**TO DO LIST:**

**ADD TASK ACTIVITY.JAVA:**

package com.skapps.android.todolist;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.RadioGroup;  
import android.widget.Toast;  
import com.skapps.android.todolist.database.AppDatabase;  
import com.skapps.android.todolist.database.TaskEntry;  
import java.util.Date;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.lifecycle.Observer;  
import androidx.lifecycle.ViewModelProvider;  
  
  
public class AddTaskActivity extends AppCompatActivity {  
  
 // Extra for the task ID to be received in the intent  
 public static final String *EXTRA\_TASK\_ID* = "extraTaskId";  
 // Extra for the task ID to be received after rotation  
 public static final String *INSTANCE\_TASK\_ID* = "instanceTaskId";  
 // Constants for priority  
 public static final int *PRIORITY\_HIGH* = 1;  
 public static final int *PRIORITY\_MEDIUM* = 2;  
 public static final int *PRIORITY\_LOW* = 3;  
 // Constant for default task id to be used when not in update mode  
 private static final int *DEFAULT\_TASK\_ID* = -1;  
 // Constant for logging  
 private static final String *TAG* = AddTaskActivity.class.getSimpleName();  
 // Fields for views  
 EditText mEditText;  
 RadioGroup mRadioGroup;  
 Button mButton;  
  
 private int mTaskId = *DEFAULT\_TASK\_ID*;  
  
 private AppDatabase mDb;  
  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_add\_task*);  
  
 initViews();  
  
 mDb = AppDatabase.*getInstance*(getApplicationContext());  
  
 if (savedInstanceState != null && savedInstanceState.containsKey(*INSTANCE\_TASK\_ID*)) {  
 mTaskId = savedInstanceState.getInt(*INSTANCE\_TASK\_ID*, *DEFAULT\_TASK\_ID*);  
 }  
  
 Intent intent = getIntent();  
 if (intent != null && intent.hasExtra(*EXTRA\_TASK\_ID*)) {  
 mButton.setText(R.string.*update\_button*);  
 setTitle("Update Task");  
 if (mTaskId == *DEFAULT\_TASK\_ID*) {  
 mTaskId = intent.getIntExtra(*EXTRA\_TASK\_ID*, *DEFAULT\_TASK\_ID*);  
  
 AddTaskViewModelFactory factory = new AddTaskViewModelFactory(mDb, mTaskId);  
 final AddTaskViewModel viewModel = new ViewModelProvider(this, factory).get(AddTaskViewModel.class);  
 viewModel.getTask().observe(this, new Observer<TaskEntry>() {  
 @Override  
 public void onChanged(TaskEntry taskEntry) {  
 viewModel.getTask().removeObserver(this);  
 populateUI(taskEntry);  
 }  
 });  
 }  
 }  
 }  
  
 @Override  
 protected void onSaveInstanceState(Bundle outState) {  
 outState.putInt(*INSTANCE\_TASK\_ID*, mTaskId);  
 super.onSaveInstanceState(outState);  
 }  
  
  
 private void initViews() {  
 mEditText = findViewById(R.id.*editTextTaskDescription*);  
 mRadioGroup = findViewById(R.id.*radioGroup*);  
  
 mButton = findViewById(R.id.*saveButton*);  
 mButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 onSaveButtonClicked();  
 }  
 });  
 }  
  
  
 private void populateUI(TaskEntry task) {  
 if (task == null ){  
 return;  
 }  
 mEditText.setText(task.getDescription());  
 setPriorityInViews(task.getPriority());  
 }  
  
  
 public void onSaveButtonClicked() {  
 String description = mEditText.getText().toString();  
 int priority = getPriorityFromViews();  
 Date date = new Date();  
  
 if(description.equals("")){  
 Toast.*makeText*(this,"Describe your task", Toast.*LENGTH\_SHORT*).show();  
 }else{  
 final TaskEntry taskEntry = new TaskEntry(description, priority, date, false );  
  
