLite is a very lightweight database which comes with Android OS. Android SQLite combines a clean SQL interface with a very small memory footprint and decent speed. Every Android application can create its own SQLite databases. SQLite is a typical relational database, containing tables (which consists of rows and columns), indexes. We can create our own tables to hold the data accordingly. This structure is referred to as a schema.

3.4 PROJECT DESCRIPTION

With social media and other easily accessible online distractions, it's more difficult than ever for us to stay focused through the course of doing our tasks. In order to handle tasks more efficiently, we must learn to manage digital distractions. This app helps us when we don't want to engage with these distractions and maintain boundaries at work. Which increases utilized amount of time.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 SYSTEM REQUIREMENTS

- **Hardware**
 - Android Phone
 - RAM – 2GB
 - Processor - Above 200 series QUALCOMM
 - ROM – 20MB
- **Software**
 - OS - Android
 - Android Version – API-16 (minimum)

4.2 RESULT

A very interactive and simple To Do app is made with Java and SQLite using android studio ide. Which is surely increasing productivity and decreases distractions and keeps focused.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

This project is undertaken using android development model and android modern libraries like Jetpack Compose and android UI Components.

5.2 FUTURE WORK

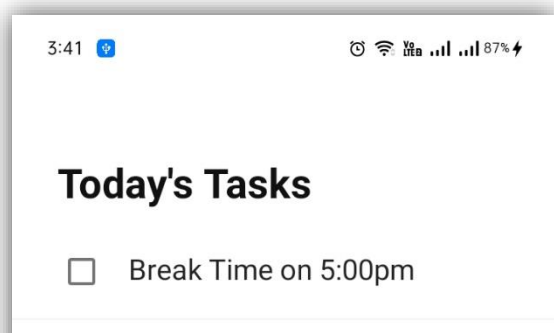
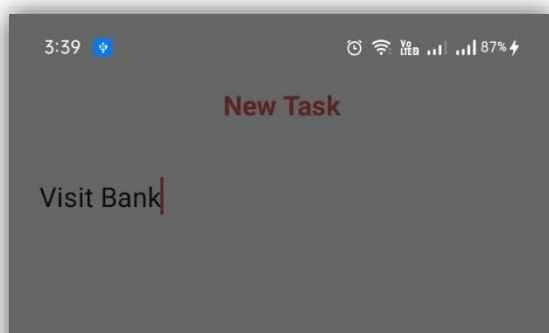
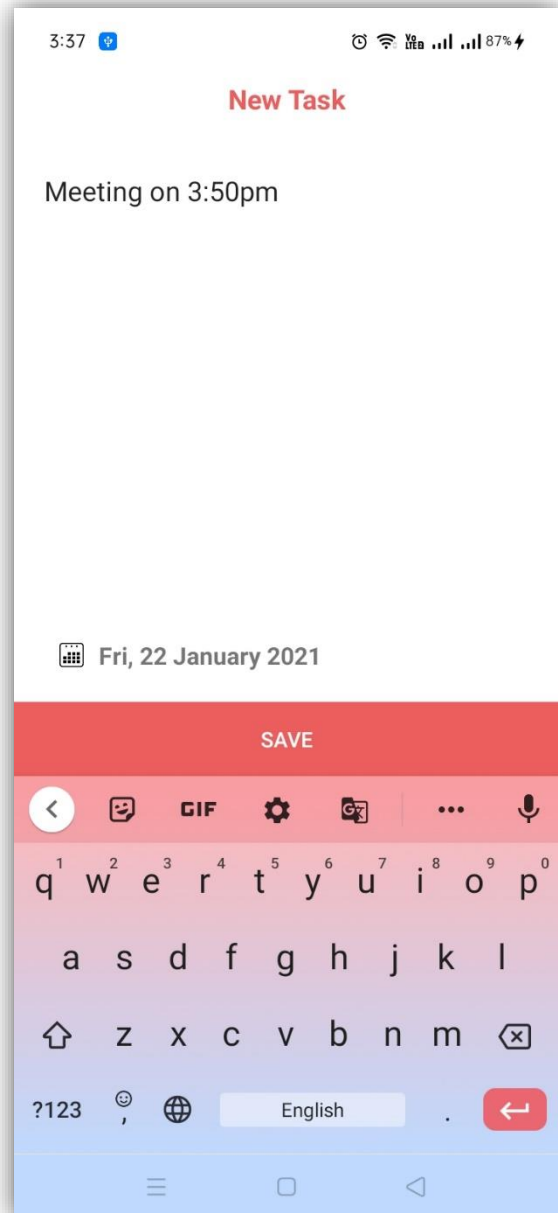
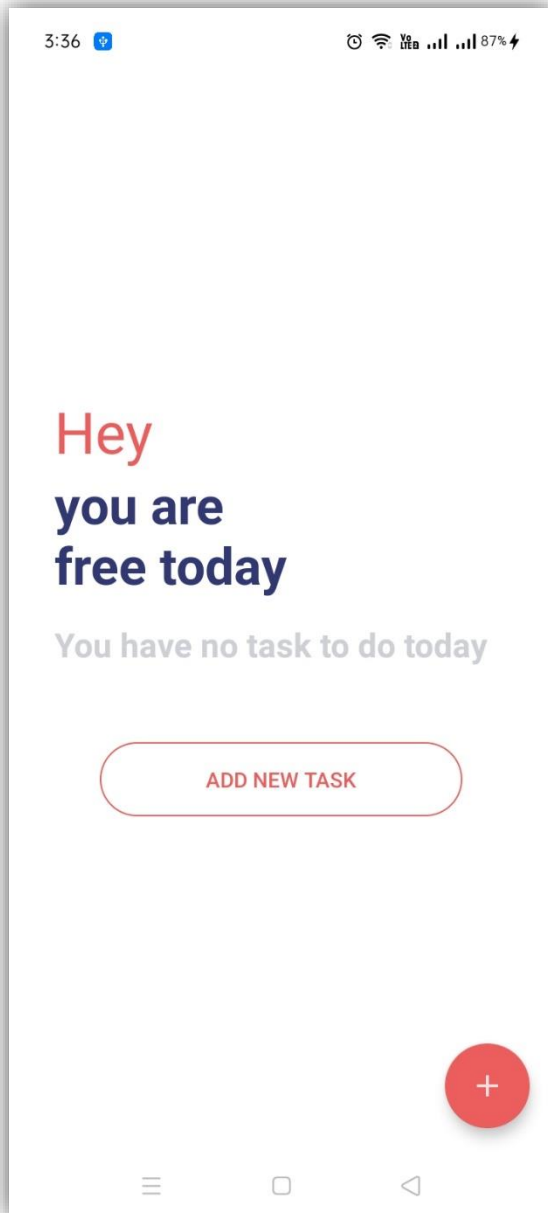
This project is made on the minimalism rule which shows the productive nature of user. It can improve by adding more features to it like timer and clock integration.

