**INTRODUCTION**

**ANDROID:**

Androidis a mobile Operating system developed by Google. It is based on a modified version of the Linux kernel and other open source software and it is primarily for touch screen mobile devices such as smart phones and tablets. Initially developed by android INC which Google brought in 2005, Android was unveiled in 2007 with the first commercial android device launched in September 2008. The OS has since gone through multiple major releases with the current version being “Pie” released in August 2018. Android OS Oreo has overtaken as the single most popular version at 20.82% and v.8.0 plus v.8.1 combined are most popular.

**TODO APPS:**

Studies have shown that people perform better when they have written down what they need to do.

What makes the to-do list such an effective productivity tool?

* They give us a structure, a plan that we can stick to
* They are proof of what we have achieved that day, week or month.

So it is better to schedule our tasks and work accordingly and be more productive than before.

**MOTIVATION:**

The era of mobile technology opens the windows to the android app. The websites are vanishing and the mobile phones are emerging. It’s the time to change from conventional websites to apps, which has become the part of our daily routine. We are introducing “ToDo.apk‟ the android application software which would be a “task scheduling app”. It works not only as a normal task scheduler, but also it can work as a digital companion. We have developed a structured, optimal and efficient mobile application, which intelligently shows up the tasks and the free slots, easy to use application which will get minimal user input and deliver highly smart and descriptive scheduled tasks for the user. This will be more of a personal application which stores all the user data in the local phone storage for ensuring user privacy.

Nowadays people find difficulty in managing their to-do list, like we college students have lot of work to do daily, and we face a problem in finding the free slots from our routine in prioritizing the tasks. In order to find the solution for this we designed this application which takes description of the task as input and displays a table view which contains all tasks and free slots.so one can get know when they will be free and they can schedule their task accordingly. There is a flexibility of rescheduling, editing, deleting the tasks being scheduled.

**LITERATURE SURVEY**

This chapter involves the literature survey of the project which gives the brief explanation of the existing system of the project.

There are plethora of application solution built on the category productivity and utilities.

Here are the examples of some of the to-do applications in the market with their features:

**1. Google keep**

Sticky notes are one of the most trusted to-do list managing methods around. Google Keep digitizes this approach into a friendly form for PCs and mobile devices.

You can jot down a reminder onto a single note or create basic lists with items that you can check off. If you don’t want to forget about a particular task, you can tell Keep to shoot you a notification at a time of your choice.

A more specific application targeted at a specific kind of task like :

**2. EveryDay**

Sometimes you don’t want to deal with the fuss. Do you really want to juggle yet another set of account credentials just to stay on top of your shopping list?

If you’d prefer something more basic, check out EveryDay. Creating an account with the app is optional. You only need to make one if you want to back up your tasks online or use the web app.

Without an account, you can still enjoy all the app’s functionality. That includes making lists, creating sub-tasks, setting up reminders, adding notes to tasks, and tinkering with other tools that can help you stay organized.

Yet another specific application but related to task

**3. Inbox by Gmail**

Isn’t inbox by Gmail for mail? Yes, it is. But unlike traditional email clients, it turns each message into a task. When you’re finished with something, you don’t mark it “read,” you check it off as “done.” You can even pin messages you want to act on later so that you don’t forget them.

A funny name but a useful application but for the task management.

**4. Remember the milk**

All the features you expect—such as labels and folder-based hierarchies—are present. But it’s the app’s recent features that have allowed it to compete with services like Todoist.

For example, there’s integration with Gmail, Google Calendar, Twitter, Evernote and more. You can dig into the official IFFFT page to find applets that’ll link remember the Milk to other services.

**PROPOSED SYSTEM**

We have seen some generic and some specific applications which serves the purpose well

But our application is different from all the above applications because we are targeting on the productivity of users more specifically on their ability to do tasks suppose students who are facing problem in doing tasks and they either forget or don’t do the task on time, which is a dent on their productivity ball. Here are some features of our application are

* **Creating new Event: Static Event** (can be periodic) - As a student, I want the app to statically schedule my events/activities so that I know when the events will be occur each week.
* **Creating new Event: Dynamic Event** (Non periodic, can locate a time for itself) - As a student, I want the app to dynamically schedule my events/activities so that the events can change to fit my needs
* **Add finished event to finished List** - As a student, I want the app to be keep a list of finished events so that I can look back on my finished list events to see what I have finished or so that I can reference what I have done before.
* **Show month view of calendar (Main View)** - As a student, I want the app to show a month view so that I can see upcoming events in the month.
* **Allow scroll to future and past month view of calendar (Main View)** - As a student, I want the app to be able to swipe left and right to future and past months so that I am able to plan for upcoming events in future months or look back and reference past events in past months.
* **Opening existing event** - As a student, I want the app to be able to open an existing event so that I can edit the details of the event.
* **Updating existing event** - As a student, I want the app to be able to update an existing event so that I can change the details of the event should they change.
* **Deleting Event** - As a student, I want the app to be able to delete an event so that I can edit my upcoming schedules in case I can no longer do/attend that event or no longer need/want to do/attend that event.