 AppExecutors.*getInstance*().diskIO().execute(new Runnable() {  
 @Override  
 public void run() {  
 if(mTaskId == *DEFAULT\_TASK\_ID*){  
 mDb.taskDao().insertTask(taskEntry);  
 }else {  
 taskEntry.setId(mTaskId);  
 mDb.taskDao().updateTask(taskEntry);  
 }  
 finish(); //automatically return to main activity  
 }  
 });  
 }  
  
  
 }  
  
  
 public int getPriorityFromViews() {  
 int priority = 1;  
 int checkedId = ((RadioGroup) findViewById(R.id.*radioGroup*)).getCheckedRadioButtonId();  
 switch (checkedId) {  
 case R.id.*radButton1*:  
 priority = *PRIORITY\_HIGH*;  
 break;  
 case R.id.*radButton2*:  
 priority = *PRIORITY\_MEDIUM*;  
 break;  
 case R.id.*radButton3*:  
 priority = *PRIORITY\_LOW*;  
 }  
 return priority;  
 }  
  
  
 public void setPriorityInViews(int priority) {  
 switch (priority) {  
 case *PRIORITY\_HIGH*:  
 ((RadioGroup) findViewById(R.id.*radioGroup*)).check(R.id.*radButton1*);  
 break;  
 case *PRIORITY\_MEDIUM*:  
 ((RadioGroup) findViewById(R.id.*radioGroup*)).check(R.id.*radButton2*);  
 break;  
 case *PRIORITY\_LOW*:  
 ((RadioGroup) findViewById(R.id.*radioGroup*)).check(R.id.*radButton3*);  
 }  
 }  
}

**ADD TASK VIEW MODEL.JAVA:**

package com.skapps.android.todolist;  
  
import com.skapps.android.todolist.database.AppDatabase;  
import com.skapps.android.todolist.database.TaskEntry;  
import androidx.lifecycle.LiveData;  
import androidx.lifecycle.ViewModel;  
  
class AddTaskViewModel extends ViewModel {  
  
 private LiveData<TaskEntry> task;  
  
  
 public AddTaskViewModel(AppDatabase database, int taskId) {  
 task = database.taskDao().loadTaskById(taskId);  
 }  
  
 public LiveData<TaskEntry> getTask() {  
 return task;  
 }  
}

**ADD TAS VIEW MODELFACTORY.JAVA:**

package com.skapps.android.todolist;  
  
import com.skapps.android.todolist.database.AppDatabase;  
  
import androidx.annotation.NonNull;  
import androidx.lifecycle.ViewModel;  
import androidx.lifecycle.ViewModelProvider;  
  
public class AddTaskViewModelFactory extends ViewModelProvider.NewInstanceFactory {  
  
 private final AppDatabase mDb;  
 private final int mTaskId;  
  
 public AddTaskViewModelFactory(AppDatabase database, int mTaskId){  
 mDb = database;  
 this.mTaskId = mTaskId;  
 }  
  
 @NonNull  
 @Override  
 public <T extends ViewModel> T create(@NonNull Class<T> modelClass) {  
 return (T) new AddTaskViewModel(mDb, mTaskId);  
 }  
}

**APPEXECUTORS.JAVA:**

package com.skapps.android.todolist;  
  
  
import android.os.Handler;  
import android.os.Looper;  
import java.util.concurrent.Executor;  
import java.util.concurrent.Executors;  
  
import androidx.annotation.NonNull;  
  
//Executors helps in sequencetial execution, otherwise we use Thread runnable which may cause race condition  
public class AppExecutors {  
  
 // For Singleton instantiation  
 private static final Object *LOCK* = new Object();  
 private static AppExecutors *sInstance*;  
 private final Executor diskIO;  
 private final Executor mainThread;  
 private final Executor networkIO;  
  
 private AppExecutors(Executor diskIO, Executor networkIO, Executor mainThread) {  
 this.diskIO = diskIO;  
 this.networkIO = networkIO;  
 this.mainThread = mainThread;  
 }  
  