5.3 REFERENCES

1. <https://developer.android.com/>
2. <https://stackoverflow.com/>

APPENDIX

A.SCREENSHOTS



3:42



🕒 📶 5G 📶 📶 87% 🔋

Today's Tasks



Break Time on 5:00pm



Meeting on 3:50pm

Tomorrow's Tasks



Visit Bank

Upcoming Tasks



Bill Payment



B.SOURCE CODE

1. MainActivity.java

```
package com.drhhusain.doit;

import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.LinearLayout;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.widget.NestedScrollView;

import java.util.ArrayList;
import java.util.HashMap;

public class MainActivity extends AppCompatActivity {

    DBHelpermydb;
    LinearLayout empty;
    NestedScrollViewscrollView;
    LinearLayouttodayContainer, tomorrowContainer, otherContainer;
    NoScrollListViewtaskListToday, taskListTomorrow, taskListUpcoming;
    ArrayList<HashMap<String, String>>todayList = new ArrayList<HashMap<String, String>>();
    ArrayList<HashMap<String, String>>tomorrowList = new ArrayList<HashMap<String,
    String>>();
    ArrayList<HashMap<String, String>>upcomingList = new ArrayList<HashMap<String,
    String>>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        getWindow().getDecorView().setSystemUiVisibility(View.SYSTEM_UI_FLAG_LIGHT_STAT
        US_BAR);
        setContentView(R.layout.activity_main);

        mydb = new DBHelper(this);
        empty = findViewById(R.id.empty);
        scrollView = findViewById(R.id.scrollView);
        todayContainer = findViewById(R.id.todayContainer);
        tomorrowContainer = findViewById(R.id.tomorrowContainer);
        otherContainer = findViewById(R.id.otherContainer);
        taskListToday = findViewById(R.id.taskListToday);
        taskListTomorrow = findViewById(R.id.taskListTomorrow);
        taskListUpcoming = findViewById(R.id.taskListUpcoming);
    }
}
```

```
    public void openAddModifyTask(View view) {  
startActivity(new Intent(this, AddModifyTask.class));  
    }
```

```
    @Override  
    public void onResume() {  
super.onResume();  
populateData();  
    }
```

```
    public void populateData() {  
mydb = new DBHelper(this);
```

```
runOnUiThread(new Runnable() {  
    public void run() {  
fetchDataFromDB();  
    }  
});  
}
```

```
    public void fetchDataFromDB() {  
todayList.clear();  
tomorrowList.clear();  
upcomingList.clear();
```

```
        Cursor today = mydb.getTodayTask();  
        Cursor tomorrow = mydb.getTomorrowTask();  
        Cursor upcoming = mydb.getUpcomingTask();
```

```
loadDataList(today, todayList);  
loadDataList(tomorrow, tomorrowList);  
loadDataList(upcoming, upcomingList);
```

```
        if (todayList.isEmpty() &&tomorrowList.isEmpty() &&upcomingList.isEmpty()) {  
empty.setVisibility(View.VISIBLE);  
scrollView.setVisibility(View.GONE);  
        } else {  
empty.setVisibility(View.GONE);  
scrollView.setVisibility(View.VISIBLE);
```

```
        if (todayList.isEmpty()) {  
todayContainer.setVisibility(View.GONE);  
        } else {  
todayContainer.setVisibility(View.VISIBLE);  
loadListView(taskListToday, todayList);  
        }
```

```
        if (tomorrowList.isEmpty()) {  
tomorrowContainer.setVisibility(View.GONE);  
        } else {
```

```

tomorrowContainer.setVisibility(View.VISIBLE);
loadListView(taskListTomorrow, tomorrowList);
    }

    if (upcomingList.isEmpty()) {
otherContainer.setVisibility(View.GONE);
    } else {
otherContainer.setVisibility(View.VISIBLE);
loadListView(taskListUpcoming, upcomingList);
    }
}
}

public void loadDataList(Cursor cursor, ArrayList<HashMap<String, String>>dataList) {
    if (cursor != null) {
cursor.moveToFirst();
        while (cursor.isAfterLast() == false) {

            HashMap<String, String>mapToday = new HashMap<String, String>();
mapToday.put("id", cursor.getString(0).toString());
mapToday.put("task", cursor.getString(1).toString());
mapToday.put("date", cursor.getString(2).toString());
mapToday.put("status", cursor.getString(3).toString());
dataList.add(mapToday);
cursor.moveToNext();
        }
    }
}

public void loadListView(NoScrollListViewlistView, final ArrayList<HashMap<String,
String>>dataList) {
ListTaskAdapter adapter = new ListTaskAdapter(this, dataList, mydb);
listView.setAdapter(adapter);
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    public void onItemClick(AdapterView<?> parent, View view,
        int position, long id) {
        Intent i = new Intent(MainActivity.this, AddModifyTask.class);
i.putExtra("isModify", true);
i.putExtra("id", dataList.get(+position).get("id"));
startActivity(i);
    }
});
}
}

```

2. DBHelper.java

```

package com.drhusain.doit;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;

```

```

import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import java.util.Date;

public class DBHelper extends SQLiteOpenHelper {

    public static final String DATABASE_NAME = "ToDoDBHelper.db";
    public static final String TABLE_NAME = "todo";

    public DBHelper(Context context) {
        super(context, DATABASE_NAME, null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // TODO Auto-generated method stub

        db.execSQL(
            "CREATE TABLE " + TABLE_NAME + "(id INTEGER PRIMARY KEY, task TEXT, "
            + "task_at DATETIME, status INTEGER)"
        );
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // TODO Auto-generated method stub
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
    }

    public boolean insertTask(String task, String task_at) {
        Date date;
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put("task", task);
        values.put("task_at", task_at);
        //values.put("dateStr", getDate(dateStr));
        values.put("status", 0);
        db.insert(TABLE_NAME, null, values);
        return true;
    }

    public boolean updateTask(String id, String task, String task_at) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();

        values.put("task", task);
        values.put("task_at", task_at);

        db.update(TABLE_NAME, values, "id = ? ", new String[]{id});
        return true;
    }
}

