**PROJECT REPORT ON**

**“ TASK TIMER ”**

**Android Application Project**


**Prepared by**

SANA AFREEN

# Index

# ACKNOWLEDGMENT

It is a great pleasure to present this report on the topic named “Android App Development” undertaken by me as part of my B. Tech curriculum.

It is a pleasure that I find myself penning down these lines to express my sincere thanks to those people who helped me along the way in completing my project. I find inadequate words to express my sincere gratitude towards them.

First and foremost I would like to express my gratitude towards our training guide Mr. Shailendra Singh for placing complete faith and confidence in my ability to carry out this project and for providing me his time, inspiration, encouragement, help, and constant interest.

The internship on “Android app development” was very helpful to me in giving the necessary background information and inspiration about the emerging field of machine learning and data science.

Their contribution and technical support in preparing this report are greatly acknowledged.

**Sana Afreen**

# INTRODUCTION

Task Timer is software in the category of Task Management, Project Management, Productivity, “Getting Things Done” (GTD), Scheduling, and Collaboration.

We have a lot of choices to help us keep track of daily obligations. A simple list on paper of things “Task Timer or task done with Timer” is enough for some people. Others prefer to use programs. People have preferences about where they keep track of tasks - with PC utilities (“thick app”), websites (“thin app”), on phone (“mobile”) apps. Some software is very general, with nothing more than a simple list of task names and due dates. Some get more sophisticated with nested tasks, where you need to complete the nested tasks in order to consider the main task complete. Some commercial software is Very sophisticated, and costs a lot - look at Microsoft Project and similar offerings. Some software is focused on specific industries, for construction, manufacturing, business consulting, website management, wedding planning, or even cooking recipes.

Task Timer is general-purpose, Windows-based software,Combined with powerful reporting mechanisms, this makes Done task in given time an effective tool for clients.

What differentiates Task Timer from most other task management packages is the option to create hierarchies of tasks to break complex tasks down in time to progressively more simple subtasks until no further simplification is required. Each task can be scheduled with in time, a priority and various other attributes. As each task is completed the parent tasks are updated in various ways to keep you informed of overall progress and your can delete the completed task.

Task Timer is a free open source software (FOSS).

## SYSTEM USED FOR PROJECT DEVELOPMENT

- Hardware used:-
  - Intel Core i3-1115G4 @ 3.00GHz  2.90 GHz
  - 8 GB DDR4 RAM
  - Intel HD Graphics 5500

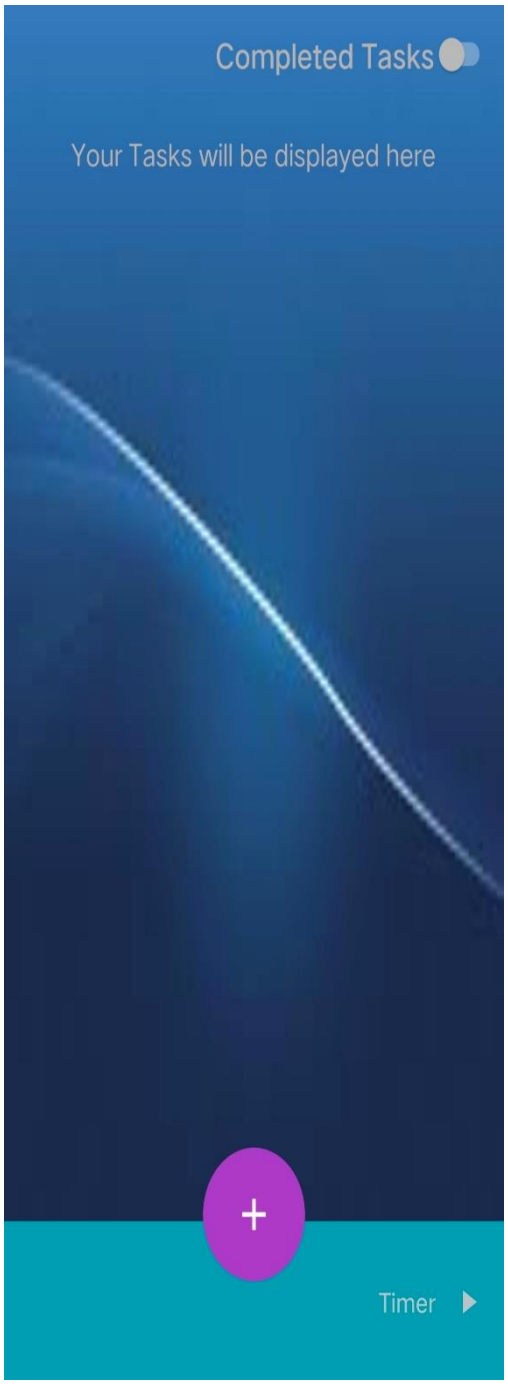
- Software used:-
  - Java SDK,JDK
  - Android studio : Android Studio Chipmunk | 2021.2.1