 //For singleton instantiation  
 public static AppExecutors getInstance() {  
 if (*sInstance* == null) {  
 synchronized (*LOCK*) {  
 *sInstance* = new AppExecutors(Executors.*newSingleThreadExecutor*(),  
 Executors.*newFixedThreadPool*(10),  
 new MainThreadExecutor());  
 }  
 }  
 return *sInstance*;  
 }  
  
 public Executor diskIO() {  
 return diskIO;  
 }  
  
 public Executor mainThread() {  
 return mainThread;  
 }  
  
 public Executor networkIO() {  
 return networkIO;  
 }  
  
 private static class MainThreadExecutor implements Executor {  
 private Handler mainThreadHandler = new Handler(Looper.*getMainLooper*());  
  
 @Override  
 public void execute(@NonNull Runnable command) {  
 mainThreadHandler.post(command);  
 }  
 }  
}

**MAINACTIVITY.JAVA:**

package com.skapps.android.todolist;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ProgressBar;  
import android.widget.TextView;  
import com.google.android.material.snackbar.Snackbar;  
import com.skapps.android.todolist.database.AppDatabase;  
import com.skapps.android.todolist.database.TaskEntry;  
import com.google.android.material.floatingactionbutton.FloatingActionButton;  
import java.util.Collections;  
import java.util.Comparator;  
import java.util.List;  
import java.util.stream.Collectors;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.AlertDialog;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.constraintlayout.widget.ConstraintLayout;  
import androidx.lifecycle.Observer;  
import androidx.lifecycle.ViewModelProvider;  
import androidx.recyclerview.widget.DividerItemDecoration;  
import androidx.recyclerview.widget.ItemTouchHelper;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import static androidx.recyclerview.widget.DividerItemDecoration.*VERTICAL*;  
  
  
public class MainActivity extends AppCompatActivity implements TaskAdapter.ItemClickListener, TaskAdapter.CheckBoxCheckListener{  
  
 private RecyclerView mRecyclerView;  
 private TaskAdapter mAdapter;  
 private ProgressBar mprogressBar;  
 private TextView mProgressValue;  
 private TextView mEmptyView;  
 private ConstraintLayout mConstraintLayout;  
  
  
 private double mTotalProgressPercent;  
 private AppDatabase mDb;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
  
 mprogressBar = findViewById(R.id.progressBar);  
 mRecyclerView = findViewById(R.id.recyclerViewTasks);  
 mProgressValue = findViewById(R.id.progressValue);  
 mEmptyView = findViewById(R.id.emptyView);  
 mConstraintLayout = findViewById(R.id.constraintLayout);  
  
  
 mRecyclerView.setLayoutManager(new LinearLayoutManager(this));  
  
 mAdapter = new TaskAdapter(this, this, this);  
 mRecyclerView.setAdapter(mAdapter);  
  
 DividerItemDecoration decoration = new DividerItemDecoration(getApplicationContext(), VERTICAL);  
 mRecyclerView.addItemDecoration(decoration);  
  
 mDb = AppDatabase.getInstance(getApplicationContext());  
  
 ViewGroup.MarginLayoutParams params = (ViewGroup.MarginLayoutParams) mConstraintLayout.getLayoutParams();  
 params.bottomMargin = 0;  
  
  
 new ItemTouchHelper(new ItemTouchHelper.SimpleCallback(ItemTouchHelper.UP | ItemTouchHelper.DOWN, ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT) {  
  
 boolean drag = false;  
  
 @Override  
 public boolean onMove(@NonNull RecyclerView recyclerView, @NonNull RecyclerView.ViewHolder viewHolder, @NonNull RecyclerView.ViewHolder target) {  
 int position\_dragged = viewHolder.getAdapterPosition();  
 int position\_target = target.getAdapterPosition();  
  