```

```

    public boolean deleteTask(String id) {
        SQLiteDatabase db = this.getWritableDatabase();
        db.delete(TABLE_NAME, "id = ? ", new String[]{id});
        return true;
    }

    public boolean updateTaskStatus(String id, Integer status) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();

        contentValues.put("status", status);

        db.update(TABLE_NAME, contentValues, "id = ? ", new String[]{id});
        return true;
    }

    public Cursor getSingleTask(String id) {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor res = db.rawQuery("select * from " + TABLE_NAME + " WHERE id = '" + id + "' order by id desc", null);
        return res;
    }

    public Cursor getTodayTask() {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor res = db.rawQuery("select * from " + TABLE_NAME + " WHERE date(task_at) = date('now', 'localtime') order by id desc", null);
        return res;
    }

    public Cursor getTomorrowTask() {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor res = db.rawQuery("select * from " + TABLE_NAME + " WHERE date(task_at) = date('now', '+1 day', 'localtime') order by id desc", null);
        return res;
    }

    public Cursor getUpcomingTask() {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor res = db.rawQuery("select * from " + TABLE_NAME + " WHERE date(task_at) > date('now', '+1 day', 'localtime') order by id desc", null);
        return res;
    }
}

```

3. AddModifyTask.java

```
package com.drhhusain.doit;

import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.GregorianCalendar;

public class AddModifyTask extends AppCompatActivity {

    Calendar calendar;
    DBHelpermydb;

    Boolean isModify = false;
    String task_id;
    TextViewtoolbar_title;
    EditTextedit_text;
    TextViewdateText;
    Button save_btn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        getWindow().getDecorView().setSystemUiVisibility(View.SYSTEM_UI_FLAG_LIGHT_STATUS_BAR);
        setContentView(R.layout.activity_add_modify_task);

        mydb = new DBHelper(getApplicationContext());
        calendar = new GregorianCalendar();
        toolbar_title = findViewById(R.id.toolbar_title);
        edit_text = findViewById(R.id.edit_text);
        dateText = findViewById(R.id.dateText);
        save_btn = findViewById(R.id.save_btn);

        dateText.setText(new SimpleDateFormat("E, dd MMMM yyyy").format(calendar.getTime()));
```

```

        Intent intent = getIntent();
        if (intent.hasExtra("isModify")) {
            isModify = intent.getBooleanExtra("isModify", false);
            task_id = intent.getStringExtra("id");
            init_modify();
        }

    }

    public void init_modify() {
        toolbar_title.setText("Modify Task");
        save_btn.setText("Update");
        LinearLayoutdeleteTask = findViewById(R.id.deleteTask);
        deleteTask.setVisibility(View.VISIBLE);
        Cursor task = mydb.getSingleTask(task_id);
        if (task != null) {
            task.moveToFirst();

            edit_text.setText(task.getString(1));

            SimpleDateFormat iso8601Format = new SimpleDateFormat("yyyy-MM-dd");
            try {
                calendar.setTime(iso8601Format.parse(task.getString(2)));
            } catch (ParseException e) {
            }

            dateText.setText(new SimpleDateFormat("E, dd MMMM yyyy").format(calendar.getTime()));

        }

    }

    public void saveTask(View v) {

        /*Checking for Empty Task*/
        if (edit_text.getText().toString().trim().length() > 0) {

            if (isModify) {
                mydb.updateTask(task_id, edit_text.getText().toString(), new SimpleDateFormat("yyyy-MM-dd").format(calendar.getTime()));
                Toast.makeText(getApplicationContext(), "Task Updated.", Toast.LENGTH_SHORT).show();
            } else {
                mydb.insertTask(edit_text.getText().toString(), new SimpleDateFormat("yyyy-MM-dd").format(calendar.getTime()));
                Toast.makeText(getApplicationContext(), "Task Added.", Toast.LENGTH_SHORT).show();
            }
            finish();

        } else {