**Requirement Specification**

**Android Studio**:

Android studio is the official IDE (Integrated Development Environment) or tool for developing application exclusively for android platform.

It has a strong editor tool for developing creative UI and emulators for different versions to test and simulate sensors without having actual Android devices.

**SQLite:** SQLite is a self-contained high reliability embedded full-featured public-domain SQL database engine. It is the most used database engine in the world.

**Advanced Java:** It is used for Enterprise applications such as authentication using login and many more web and android applications defined in java are very useful and we can make use of this to develop our application in an efficient way.

**Firebase:** Firebase is a mobile and web app development platform that provides developers with a plethora of tools and services to help them develop high-quality apps, grow their user base, and earn more profit.

**Android Week View library**: it supports day view, week view and the month view infinitely

along with the time.

**DESIGN AND IMPLEMENTATION**

**CODE**

package com.example.lenovo.todolist;

public class MainActivity extends AppCompatActivity {

int id;

public static ListView mListView, checkListView;

public static List<Item> items = new ArrayList<>();

public static List<Item> tmp = new ArrayList<>();

public static ArrayList<Categorie> cat = new ArrayList<>();

TextView nb\_tasks,tvTask;

public static boolean aff\_done, aff\_todo, aff\_passed, aff\_ondate;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mListView = findViewById(R.id.listView);

checkListView = findViewById(R.id.checkCat);

nb\_tasks = findViewById(R.id.nb\_tasks);

tvTask=findViewById( R.id.tvTask );

aff\_done = true;

aff\_todo = true;

aff\_passed = true;

aff\_ondate = true;

id = 0;

CheckBox checkToDo = findViewById(R.id.switch\_todo);

checkToDo.setChecked(true);

tvTask.setOnClickListener( new View.OnClickListener() {

@Override

public void onClick(View v) {

startActivity( new Intent( MainActivity.this,WeekWorkActivity.class ) );

}

} );

checkToDo.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {

@Override

public void onCheckedChanged(CompoundButton compoundButton, boolean b) {

if (b)

aff\_todo = true;

else

aff\_todo = false;

affListCorresponding();

}

});

CheckBox checkDone = (CheckBox) findViewById(R.id.switch\_done);

checkDone.setChecked(true);

checkDone.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {

public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

if (isChecked)

aff\_done = true;

else

aff\_done = false;

affListCorresponding();

}

});

CheckBox checkPassed = (CheckBox) findViewById(R.id.switch\_passed);

checkPassed.setChecked(true);

checkPassed.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {

public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

if (isChecked)

aff\_passed = true;

else

aff\_passed = false;

affListCorresponding();

}

});

CheckBox checkOnDate = (CheckBox) findViewById(R.id.switch\_ondate);

checkOnDate.setChecked(true);

checkOnDate.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {

public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

if (isChecked)

aff\_ondate = true;

else

aff\_ondate = false;

affListCorresponding();

}

});

getData();

getCatData();

if (cat.size() == 0)

cat.add(new Categorie("none", Color.parseColor("#262D3B")));

mListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> adapterView, View view, int position, long id) {

Intent intentMain = new Intent(MainActivity.this, EditItem.class);

Item item = (Item) mListView.getAdapter().getItem(position);

String title = item.getTitle();

String time = item.getTime();

String txt = item.getText();

String date = item.getDate();

String categorie = item.getCategorie();

intentMain.putExtra("position", String.valueOf(position));

intentMain.putExtra("title", title);

intentMain.putExtra("txt", txt);

intentMain.putExtra("date", date);

intentMain.putExtra("time", time);

intentMain.putExtra("categorie", categorie);

startActivityForResult(intentMain, 1);

}

});

ItemAdapter adapter = new ItemAdapter(MainActivity.this, items);

checkAdapter adapter1 = new checkAdapter(MainActivity.this, cat);

checkListView.setAdapter(adapter1);

mListView.setAdapter(adapter);

checkDate();

}

public void getData() {

List<Item> list = new ArrayList<>();

Item tmp;

SQLiteDatabase mydatabase = openOrCreateDatabase("todolist", MODE\_PRIVATE, null);

mydatabase.execSQL("CREATE TABLE IF NOT EXISTS tasks(Titre VARCHAR, Date VARCHAR, Status VARCHAR, Txt VARCHAR, Cat VARCHAR);");

Cursor resultSet = mydatabase.rawQuery("Select \* from tasks", null);

resultSet.moveToFirst();

int count = 0;

while (count < resultSet.getCount())

{

String title = resultSet.getString(resultSet.getColumnIndex("Titre"));

String date = resultSet.getString(resultSet.getColumnIndex("Date"));

String status = resultSet.getString(resultSet.getColumnIndex("Status"));

String txt = resultSet.getString(resultSet.getColumnIndex("Txt"));

String cat = resultSet.getString(resultSet.getColumnIndex("Cat"));

Date d = new Date();

// SimpleDateFormat newDateFormat = new SimpleDateFormat("EE d MMM yyyyHH:mm","15");

SimpleDateFormat newDateFormat =new SimpleDateFormat( "EEE, MMM d, ''yy" );

try {

d = newDateFormat.parse(date);

} catch (ParseException e) {

e.printStackTrace();