 Collections.swap(mAdapter.getTasks(), position\_dragged, position\_target);  
 mAdapter.notifyItemMoved(position\_dragged, position\_target);  
  
 return false;  
 }  
  
 @Override  
 public void onSelectedChanged(@Nullable RecyclerView.ViewHolder viewHolder, int actionState) {  
 super.onSelectedChanged(viewHolder, actionState);  
  
 if(actionState == ItemTouchHelper.ACTION\_STATE\_DRAG) {  
 drag = true;  
 }  
  
 if(actionState == ItemTouchHelper.ACTION\_STATE\_IDLE && drag) {  
 Log.d("DragTest","DRAGGGING stop");  
 drag= false;  
  
 final List<TaskEntry> NewTasks = mAdapter.getTasks();  
 AppExecutors.getInstance().diskIO().execute(new Runnable() {  
 @Override  
 public void run() {  
 try {  
 mDb.taskDao().deleteAll();  
  
 for(int i =0; i < NewTasks.size(); i++){  
 TaskEntry task = NewTasks.get(i);  
 mDb.taskDao().insertTask(new TaskEntry(  
 task.getDescription(),  
 task.getPriority(),  
 task.getUpdatedAt(),  
 task.isChecked()  
 ));  
 }  
 }catch (Exception ignored){}  
  
 }  
 });  
 }  
 }  
  
 @Override  
 public void onSwiped(@NonNull final RecyclerView.ViewHolder viewHolder, int swipeDir) {  
 int position = viewHolder.getAdapterPosition();  
 List<TaskEntry> tasks = mAdapter.getTasks();  
 final TaskEntry taskToBeDeleted = tasks.get(position);  
  
 AppExecutors.getInstance().diskIO().execute(new Runnable() {  
 @Override  
 public void run() {  
 try {  
 mDb.taskDao().deleteTask(taskToBeDeleted);  
 }catch (Exception ignored){}  
 }  
 });  
  
 Snackbar snackbar = Snackbar  
 .make(viewHolder.itemView, "Task deleted!", Snackbar.LENGTH\_LONG);  
 snackbar.setAction("UNDO", new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 AppExecutors.getInstance().diskIO().execute(new Runnable() {  
 @Override  
 public void run() {  
 try {  
 mDb.taskDao().insertTask(taskToBeDeleted);  
 }catch (Exception ignored){}  
 }  
 });  
  
 }  
 }).show();  
 }  
 }).attachToRecyclerView(mRecyclerView);  
  
  
 FloatingActionButton fabButton = findViewById(R.id.fab);  
  
 fabButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
  
 Intent addTaskIntent = new Intent(MainActivity.this, AddTaskActivity.class);  
 startActivity(addTaskIntent);  
 }  
 });  
 setupViewModel();  
 }  
  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.menu, menu);  
 return true;  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 switch (item.getItemId()){  
 case R.id.delete\_all\_tasks:  
 new AlertDialog.Builder(this)  
 .setCancelable(false)  
 .setTitle("Delete all tasks")  
 .setPositiveButton("No", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 dialog.cancel();  
 }  
 })  
 .setNegativeButton("Yes", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 AppExecutors.getInstance().diskIO().execute(new Runnable() {  
 @Override  
 public void run() {  
 mDb.taskDao().deleteAll();  
 }  
 });  
 }  
 })  
 .setMessage( "Do you want to delete all tasks?")  
 .show();  
 break;  
 case R.id.uncheck\_all:  
 new AlertDialog.Builder(this)  
 .setCancelable(false)  
 .setTitle("Reset task list")  
 .setPositiveButton("No", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 dialog.cancel();  
 }  
 })  
 .setNegativeButton("Yes", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
  