```

```

Toast.makeText(getApplicationContext(), "Empty task can't be saved.",
Toast.LENGTH_SHORT).show();
    }

}

    public void deleteTask(View v) {
mydb.deleteTask(task_id);
Toast.makeText(getApplicationContext(), "Task Removed",
Toast.LENGTH_SHORT).show();
finish();
    }

    public void chooseDate(View view) {
        final View dialogView = View.inflate(this, R.layout.date_picker, null);
        final DatePicker datePicker = dialogView.findViewById(R.id.date_picker);
        datePicker.updateDate(calendar.get(Calendar.YEAR), calendar.get(Calendar.MONTH),
calendar.get(Calendar.DAY_OF_MONTH));

        final AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setView(dialogView);
        builder.setTitle("Choose Date");
        builder.setNegativeButton("Cancel", null);
        builder.setPositiveButton("Set", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {

                calendar = new GregorianCalendar(datePicker.getYear(), datePicker.getMonth(),
datePicker.getDayOfMonth());
                dateText.setText(new SimpleDateFormat("E, dd MMMM yyyy").format(calendar.getTime()));

            }
        });
        builder.show();
    }
}

```

4. ListTaskAdapter.java

```

package com.drhusain.doit;

import android.app.Activity;
import android.text.Html;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.TextView;

```

```

import java.util.ArrayList;
import java.util.HashMap;

public class ListTaskAdapter extends BaseAdapter {
    private Activity activity;
    private ArrayList<HashMap<String, String>> data;
    private DBHelper database;

    public ListTaskAdapter(Activity a, ArrayList<HashMap<String, String>> d, DBHelpermydb)
    {
        activity = a;
        data = d;
        database = mydb;
    }

    public int getCount() {
        return data.size();
    }

    public Object getItem(int position) {
        return position;
    }

    public long getItemId(int position) {
        return position;
    }

    public View getView(int position, View convertView, ViewGroup parent) {
        ListTaskViewHolder holder = null;
        if (convertView == null) {
            holder = new ListTaskViewHolder();
            convertView = LayoutInflater.from(activity).inflate(R.layout.task_list_row, parent, false);
            holder.task_name = convertView.findViewById(R.id.task_name);
            holder.checkBtn = convertView.findViewById(R.id.checkBtn);
            convertView.setTag(holder);
        } else {
            holder = (ListTaskViewHolder) convertView.getTag();
        }

        final HashMap<String, String>singleTask = data.get(position);
        final ListTaskViewHoldertmpHolder = holder;

        holder.task_name.setId(position);
        holder.checkBtn.setId(position);

        try {

            holder.checkBtn.setOnCheckedChangeListener(null);
            if (singleTask.get("status").contentEquals("1")) {
                holder.task_name.setText(Html.fromHtml("<strike>" + singleTask.get("task") + "</strike>"));
                holder.checkBtn.setChecked(true);
            } else {

```

```

holder.task_name.setText(singleTask.get("task"));
holder.checkBtn.setChecked(false);
    }

holder.checkBtn.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)
    {
        if (isChecked) {
            database.updateTaskStatus(singleTask.get("id"), 1);
            tmpHolder.task_name.setText(Html.fromHtml("<strike>" + singleTask.get("task") +
            "</strike>"));
        } else {
            database.updateTaskStatus(singleTask.get("id"), 0);
            tmpHolder.task_name.setText(singleTask.get("task"));
        }
    }
});

    } catch (Exception e) {
    }
    return convertView;
}
}

class ListTaskViewHolder {
    TextView task_name;
    CheckBox checkBtn;
}

```

5. NoScrollListView.java

```

package com.drhusain.doit;

import android.content.Context;
import android.util.AttributeSet;
import android.view.ViewGroup;
import android.widget.ListView;

// creates a list view with scrolling disabled
public class NoScrollListView extends ListView {

    public NoScrollListView(Context context) {
        super(context);
    }
    public NoScrollListView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }
    public NoScrollListView(Context context, AttributeSet attrs, int defStyle) {

```

```
super(context, attrs, defStyle);
    }
    @Override
    public void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        int heightMeasureSpec_custom = MeasureSpec.makeMeasureSpec(
Integer.MAX_VALUE>> 2, MeasureSpec.AT_MOST);
super.onMeasure(widthMeasureSpec, heightMeasureSpec_custom);
ViewGroup.LayoutParams params = getLayoutParams();
params.height = getMeasuredHeight();
    }
}
```