}

tmp = new Item(title, txt, d);

if (status.equals(Item.Status.DONE.toString()))

tmp.setStatus(Item.Status.DONE);

else

tmp.setStatus(Item.Status.TODO);

tmp.setCategorie(cat);

list.add(tmp);

count++;

resultSet.moveToNext();

}

items = list;

}

public void getCatData() {

ArrayList<Categorie> list = new ArrayList<>();

Categorie tmp;

SQLiteDatabase mydatabase = openOrCreateDatabase("todolist", MODE\_PRIVATE, null);

mydatabase.execSQL("CREATE TABLE IF NOT EXISTS cats(Name VARCHAR, Color VARCHAR);");

Cursor resultSet = mydatabase.rawQuery("Select \* from cats", null);

resultSet.moveToFirst();

int count = 0;

while (count < resultSet.getCount()) {

String name = resultSet.getString(resultSet.getColumnIndex("Name"));

String color = resultSet.getString(resultSet.getColumnIndex("Color"));

tmp = new Categorie(name, Integer.parseInt(color));

list.add(tmp);

count++;

resultSet.moveToNext();

}

cat = list;

}

public String addToDataBase(int i) {

Item tmp = items.get(i);

String query = "'";

query += tmp.getTitle() + "','";

query += tmp.getDate() + tmp.getTime() + "','";

query += tmp.getStatus().toString() + "','";

query += tmp.getText() + "','";

query += tmp.getCategorie() + "'";

return query;}

public void saveData() {

String query;

SQLiteDatabase mydatabase = openOrCreateDatabase("todolist", MODE\_PRIVATE, null);

mydatabase.execSQL("DROP TABLE IF EXISTS tasks");

mydatabase.execSQL("CREATE TABLE IF NOT EXISTS tasks(Titre VARCHAR, Date VARCHAR, Status VARCHAR, Txt VARCHAR, Cat VARCHAR);");

for (int i = 0; i < items.size(); i++) {

query = addToDataBase(i);

mydatabase.execSQL("INSERT INTO tasks VALUES(" + query + ");");

}

}

public void saveCategory() {

String query;

SQLiteDatabase mydatabase = openOrCreateDatabase("todolist", MODE\_PRIVATE, null);

mydatabase.execSQL("DROP TABLE IF EXISTS cats");

mydatabase.execSQL("CREATE TABLE IF NOT EXISTS cats(Name VARCHAR, Color VARCHAR);");

for (int i = 0; i < cat.size(); i++) {

query = "'" + cat.get(i).getName() + "','" + String.valueOf(cat.get(i).getColor()) + "'";

mydatabase.execSQL("INSERT INTO cats VALUES(" + query + ");");

}

}

public void settings(View V) {

DrawerLayout mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer\_layout);

mDrawerLayout.openDrawer(Gravity.LEFT);

}

public void add(View v) {

Intent intentMain = new Intent(MainActivity.this, AddItem.class);

startActivityForResult(intentMain, 1);

}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) { if (requestCode == 1) {

if (resultCode == Activity.RESULT\_OK) {

String title = data.getStringExtra("title");

String txt = data.getStringExtra("txt");

String date = data.getStringExtra("date");

String delete = data.getStringExtra("delete");

String category = data.getStringExtra("categorie");

SimpleDateFormat newDateFormat = new SimpleDateFormat("EE d MMM yyyy k:m");

Date d = null;

try {

d = newDateFormat.parse(date);

} catch (ParseException e) {

e.printStackTrace();

}

if (data.getStringExtra("edit").equals("true")) {

int position = Integer.parseInt(data.getStringExtra("position"));

try {

modifyItem(position, title, txt, d, delete, category);

} catch (ParseException e) {

e.printStackTrace();

}

} else {

Item newItem = new Item(title, txt, d);

newItem.setCategorie(category);

try {

addToList(newItem);

} catch (ParseException e) {

e.printStackTrace();

}

}

}

if (resultCode == Activity.RESULT\_CANCELED) {

//here goes nothing

}

}

if (requestCode == 2) {

if (resultCode == Activity.RESULT\_OK) {

saveCategory();

checkCategories();

affListCorresponding();

((checkAdapter) checkListView.getAdapter()).notifyDataSetChanged();

}

if (resultCode == Activity.RESULT\_CANCELED) {

saveCategory();

checkCategories();

affListCorresponding();

((checkAdapter) checkListView.getAdapter()).notifyDataSetChanged();

}}}

public void checkDate() {

int i = 0;

Date d;

d = new Date();

nb\_tasks.setText(String.valueOf(items.size()) + " Tasks");

while (i < items.size()) {

if (!(items.get(i).getRealDate().after(d))) {

items.get(i).setPassed(true);

items.get(i).setDateColor("#FF0000");

} else {

items.get(i).setPassed(false);

items.get(i).setDateColor("#121212");

} i++;

}}

public void addToList(Item item) throws ParseException {

items.add(item);

checkDate();

saveData();

Date f = new Date();

int c = 0;

int color = Color.BLUE;

while (c < cat.size()) {

if (item.getCategorie().equals(cat.get(c).getName())) {

color = cat.get(c).getColor();