 AppExecutors.getInstance().diskIO().execute(new Runnable() {  
 @Override  
 public void run() {  
 for(TaskEntry taskEntry: mAdapter.getTasks()){  
 taskEntry.setChecked(false);  
 mDb.taskDao().updateTask(taskEntry);  
 }  
 }  
 });  
 }  
 })  
 .setMessage( "Do you want to un-check all tasks?")  
 .show();  
 break;  
 case R.id.sort\_by\_priority:  
 new AlertDialog.Builder(this)  
 .setCancelable(false)  
 .setTitle("Sort tasks")  
 .setPositiveButton("No", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 dialog.cancel();  
 }  
 })  
 .setNegativeButton("Yes", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 AppExecutors.getInstance().diskIO().execute(new Runnable() {  
 @Override  
 public void run() {  
 try {  
 final List<TaskEntry> tasks = mAdapter.getTasks();  
 mDb.taskDao().deleteAll();  
 Collections.sort(tasks, new Comparator<TaskEntry>(){  
 public int compare(TaskEntry o1, TaskEntry o2){  
 return o1.getPriority() - o2.getPriority();  
 }  
 });  
  
 for(int i =0; i < tasks.size(); i++){  
 TaskEntry task = tasks.get(i);  
 mDb.taskDao().insertTask(new TaskEntry(  
 task.getDescription(),  
 task.getPriority(),  
 task.getUpdatedAt(),  
 task.isChecked()  
 ));  
 }  
 }catch (Exception ignored){}  
 }  
 });  
 }  
 })  
 .setMessage( "Do you want to sort list by priority?")  
 .show();  
 break;  
 case R.id.sort\_by\_check:  
 new AlertDialog.Builder(this)  
 .setCancelable(false)  
 .setTitle("Show pending tasks first")  
 .setPositiveButton("No", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 dialog.cancel();  
 }  
 })  
 .setNegativeButton("Yes", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 AppExecutors.getInstance().diskIO().execute(new Runnable() {  
 @Override  
 public void run() {  
 try {  
 final List<TaskEntry> tasks = mAdapter.getTasks();  
 mDb.taskDao().deleteAll();  
  
 Collections.sort(tasks, new Comparator<TaskEntry>() {  
 @Override  
 public int compare(TaskEntry o1, TaskEntry o2) {  
 return Boolean.compare(o1.isChecked(), o2.isChecked());  
 }  
 });  
  
  
 for(int i =0; i < tasks.size(); i++){  
 TaskEntry task = tasks.get(i);  
 mDb.taskDao().insertTask(new TaskEntry(  
 task.getDescription(),  
 task.getPriority(),  
 task.getUpdatedAt(),  
 task.isChecked()  
 ));  
 }  
 }catch (Exception ignored){}  
 }  
 });  
 }  
 })  
 .setMessage( "Do you want to sort list to show pending tasks first?")  
 .show();  
 return true;  
 }  
 return super.onOptionsItemSelected(item);  
 }  
  
  
 @Override  
 public void onItemClickListener(int itemId) {  
 Intent intent = new Intent(MainActivity.this, AddTaskActivity.class);  
 intent.putExtra(AddTaskActivity.EXTRA\_TASK\_ID, itemId);  
 startActivity(intent);  
 }  
  
 @Override  
 public void onCheckBoxCheckListener(final TaskEntry taskEntry) {  
 AppExecutors.getInstance().diskIO().execute(new Runnable() {  
 @Override  
 public void run() {  
 mDb.taskDao().updateTask(taskEntry);  
 }  
 });  
 }  
  
  
 private void setupViewModel() {  
 MainViewModel viewModel = new ViewModelProvider(this).get(MainViewModel.class);  
 viewModel.getTasks().observe(this, new Observer<List<TaskEntry>>() {  
 @Override  
 public void onChanged(List<TaskEntry> taskEntries) { //runs on main thread  
  
  
 if(taskEntries.isEmpty()){  
 mprogressBar.setVisibility(View.INVISIBLE);  
 mProgressValue.setVisibility(View.INVISIBLE);  
 mRecyclerView.setVisibility(View.INVISIBLE);  
 mEmptyView.setVisibility(View.VISIBLE);  
 }else {  
 mprogressBar.setVisibility(View.VISIBLE);  
 mProgressValue.setVisibility(View.VISIBLE);  
 mRecyclerView.setVisibility(View.VISIBLE);  
 mEmptyView.setVisibility(View.GONE);  
 }  
  
 calculatePercent(taskEntries);  
 mprogressBar.setProgress((int)mTotalProgressPercent);  
 mProgressValue.setText((int)mTotalProgressPercent + " %");  
  
 mAdapter.setTasks(taskEntries);  
 }  
 });  
 }  
  
 private void calculatePercent(List<TaskEntry> taskEntries) {  
 int countChecked = 0;  
 for(TaskEntry i: taskEntries){  
 if(i.isChecked()) countChecked++;  
 }  
 mTotalProgressPercent = (double)countChecked/taskEntries.size() \*100;  
 }  
}