## Project work

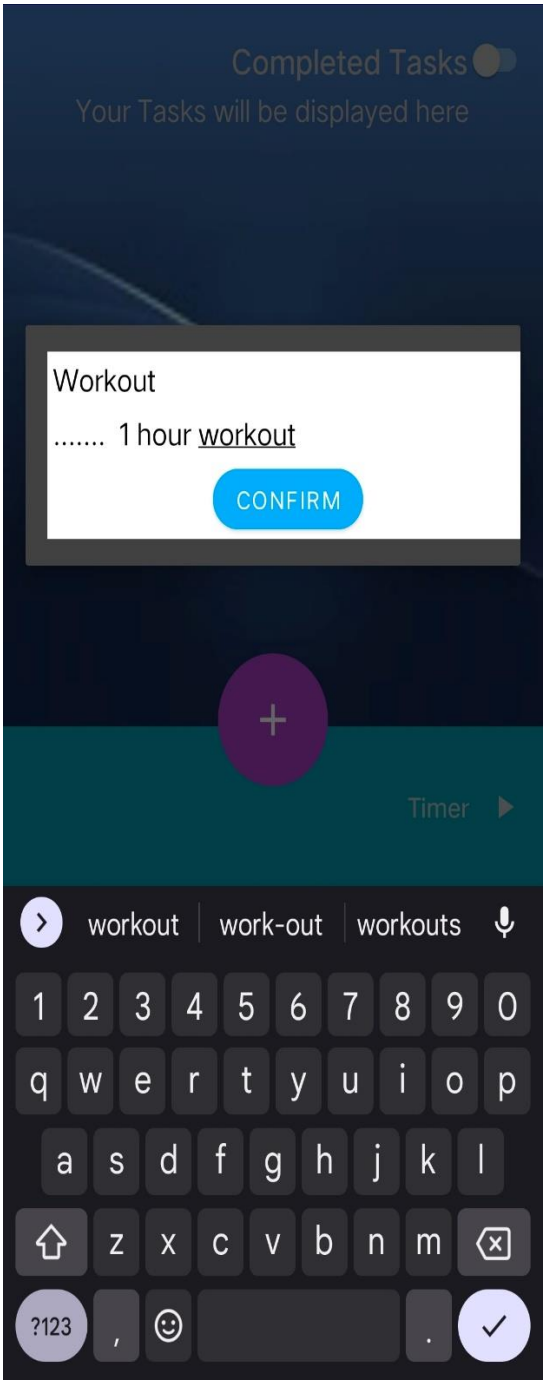
In this training, a  project " Task Timer" app has been created for the understanding of the android studio and its application completely. This project helps in understanding how the Android studio, its classes and system services work parallel to each other.