}

c++;

}

int delay = (int) (item.getRealDate().getTime() - f.getTime());

if (delay > 0)

scheduleNotification(getNotification(item.getTitle(), item.getText(), color), delay);

affListCorresponding();

}

public void modifyItem(int position, String title, String txt, Date d, String delete, String cate) throws ParseException {

Item item = items.get(position);

if (delete.equals("false")) {

item.setTitle(title);

item.setText(txt);

item.setDueDate(d);

item.setCategorie(cate);

} else

items.remove(item);

checkDate();

saveData();

Date f = new Date();

int delay = (int) (d.getTime() - f.getTime());

int color = Color.BLUE;

int c = 0;

while (c < cat.size()) {

if (item.getCategorie().equals(cat.get(c).getName())) {

color = cat.get(c).getColor();

}

c++;

}

if (delay > 0)

scheduleNotification(getNotification(title, txt, color), delay);

affListCorresponding();

}

public boolean showCatForItem(Item item) {

int i = 0;

while (i < cat.size()) {

if (cat.get(i).getName().equals(item.getCategorie())) {

return cat.get(i).getShow();

}

i++;

}

return false;

}

public void affListCorresponding() {

int nb\_items = items.size();

boolean t, p;

int i = 0;

tmp.clear();

while (i < nb\_items) {

t = false;

p = false;

if (aff\_done && items.get(i).getStatus() == Item.Status.DONE)

t = true;

if (aff\_todo && items.get(i).getStatus() == Item.Status.TODO)

t = true;

if ((aff\_passed && items.get(i).getPassed()))

p = true;

if ((aff\_ondate && !items.get(i).getPassed()))

p = true;

if (t && p && showCatForItem(items.get(i)))

tmp.add(items.get(i));

i++;

}

ItemAdapter adapter = new ItemAdapter(MainActivity.this, tmp);

mListView.setAdapter(adapter);

if (tmp.size() > 1)

((TextView) findViewById(R.id.nb\_tasks)).setText(String.valueOf(tmp.size()) + " Tasks");

else

((TextView) findViewById(R.id.nb\_tasks)).setText(String.valueOf(tmp.size()) + " Task");

adapter.notifyDataSetChanged();

}

public void todoClick(View v) {

final int position = mListView.getPositionForView((View) v.getParent());

SwipeLayout s = (SwipeLayout) mListView.getChildAt(position);

Item a = items.get(position);

a.setStatus(Item.Status.TODO);

affListCorresponding();

saveData();

s.close(true);

}

public void catCheck(View v) {

final int position = checkListView.getPositionForView((View) v.getParent());

CheckBox checkBox = (CheckBox) v;

if (checkBox.isChecked())

cat.get(position).setShow(true);

else

cat.get(position).setShow(false);

affListCorresponding();

}

public void doneClick(View v) {

final int position = mListView.getPositionForView((View) v.getParent());

SwipeLayout s = (SwipeLayout) mListView.getChildAt(position);

Item a = items.get(position);

a.setStatus(Item.Status.DONE);

s.close(true);

ItemAdapter b = (ItemAdapter) mListView.getAdapter();

affListCorresponding();

b.notifyDataSetChanged();

saveData();

}

private void scheduleNotification(Notification notification, int delay) {

Intent notificationIntent = new Intent(this, NotificationPublisher.class);

notificationIntent.putExtra(NotificationPublisher.NOTIFICATION\_ID, 1);

notificationIntent.putExtra(NotificationPublisher.NOTIFICATION, notification);

PendingIntent pendingIntent = PendingIntent.getBroadcast(this, 0, notificationIntent, PendingIntent.FLAG\_UPDATE\_CURRENT);

long futureInMillis = SystemClock.elapsedRealtime() + delay;

AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM\_SERVICE);

alarmManager.set(AlarmManager.ELAPSED\_REALTIME\_WAKEUP, futureInMillis, pendingIntent);

}

@RequiresApi(api = Build.VERSION\_CODES.JELLY\_BEAN)

private Notification getNotification(String Title, String content, int color) {

Notification.Builder builder = new Notification.Builder(this);

builder.setContentTitle(Title);

builder.setContentText(content);

builder.setSmallIcon(R.drawable.ic\_notification);

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.LOLLIPOP) {

builder.setColor(color);

}

affListCorresponding();

return builder.build();

}

public static ArrayList<Categorie> getCatA() {

ArrayList<Categorie> tmp = new ArrayList<Categorie>();

int i = 0;

while (i < cat.size())

tmp.add(cat.get(i++));

return tmp;

}

public void closeMenu(View v) {

DrawerLayout d = ((DrawerLayout) findViewById(R.id.drawer\_layout));

d.closeDrawers();

}

public static ArrayList<Categorie> getCat() {

return (getCatA());

}

public void checkCategories() {

int i = 0;

while (i < items.size()) {

int c = 0;

boolean found = false;

while (c < cat.size()) {

if (items.get(i).getCategorie().equals(cat.get(c).getName()))

found = true;

c++;

}

if (!found)

items.get(i).setCategorie("none");

i++;