**MAINVIEWMODEL.JAVA:**

package com.skapps.android.todolist;  
  
import android.app.Application;  
  
import com.skapps.android.todolist.database.AppDatabase;  
import com.skapps.android.todolist.database.TaskEntry;  
  
import java.util.List;  
  
import androidx.annotation.NonNull;  
import androidx.lifecycle.AndroidViewModel;  
import androidx.lifecycle.LiveData;  
  
public class MainViewModel extends AndroidViewModel {  
  
 private LiveData<List<TaskEntry>> tasks;  
  
 public MainViewModel(@NonNull Application application){  
 super(application);  
 AppDatabase database = AppDatabase.*getInstance*(this.getApplication());  
 tasks = database.taskDao().loadAllTasks();  
 }  
  
 public LiveData<List<TaskEntry>> getTasks() {  
 return tasks;  
 }  
}

**TASKADAPTOR.JAVA:**

package com.skapps.android.todolist;  
  
import android.content.Context;  
import android.graphics.Color;  
import android.graphics.drawable.GradientDrawable;  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.CheckBox;  
import android.widget.TextView;  
  
import com.skapps.android.todolist.database.TaskEntry;  
  
import java.text.SimpleDateFormat;  
import java.util.List;  
import java.util.Locale;  
  
import androidx.annotation.NonNull;  
import androidx.core.content.ContextCompat;  
import androidx.recyclerview.widget.RecyclerView;  
  
public class TaskAdapter extends RecyclerView.Adapter<TaskAdapter.TaskViewHolder> {  
  
  
 private static final String *DATE\_FORMAT* = "dd/MM/yyy";  
  
 final private ItemClickListener mItemClickListener;  
 final private CheckBoxCheckListener mCheckBoxCheckListener;  
 // Class variables for the List that holds task data and the Context  
 private List<TaskEntry> mTaskEntries;  
 private Context mContext;  
  
 // Date formatter  
 private SimpleDateFormat dateFormat = new SimpleDateFormat(*DATE\_FORMAT*, Locale.*getDefault*());  
  
 public TaskAdapter(Context context, ItemClickListener listener, CheckBoxCheckListener mCheckBoxCheckListener) {  
 mContext = context;  
 mItemClickListener = listener;  
 this.mCheckBoxCheckListener = mCheckBoxCheckListener;  
 }  
  
  
 @NonNull  
 @Override  
 public TaskViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 // Inflate the task\_layout to a view  
 View view = LayoutInflater.from(mContext)  
 .inflate(R.layout.task\_layout, parent, false);  
  
 return new TaskViewHolder(view);  
 }  
  
  
 @Override  
 public void onBindViewHolder(TaskViewHolder holder, int position) {  
 // Determine the values of the wanted data  
 TaskEntry taskEntry = mTaskEntries.get(position);  
 String description = taskEntry.getDescription();  
 int priority = taskEntry.getPriority();  
 String updatedAt = dateFormat.format(taskEntry.getUpdatedAt());  
  
  
 holder.taskDescriptionView.setText(description);  
 holder.updatedAtView.setText(updatedAt);  
  
 holder.priorityView.setText(Integer.toString(position+1));  
  
  
 holder.checkBox.setChecked(taskEntry.isChecked());  
  
 if(taskEntry.isChecked()){  
 holder.taskDescriptionView.setBackgroundResource(R.drawable.strike\_through);  
 holder.taskDescriptionView.setTextColor(Color.GRAY);  
  