**Application Icon:**

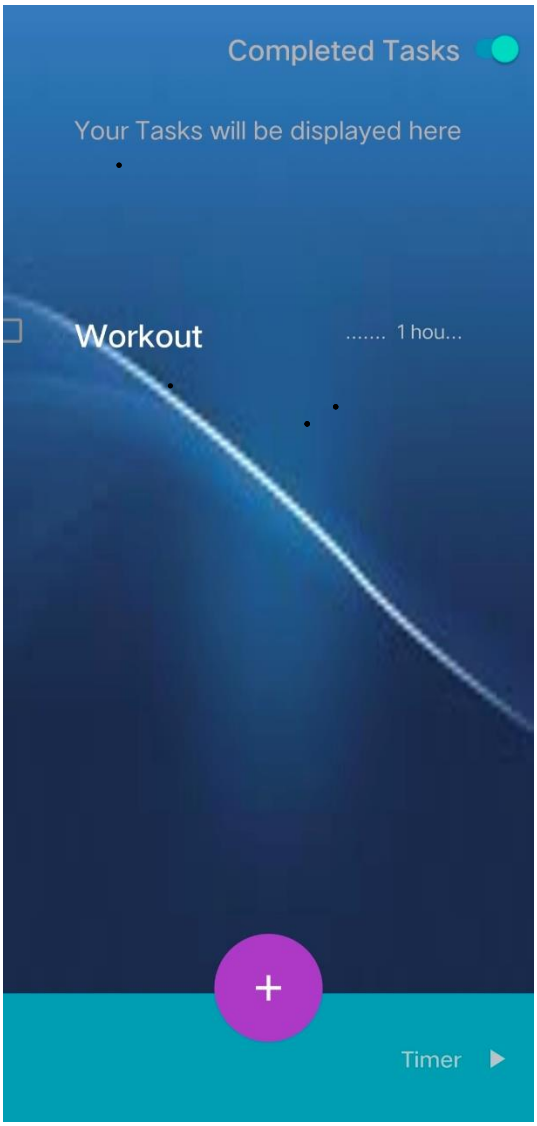
**Home Screen**



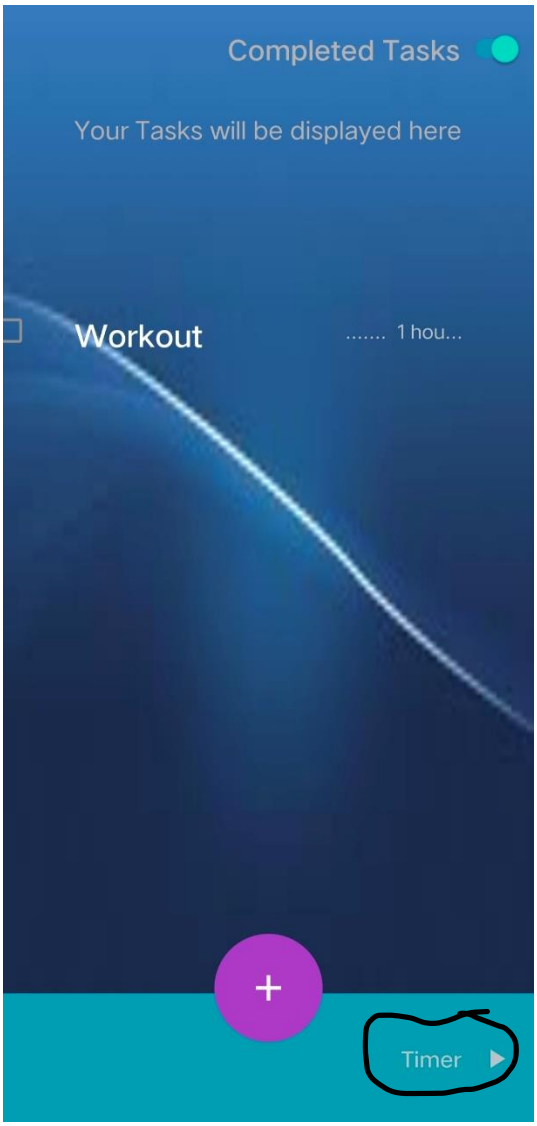

**Adding Task**



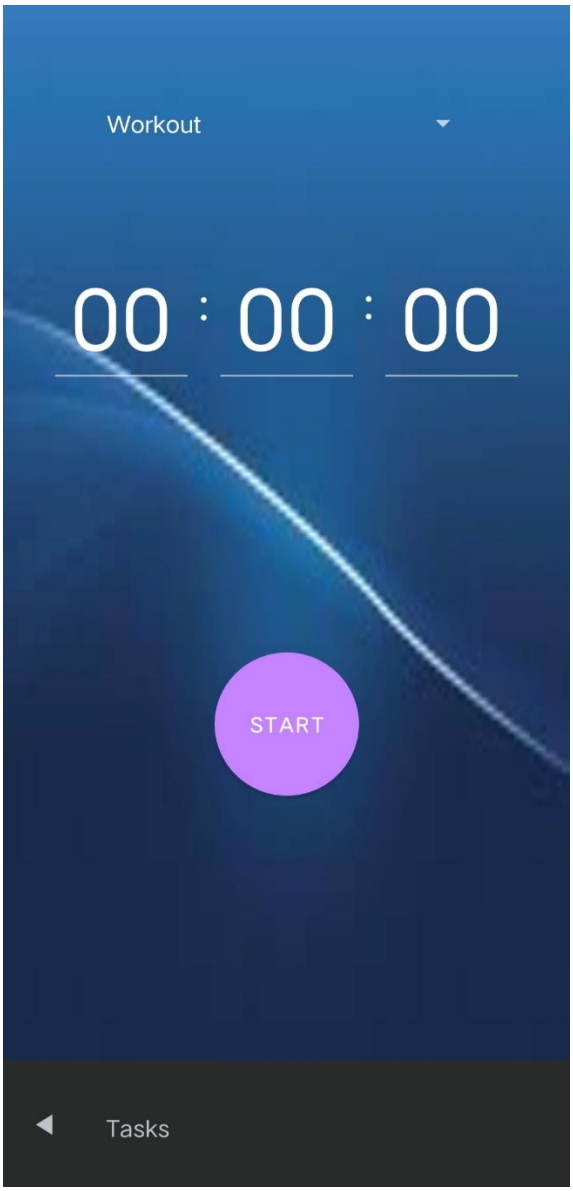
**Display of task in home screen**



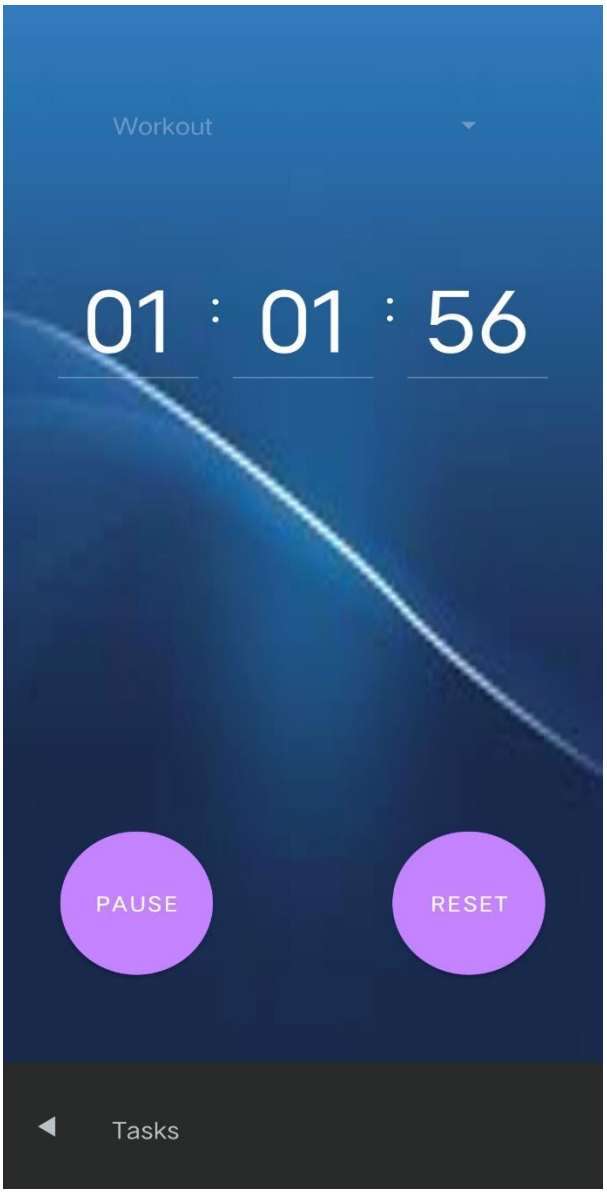

**Select timer option to set time**



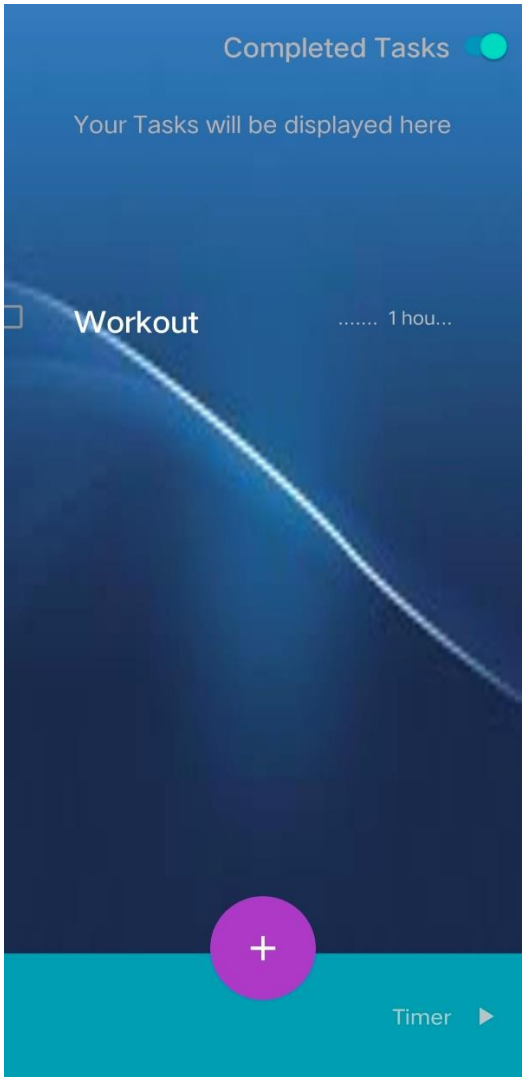
**Timer screen**



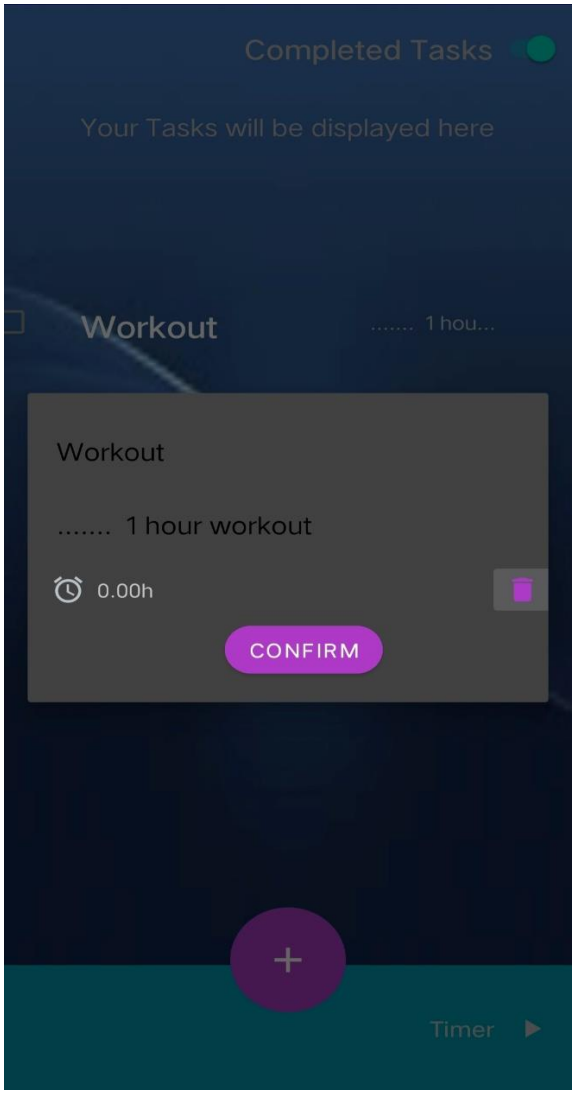

**Timer setting**



**Enable Task completion**




**Deletion of completed task**



## MainActivity.java

```
 package com.example.studyapp.activities;

import android.animation.ObjectAnimator;
import android.app.Dialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.FragmentManager;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProviders;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.studyapp.R;
import com.example.studyapp.databinding.ActivityMainBinding;
import com.example.studyapp.taskdb.Task;
import com.example.studyapp.taskdb.TaskListAdapter;
import com.example.studyapp.taskdb.TaskViewModel;

import java.io.Serializable;
import java.util.Arrays;
import java.util.List;

public class MainActivity extends AppCompatActivity implements TaskListAdapter.OnTaskClickListener{

    private ActivityMainBinding binding;
    private TaskViewModel mTaskViewModel;
    FragmentManager fragmentManager;
    TaskListAdapter adapter;
    SharedPreferences pref;
    SharedPreferences.Editor prefEdit;
```

```
String[] titles;

ViewGroup allViews;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding = ActivityMainBinding.inflate(getLayoutInflater());
    View view = binding.getRoot();
    setContentView(view);
    getSupportActionBar().hide();

    allViews = (ViewGroup) view;

    fragmentManager = getSupportFragmentManager();

    pref = getSharedPreferences(getString(R.string.timer_prefs), Context.MODE_PRIVATE);
    prefEdit = pref.edit();


    //Timer Button Listener
    ImageButton timerButton = binding.timerButton;
    timerButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            //Start timer fragment
            Intent timerIntent = new Intent(MainActivity.this, TimerActivity.class);
            timerIntent.putExtra("SpinnerList", (Serializable) adapter.mTasks);