}

affListCorresponding();}

public void addCategorie(View v) {

Intent intentMain = new Intent(MainActivity.this, addCategory.class);

startActivityForResult(intentMain, 2);

checkCategories();}}

**activity\_main.xml**

<?xml version="1.0" encoding="utf-8"?>

<android.support.v4.widget.DrawerLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/drawer\_layout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="@drawable/bgpl"

tools:context=".MainActivity">

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior">

<RelativeLayout

android:id="@+id/top"

android:layout\_width="match\_parent"

android:layout\_height="50dp"

android:background="@color/colorPrimary">

<TextView

android:id="@+id/nb\_tasks"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:gravity="center"

android:text="nb Tasks"

android:textAlignment="center"

android:textColor="#FFFFFF"

android:textSize="18dp"

android:textStyle="bold" />

<ImageView

android:layout\_width="40dp"

android:layout\_height="40dp"

android:layout\_centerVertical="true"

android:contentDescription="menu\_list"

android:onClick="settings"

android:paddingLeft="10dp"

android:src="@drawable/menu" />

</RelativeLayout>

<RelativeLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_below="@+id/top">

<ListView

android:id="@+id/listView"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_alignParentStart="true"

android:layout\_alignParentTop="true" />

<ImageView

android:id="@+id/add\_btn"

android:layout\_width="80dp"

android:layout\_height="80dp"

android:layout\_alignParentBottom="true"

android:layout\_alignParentEnd="true"

android:layout\_alignParentRight="true"

android:layout\_margin="10dp"

android:onClick="add"

android:src="@drawable/plus" />

<TextView

android:id="@+id/tvTask"

android:layout\_width="234dp"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:layout\_alignParentStart="true"

android:layout\_marginBottom="23dp"

android:text="Enter the fixed tasks"

android:textAlignment="center"

android:textColor="#f4ebec"

android:textSize="20dp"

android:textStyle="bold"

android:typeface="serif" />

</RelativeLayout>

</RelativeLayout>

<LinearLayout

android:id="@+id/left\_drawer"

android:layout\_width="200dp"

android:layout\_height="fill\_parent"

android:layout\_gravity="start|left"

android:background="#FFFFFF"

android:fillViewport="true"

android:orientation="vertical">

<RelativeLayout

android:id="@+id/top2"

android:layout\_width="match\_parent"

android:layout\_height="50dp"

android:background="@color/colorPrimary">

<TextView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:gravity="center"

android:text="Settings"

android:textAlignment="center"

android:textColor="#FFFFFF"

android:textSize="18dp"

android:textStyle="bold" />

<ImageView

android:layout\_width="40dp"

android:layout\_height="40dp"

android:layout\_alignParentEnd="true"

android:layout\_centerVertical="true"

android:onClick="closeMenu"

android:padding="10dp"

android:src="@drawable/close" />

</RelativeLayout>

<LinearLayout

android:background="#EFEFEF"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingLeft="2dp">

<ImageView

android:id="@+id/doneT"

android:layout\_width="20dp"

android:layout\_height="20dp"

android:layout\_gravity="center\_vertical"

android:src="@drawable/donetask" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:clickable="false"

android:padding="10dp"

android:text="Status"

android:textSize="20sp" />

</LinearLayout>

<CheckBox

android:id="@+id/switch\_done"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Done" />

<CheckBox

android:id="@+id/switch\_todo"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="To do" />

<LinearLayout

android:background="#EFEFEF"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingLeft="2dp">

<ImageView

android:layout\_width="20dp"

android:layout\_height="20dp"

android:layout\_gravity="center\_vertical"

android:src="@drawable/ic\_date\_range\_black\_24dp" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:clickable="false"

android:padding="10dp"

android:text="Date"

android:textSize="20sp" />

</LinearLayout>

<CheckBox

android:id="@+id/switch\_passed"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Passed" />

<CheckBox

android:id="@+id/switch\_ondate"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Up to Date" />

<LinearLayout

android:background="#EFEFEF"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingLeft="2dp">

<ImageView

android:layout\_width="20dp"

android:layout\_height="20dp"

android:layout\_gravity="center\_vertical"

android:src="@drawable/label" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:onClick="addCategorie"

android:padding="10dp"

android:text="Categories"

android:textSize="20sp" />

<ImageView

android:layout\_width="20dp"

android:layout\_height="20dp"

android:layout\_gravity="center\_vertical"

android:onClick="addCategorie"

android:src="@drawable/config" />

</LinearLayout>

<ListView

android:id="@+id/checkCat"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:fillViewport="true">