 }else {  
 holder.taskDescriptionView.setBackgroundResource(0);  
 holder.taskDescriptionView.setTextColor(ContextCompat.getColor(mContext, R.color.list\_item\_text\_color));  
  
  
 }  
  
  
 GradientDrawable priorityCircle = (GradientDrawable) holder.priorityView.getBackground();  
  
 int priorityColor = getPriorityColor(priority);  
 priorityCircle.setColor(priorityColor);  
  
  
 }  
  
 /\*  
 Helper method for selecting the correct priority circle color.  
 P1 = red, P2 = orange, P3 = yellow  
 \*/  
 private int getPriorityColor(int priority) {  
 int priorityColor = 0;  
  
 switch (priority) {  
 case 1:  
 priorityColor = ContextCompat.getColor(mContext, R.color.materialRed);  
 break;  
 case 2:  
 priorityColor = ContextCompat.getColor(mContext, R.color.materialOrange);  
 break;  
 case 3:  
 priorityColor = ContextCompat.getColor(mContext, R.color.materialYellow);  
 break;  
 default:  
 break;  
 }  
 return priorityColor;  
 }  
  
  
 @Override  
 public int getItemCount() {  
 if (mTaskEntries == null) {  
 return 0;  
 }  
 return mTaskEntries.size();  
 }  
  
 public List<TaskEntry> getTasks() {  
 return mTaskEntries;  
 }  
  
  
  
 public void setTasks(List<TaskEntry> taskEntries) {  
 mTaskEntries = taskEntries;  
 notifyDataSetChanged();  
 }  
  
  
  
 public interface ItemClickListener {  
 void onItemClickListener(int itemId);  
 }  
  
 public interface CheckBoxCheckListener {  
 void onCheckBoxCheckListener(TaskEntry taskEntry);  
 }  
  
  
 // Inner class for creating ViewHolders  
 class TaskViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {  
  
  
 TextView taskDescriptionView;  
 TextView updatedAtView;  
 TextView priorityView;  
 CheckBox checkBox;  
  
  
 public TaskViewHolder(View itemView) {  
 super(itemView);  
  
 taskDescriptionView = itemView.findViewById(R.id.taskDescription);  
 updatedAtView = itemView.findViewById(R.id.taskUpdatedAt);  
 priorityView = itemView.findViewById(R.id.priorityTextView);  
 checkBox = itemView.findViewById(R.id.checkBox);  
 itemView.setOnClickListener(this);  
  
 checkBox.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if(mTaskEntries.get(getAdapterPosition()).isChecked()){  
 mTaskEntries.get(getAdapterPosition()).setChecked(false);  
 }else {  
 mTaskEntries.get(getAdapterPosition()).setChecked(true);  
 }  
 Log.d("adpatercheck", "is checked " + mTaskEntries.get(getAdapterPosition()).isChecked());  
 mCheckBoxCheckListener.onCheckBoxCheckListener(mTaskEntries.get(getAdapterPosition()));  
 }  
 });  
  
 }  
  
 @Override  
 public void onClick(View view) {  
 int elementId = mTaskEntries.get(getAdapterPosition()).getId();  
 mItemClickListener.onItemClickListener(elementId);  
 }  
  
 }  
}

**XML CODES:**

**ACTIVITY\_ADD\_TASK.XML:**

<?xml version="1.0" encoding="utf-8"?>  
  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:paddingBottom="@dimen/activity\_vertical\_margin"  
 android:paddingEnd="@dimen/activity\_horizontal\_margin"  
 android:paddingLeft="@dimen/activity\_horizontal\_margin"  
 android:paddingRight="@dimen/activity\_horizontal\_margin"  
 android:paddingStart="@dimen/activity\_horizontal\_margin"  
 android:paddingTop="@dimen/activity\_vertical\_margin">  
  
 <EditText  
 android:id="@+id/editTextTaskDescription"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="start"  
 android:hint="@string/edit\_task\_description"  
 android:paddingBottom="@dimen/activity\_horizontal\_margin"  
 android:autofillHints="Task Description"  
 android:inputType="text" />  
  