            startActivity(timerIntent);
        }
    });

    //Create Button Listener
    ImageButton createButton = binding.createButton;
    createButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            showCreateTaskDialog();
            //Task testTask = new Task("Test Task", "", "", 0.00f);
            //mTaskViewModel.insert(testTask);
        }
    });

    //Recycler View Adapter
    mTaskViewModel = ViewModelProviders.of(this).get(TaskViewModel.class);


    RecyclerView recyclerView = binding.taskRecycler;
    adapter = new TaskListAdapter(this, this);
    recyclerView.setAdapter(adapter);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));
```

```
    mTaskViewModel.getUncompletedTasks().observe(MainActivity.this, new Observer<List<Task>>() {
        @Override
        public void onChanged(List<Task> tasks) {
            adapter.setTasks(tasks);
        }
    });

    Switch completedSwitch = binding.completedSwitch;

    completedSwitch.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
            if (b) {
                mTaskViewModel.getAllTasks().observe(MainActivity.this, new Observer<List<Task>>() {
                    @Override
                    public void onChanged(List<Task> tasks) {
                        adapter.setTasks(tasks);
                    }
                });
            } else {
                mTaskViewModel.getUncompletedTasks().observe(MainActivity.this, new Observer<List<Task>>() {
                    @Override
                    public void onChanged(List<Task> tasks) {
                        adapter.setTasks(tasks);
                    }
                });
            }
        }
    });



}

@Override
protected void onStart() {
    super.onStart();

    ObjectAnimator fadeIn;

    for(int i = 0; i < allViews.getChildCount(); i++) {
        allViews.getChildAt(i);
        fadeIn = ObjectAnimator.ofFloat(allViews.getChildAt(i), "alpha", 0f, 1f);
        fadeIn.setDuration(750);
        fadeIn.start();
    }
}

private void showCreateTaskDialog() {
    //TODO: Find a way to use view binding here
    final Dialog dialog = new Dialog(MainActivity.this);
```

```
        dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