</ListView>

<View

android:layout\_width="match\_parent"

android:layout\_height="1dp"

android:background="#DFDFDF" />

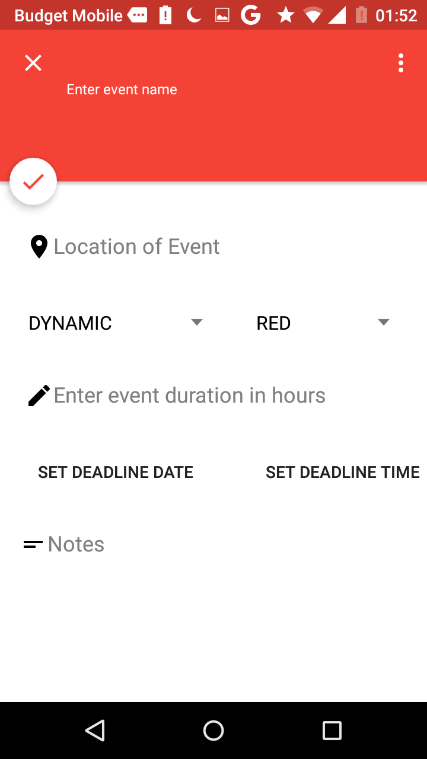
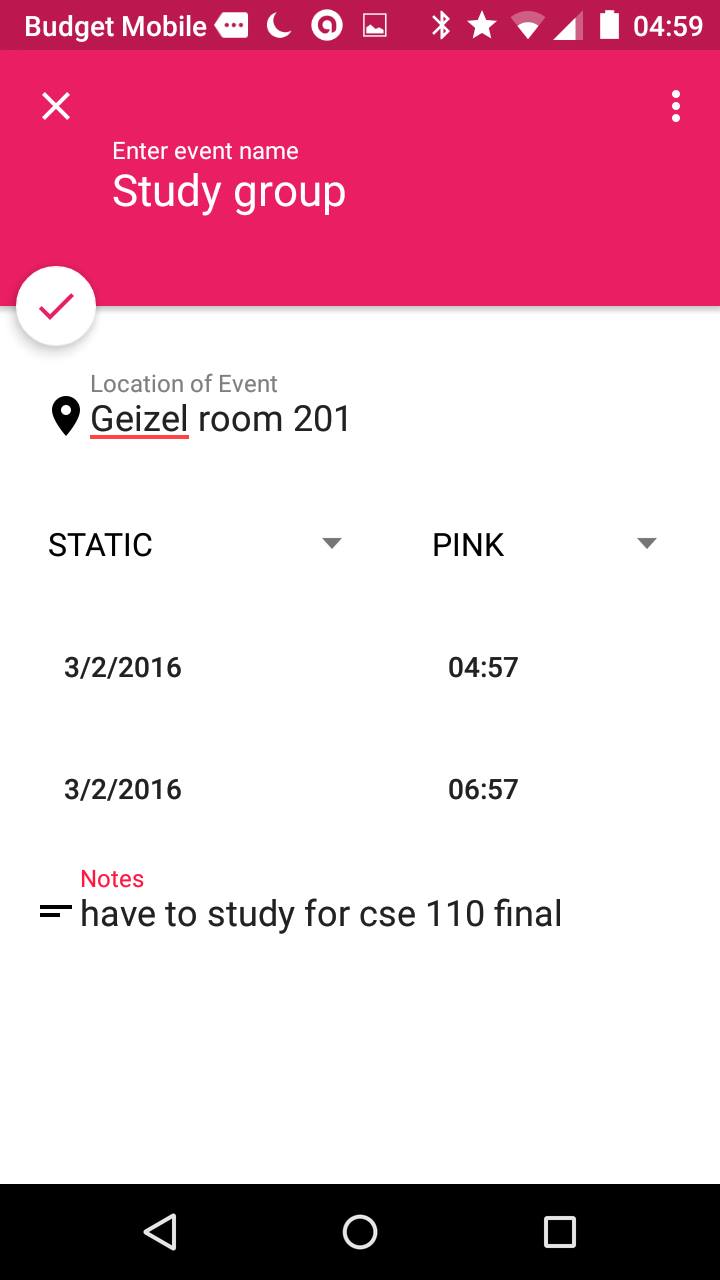
</LinearLayout>

</android.support.v4.widget.DrawerLayout>

**RESULTS**

**** 

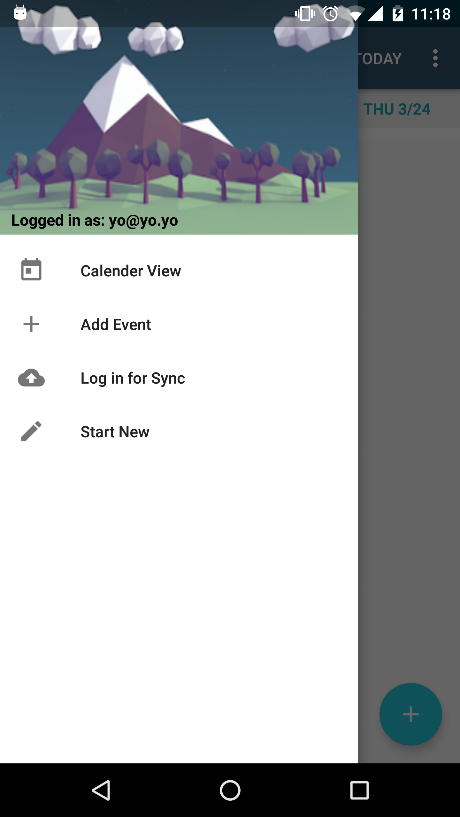
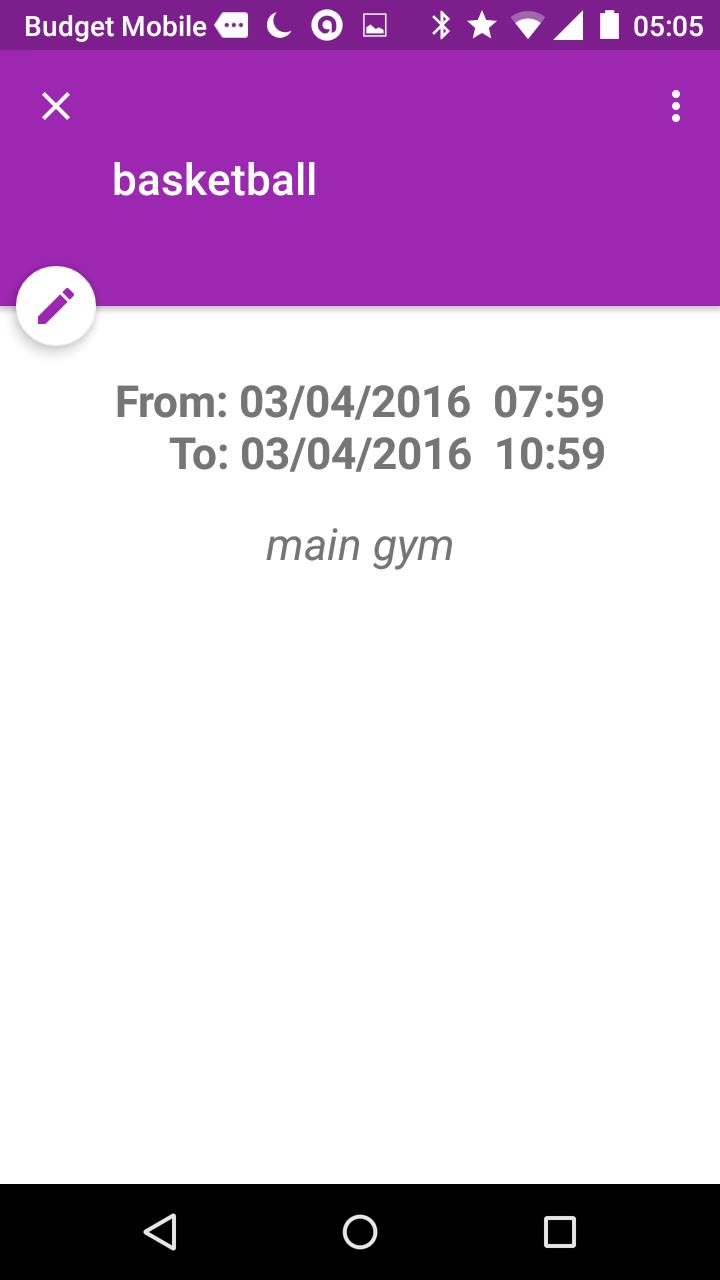
Our task scheduler application opens with a splash screen activity. Followed by that the right side screen will appear. When we click on the plus mark then the new activity starts and there we can schedule the tasks. When we click on menu button we can select among day view, week view and month view.