        dialog.setCancelable(true);
        dialog.setContentView(R.layout.task_create_dialog);

        EditText taskTitle = dialog.findViewById(R.id.titleEditText);
        EditText taskDescription = dialog.findViewById(R.id.descriptionEditText);
        Button confirmButton = dialog.findViewById(R.id.dialogButton);

        confirmButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                String taskTitleString;
                String taskDescriptionString;
                titles = new String[adapter.mTasks.size()];

                for(int i = 0; i < adapter.mTasks.size(); i++) {
                    titles[i] = adapter.mTasks.get(i).getTitle();
                }

                //Check if fields have been filled and assign values
                if(taskTitle.getText().toString().matches("")) {
                    Toast.makeText(getApplicationContext(), "Title is required", Toast.LENGTH_SHORT).show();
                    return;
                }
                taskTitleString = taskTitle.getText().toString();

                if(Arrays.asList(titles).contains(taskTitleString)) {
                    Toast.makeText(getApplicationContext(), "A task with this title already exists",
Toast.LENGTH_SHORT).show();
                    return;
                }

                taskDescriptionString = taskDescription.getText().toString();

                Task newTask = new Task(taskTitleString, taskDescriptionString, 0.00f, 0);
                mTaskViewModel.insert(newTask);
                dialog.dismiss();
            }
        });



        dialog.show();


    }

    private void showTaskDescriptionDialog(Task current) {
        final Dialog dialog = new Dialog(MainActivity.this);
```

```
        dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
        dialog.setCancelable(true);
        dialog.setContentView(R.layout.task_description_dialog);

        EditText taskTitle = dialog.findViewById(R.id.titleEditText);
        EditText taskDescription = dialog.findViewById(R.id.descriptionEditText);
        TextView descriptionTimeText = dialog.findViewById(R.id.descriptionTimeText);
        Button confirmButton = dialog.findViewById(R.id.dialogButton);
        ImageButton deleteButton = dialog.findViewById(R.id.deleteButton);