** **

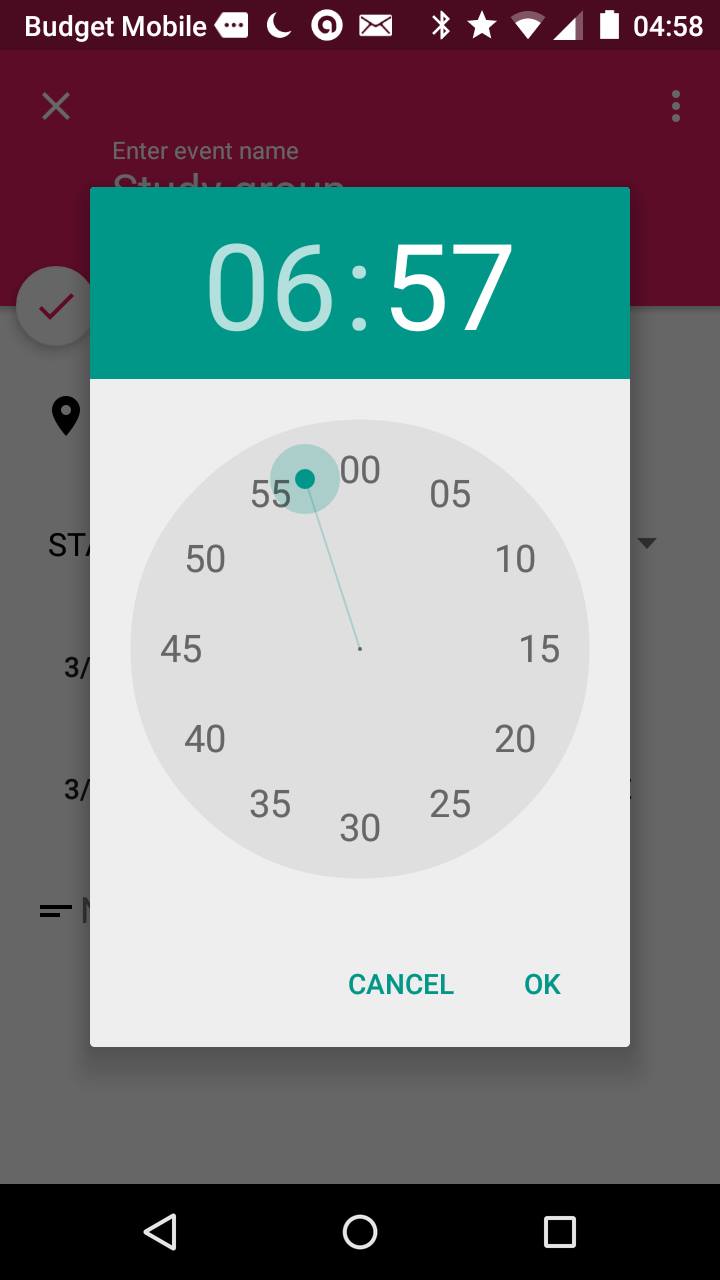
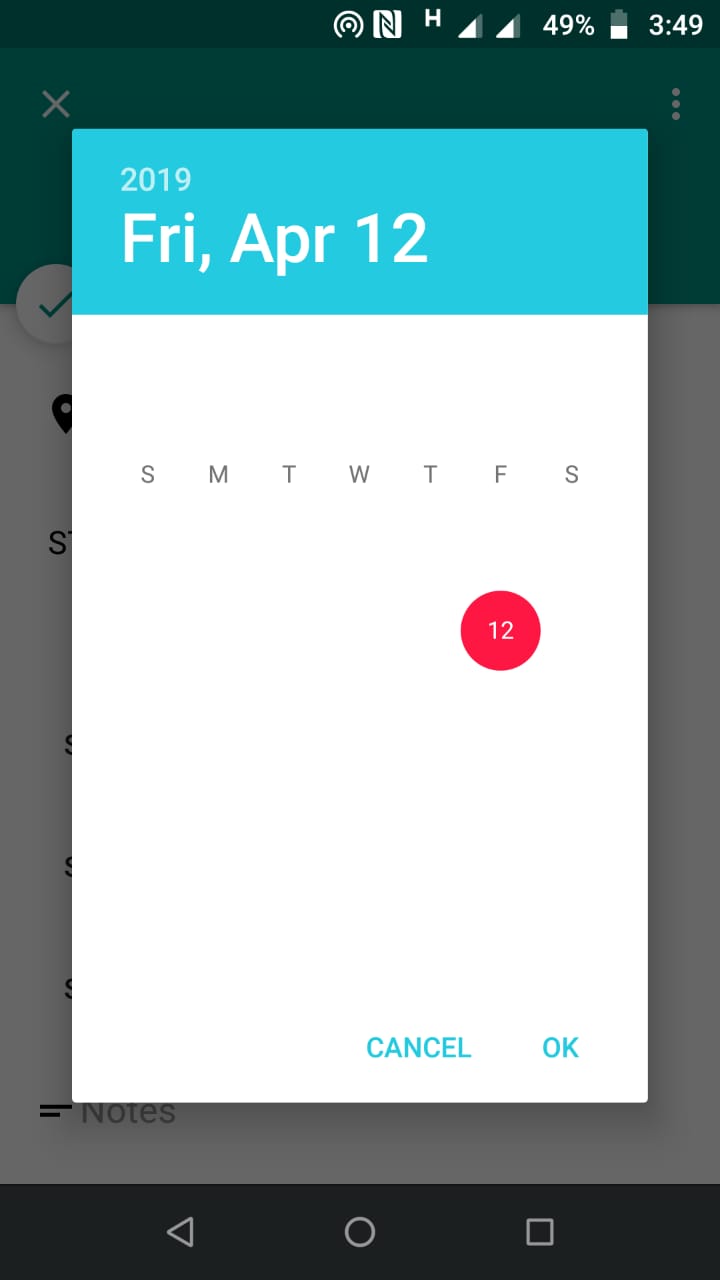
Here we can enter the tasks and its category. i.e. whether the task is static or dynamic,

The name of the task, location of the task, the colour with which the task should appear.

We can also add the notes regarding the task. If it is a static task then start date and time and end date and time should be added. When we click on right mark the event will be added and displayed on the table with the chosen color.

** **

When we click on navigation drawer we get the image what we have at the left side. When we click on the scheduled event it will display the name of the event and the description of the event. we can also delete this task from here.

** **

**Time picker dialog Date picker dialog**

This is the UI for choosing time and date for the event.

**CONCLUSION**

Task scheduler or to-do apps are very helpful. They are the proof of what we have achieving in our life .they gives the clarity of whether we are really making use of our time or we just wasting it. Keeping that in mind we developed this application which takes our tasks as input and displays on table which contains all tasks highlighted with the chosen color and also their free slots.so one can get know when they will be free and they can schedule their task accordingly. There is also flexibility of rescheduling, editing, deleting the tasks being scheduled.

**REFERENCES**

>Fundamentals of Database Systems by “Ramez Elmasri”

>GitHub

>Developer.Android.com

>Quora

> Stackoverflow