        taskTitle.setText(current.getTitle());
        taskDescription.setText(current.getDescription());


        if(pref.contains(current.getTitle())) {
            float newTime = pref.getFloat(current.getTitle(), 0.00f);
            String newTimeString = String.format("%.2f", newTime);
            descriptionTimeText.setText(newTimeString + "h");
        }

        confirmButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                String taskTitleString;
                String taskDescriptionString;

                titles = new String[adapter.mTasks.size()];

                //Check if fields have been filled and assign values
                if(taskTitle.getText().toString().matches("")) {
                    Toast.makeText(getApplicationContext(), "Title is required", Toast.LENGTH_SHORT).show();
                    return;
                }
                taskTitleString = taskTitle.getText().toString();

                if(Arrays.asList(titles).contains(taskTitleString)) {
                    Toast.makeText(getApplicationContext(), "A task with this title already exists",
Toast.LENGTH_SHORT).show();
                    return;
                }

                taskDescriptionString = taskDescription.getText().toString();

                Task updatedTask = new Task(current.getId(), taskTitleString, taskDescriptionString,
current.getTimeProgress(), current.getIsComplete());
                mTaskViewModel.update(updatedTask);
                dialog.dismiss();
            }
        });

        deleteButton.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View view) {
            Log.d("title", "Title: " + current.getTitle());
            mTaskViewModel.delete(current);

            prefEdit.remove(current.getTitle()).apply();

            dialog.dismiss();
        }
    });

    dialog.show();

}

@Override
public void onTaskClick(Task current) {
    showTaskDescriptionDialog(current);
}

@Override
public void onCheckClick(Task current, int newCompleted) {
    Task newCheckTask = new Task(current.getId(), current.getTitle(), current.getDescription(),
current.getTimeProgress(), newCompleted);
    mTaskViewModel.update(newCheckTask);
}


}
```

## TimerActivity.java

```
package com.example.studyapp.activities;

import android.animation.ObjectAnimator;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Build;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.Spinner;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.NotificationCompat;

import com.example.studyapp.R;
import com.example.studyapp.databinding.ActivityTimerBinding;
import com.example.studyapp.taskdb.Task;

import java.util.List;

public class TimerActivity extends AppCompatActivity {

    private ActivityTimerBinding binding;
    private EditText hourText, minuteText, secondsText;
    private String hourString, minutesString, secondsString;
    private Integer hours, minutes, seconds;
    private long startTimeInMillis, timeLeftInMillis, timeElapsedInMillis, endTime;
    private float rHours, savedTime;
    private Button startButton, pauseButton, resetButton;
    private CountDownTimer countDownTimer;
    private Spinner timerSpinner;
    private boolean timerRunning;
    private String selectedTask;

    private static final String CHANNEL_ID = "timer_channel";
    private NotificationManager notificationManager;
```

```
ViewGroup allViews;

SharedPreferences pref;
SharedPreferences.Editor prefEdit;
SharedPreferences saveTimePrefs;
SharedPreferences.Editor saveTimePrefsEdit;

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   binding = ActivityTimerBinding.inflate(getLayoutInflater());
   View view = binding.getRoot();
   setContentView(view);
   getSupportActionBar().hide();

   allViews = (ViewGroup) view;

   create_notification_channel();

   pref = getSharedPreferences(getString(R.string.timer_prefs), Context.MODE_PRIVATE);
   prefEdit = pref.edit();

   saveTimePrefs = getSharedPreferences(getString(R.string.save_time_prefs), Context.MODE_PRIVATE);
   saveTimePrefsEdit = saveTimePrefs.edit();

   //Initialize buttons & edit texts
   startButton = binding.startButton;
   pauseButton = binding.pauseButton;
   resetButton = binding.resetButton;

   hourText = binding.hourText;
   minuteText = binding.minuteText;
   secondsText = binding.secondsText;
   timerSpinner = binding.spinner;


   //Timer button listeners
   startButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
         if(timerSpinner.getSelectedItem() == null) {
            Toast.makeText(TimerActivity.this, "Please select a task", Toast.LENGTH_SHORT).show();
            return;
         }

         selectedTask = timerSpinner.getSelectedItem().toString();

         hourString = hourText.getText().toString();
         minutesString = minuteText.getText().toString();
         secondsString = secondsText.getText().toString();

         if(hourString == "" || minutesString == "" || secondsString == "") {
```

```
            Toast.makeText(TimerActivity.this, "Error with time input", Toast.LENGTH_SHORT).show();
            return;
        }

        hours = Integer.parseInt(hourString);
        minutes = Integer.parseInt(minutesString);
        seconds = Integer.parseInt(secondsString);

        if(hours == 0 && minutes == 0 && seconds == 0) {
            Toast.makeText(getApplicationContext(), "Please enter a value", Toast.LENGTH_LONG).show();
            return;
        } else if(minutes > 60 || seconds > 60) {
            Toast.makeText(getApplicationContext(), "Minute or seconds value is too high",
Toast.LENGTH_LONG).show();
            return;
        }

        timeLeftInMillis = calculateStartTime(hours, minutes, seconds);
        startTimeInMillis = timeLeftInMillis;

        startButton.setVisibility(View.INVISIBLE);
        pauseButton.setVisibility(View.VISIBLE);
        resetButton.setVisibility(View.VISIBLE);

        hourText.setEnabled(false);
        minuteText.setEnabled(false);
        secondsText.setEnabled(false);

        startTimer();
      }
    });



    pauseButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        if(timerRunning) {
          pauseTimer();

          pauseButton.setText("Resume");
        } else {
          startTimer();

          pauseButton.setText("Pause");
        }

      }
    });

    resetButton.setOnClickListener(new View.OnClickListener() {
      @Override
```

```
        public void onClick(View view) {
            resetTimer();
        }
    });

    //Task Button Listener
    ImageButton taskButton = binding.taskButton;
    taskButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            startActivity(new Intent(TimerActivity.this, MainActivity.class));
        }
    });
}

@Override
protected void onResume() {
    super.onResume();

    Intent mainIntent = getIntent();
    List<Task> spinnerList = (List<Task>) mainIntent.getSerializableExtra("SpinnerList");

    String[] spinnerTitleArray = new String[spinnerList.size()];

    for(int i = 0; i < spinnerList.size(); i++) {
        spinnerTitleArray[i] = spinnerList.get(i).getTitle();
    }



    ArrayAdapter<String> spinnerAdapter = new ArrayAdapter<String>(this,
android.R.layout.simple_spinner_item, spinnerTitleArray);
    spinnerAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

    timerSpinner.setAdapter(spinnerAdapter);
}

@Override
protected void onStop() {
    super.onStop();

    saveTimePrefsEdit.putLong("millisLeft", timeLeftInMillis);
    saveTimePrefsEdit.putBoolean("timerRunning", timerRunning);
    saveTimePrefsEdit.putLong("endTime", endTime);

    saveTimePrefsEdit.apply();
}

@Override
protected void onStart() {
    super.onStart();
```

```
    ObjectAnimator fadeIn;

    for(int i = 0; i < allViews.getChildCount(); i++) {
        allViews.getChildAt(i);
        fadeIn = ObjectAnimator.ofFloat(allViews.getChildAt(i), "alpha", 0f, 1f);
        fadeIn.setDuration(750);
        fadeIn.start();
    }

    timeLeftInMillis = saveTimePrefs.getLong("millisLeft", 0);
    timerRunning = saveTimePrefs.getBoolean("timerRunning", false);

    if(timerRunning) {
        endTime = saveTimePrefs.getLong("endTime", 0);
        timeLeftInMillis = endTime - System.currentTimeMillis();

        if(timeLeftInMillis < 0) {
            timeLeftInMillis = 0;
            timerRunning = false;
            updateCountDownText();

            hourText.setText("00");
            minuteText.setText("00");
            secondsText.setText("00");

            startButton.setVisibility(View.VISIBLE);
            resetButton.setVisibility(View.INVISIBLE);
            pauseButton.setVisibility(View.INVISIBLE);

            hourText.setEnabled(true);
            minuteText.setEnabled(true);
            secondsText.setEnabled(true);

            timerSpinner.setEnabled(true);
        } else {
            startButton.setVisibility(View.INVISIBLE);
            resetButton.setVisibility(View.VISIBLE);
            pauseButton.setVisibility(View.VISIBLE);

            startTimer();
        }
    }


}

private void resetTimer() {
    countDownTimer.cancel();
    timerRunning = false;

    timeLeftInMillis = 0;
    hourText.setText("00");
```

```
        minuteText.setText("00");
        secondsText.setText("00");

        hourText.setEnabled(true);
        minuteText.setEnabled(true);
        secondsText.setEnabled(true);

        timerSpinner.setEnabled(true);

        startButton.setVisibility(View.VISIBLE);
        resetButton.setVisibility(View.INVISIBLE);
        pauseButton.setVisibility(View.INVISIBLE);

    }

    private void startTimer() {
        endTime = System.currentTimeMillis() + timeLeftInMillis;

        if(!pref.contains(selectedTask)) {
            prefEdit.putFloat(selectedTask, 0.00f);
            prefEdit.apply();
        }

        timerSpinner.setEnabled(false);

        savedTime = pref.getFloat(selectedTask, 0.00f);
        rHours = 0.00f + savedTime;

        countDownTimer = new CountDownTimer(timeLeftInMillis, 1000) {
            @Override
            public void onTick(long l) {
                timeLeftInMillis = l;
                timeElapsedInMillis = startTimeInMillis - timeLeftInMillis;

                rHours += (1.00/60.00)/60.00;

                prefEdit.putFloat(selectedTask, rHours);
                prefEdit.apply();

                updateCountDownText();
            }

            @Override
            public void onFinish() {

                NotificationCompat.Builder builder = new NotificationCompat.Builder(TimerActivity.this,
CHANNEL_ID)
                        .setSmallIcon(R.drawable.ic_baseline_alarm_24)
                        .setContentTitle("Timer Finished")
                        .setPriority(NotificationCompat.PRIORITY_DEFAULT)
                        ;
```

```
            notificationManager.notify(1, builder.build());

            resetTimer();

          }
        }.start();

        timerRunning = true;
    }

    private void pauseTimer() {
        countDownTimer.cancel();
        timerRunning = false;
    }

    private void updateCountDownText() {
        Integer uHours   = (int) ((timeLeftInMillis / (1000*60*60)) % 24);
        Integer uMinutes = (int) ((timeLeftInMillis / (1000*60)) % 60);
        Integer uSeconds = (int) (timeLeftInMillis / 1000) % 60;

        hourText.setText(String.format("%02d", uHours));
        minuteText.setText(String.format("%02d", uMinutes));
        secondsText.setText(String.format("%02d", uSeconds));
    }

    private long calculateStartTime(int hours, int minutes, int seconds) {
        int hoursInMillis, minsInMillis, secsInMillis, totalMillis;

        hoursInMillis = ((hours * 60) * 60) * 1000;
        minsInMillis = (minutes * 60) * 1000;
        secsInMillis = seconds * 1000;

        totalMillis = hoursInMillis + minsInMillis + secsInMillis;
        return totalMillis;
    }

    private void create_notification_channel() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            CharSequence name = getString(R.string.channel_name);
            String description = getString(R.string.channel_description);
            int importance = NotificationManager.IMPORTANCE_DEFAULT;
            NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name, importance);
            channel.setDescription(description);

            notificationManager = getSystemService(NotificationManager.class);
            notificationManager.createNotificationChannel(channel);

        }
    }
}
```

**Android Manifest File**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.studyapp" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.StudyApp" >
        <activity
            android:name=".activities.MainActivity"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".activities.TimerActivity"
            android:exported="true" />

    </application>

</manifest>
```

## Full Project Drive link:

**https://drive.google.com/drive/folders/1y9S1jTlpQ91jLZYxfzvyCAZWhBZotkug?usp=sharing**

## FUTURE SCOPE

Android app development is the latest technology and research work is still being done in this field. There are far-reaching implications of the current research work. Projects like this kindle the curiosity of individuals to do research work.

The future scope of Android app development seems quite bright. This project has its fair share of future possibilities such as:-

- Most of what we demand our attention is related to present task at hand developing a technique for planning into the future is important. The more skill that is developed in handling these type of action, the more that can be accomplished.

- Create task using the tool you use for your to-do list, whether it is calendar or notetaking app or even a physical person or planner. Use the documentation you produced before and create the task on your to-do list for the future work with as much detail as possible.

- Growing a busines requires investing in future. The daily business processes we all manage are critical, but those seeds that show up unexpectedly in conversations or as you read can represent important future endeavors.

- If you have already collected all the information available to you, it will be easier to start. Otherwise, take the time to put together anything else you need.

## Project Summary

The era of mobile technology has made the life easier. Now, apps are used in the place of machines i.e. apps are replacing instrument. Apps like calculators, notes, music and video players are being continuously used in the place of instrument with same work.

So, this whole project has only one concept, that to record your daily task and to-do lists with a timer set on every task. You can add a task and also edit it. Also, you can delete all your task from the list or simply delete any one of the tasks from the list. You can even provide the task description along with the task title. The whole project has a simple-looking UI design.

## REFERENCES


- www.google.com
- www.wikipedia.org
- www.medium.com
- www.Productivity95.com