  
 <TextView  
 style="@style/TextAppearance.AppCompat.Medium"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="start"  
 android:layout\_marginBottom="8dp"  
 android:layout\_marginTop="8dp"  
 android:text="@string/priority\_string"  
 android:textColor="@android:color/primary\_text\_light" />  
  
  
 <RadioGroup  
 android:id="@+id/radioGroup"  
 android:layout\_width="match\_parent"  
 android:layout\_height="40dp"  
 android:layout\_gravity="center"  
 android:layout\_marginBottom="@dimen/activity\_horizontal\_margin"  
 android:orientation="horizontal"  
 android:weightSum="3">  
  
 <RadioButton  
 android:id="@+id/radButton1"  
 android:layout\_width="0dp"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:background="@color/materialRed"  
 android:checked="true"  
 android:text="@string/high\_priority"  
 android:theme="@style/WhiteRadioButton" />  
  
 <RadioButton  
 android:id="@+id/radButton2"  
 android:layout\_width="0dp"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:background="@color/materialOrange"  
 android:text="@string/med\_priority"  
 android:theme="@style/WhiteRadioButton" />  
  
 <RadioButton  
 android:id="@+id/radButton3"  
 android:layout\_width="0dp"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:background="@color/materialYellow"  
 android:text="@string/low\_priority"  
 android:theme="@style/WhiteRadioButton" />  
  
 </RadioGroup>  
  
  
 <Button  
 android:id="@+id/saveButton"  
 style="@style/TextAppearance.AppCompat.Large"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:background="@color/colorPrimary"  
 android:text="@string/add\_button"  
 android:textColor="@android:color/white" />  
  
</LinearLayout>

**ACTIVITY\_MAIN.XML:**

<?xml version="1.0" encoding="utf-8"?>  
  
<FrameLayout 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">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:id="@+id/constraintLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_marginBottom="64dp"  
 android:orientation="horizontal">  
  
 <ProgressBar  
 android:id="@+id/progressBar"  
 style="@android:style/Widget.ProgressBar.Horizontal"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="16dp"  
 android:layout\_marginEnd="16dp"  
 android:layout\_marginBottom="16dp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 android:progressDrawable="@drawable/progress\_bar"/>  
  
 <TextView  
 android:id="@+id/progressValue"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginEnd="8dp"  
 android:text="0%"  
 android:textColor="@android:color/white"  
 android:textStyle="bold"  
 app:layout\_constraintBottom\_toBottomOf="@+id/progressBar"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="@+id/progressBar"  
 app:layout\_constraintTop\_toTopOf="@+id/progressBar" />  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/recyclerViewTasks"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:clipToPadding="false"  
 android:paddingBottom="100dp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/progressBar" />  
  
 <TextView  
 android:id="@+id/emptyView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 android:text="@string/you\_ve\_got\_nothing\_to\_do"  
 style="@style/ListItemTextStyle"  
 android:drawableTop="@drawable/ic\_empty\_img"  
 android:drawablePadding="16dp"  
 android:visibility="gone"  
 />  
  
 </androidx.constraintlayout.widget.ConstraintLayout>  
  
 <include layout="@layout/content\_main" />  
  
</FrameLayout>

**CONTENT\_MAIN.XML:**

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:orientation="vertical"  
 android:layout\_gravity="bottom"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
  
  
 <!-- linearLayour for fab button-->  
 <LinearLayout  
 android:layout\_weight="1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:gravity="end">  
  
 <com.google.android.material.floatingactionbutton.FloatingActionButton  
 android:id="@+id/fab"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="bottom|end|start"  
 android:layout\_margin="@dimen/fab\_margin"  
 android:backgroundTint="@color/materialRed"  
 android:tint="@android:color/white"  
 app:srcCompat="@android:drawable/ic\_input\_add" />  
 </LinearLayout>  
  
</LinearLayout>

**TASK\_LAYOUT.XML:**

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:paddingBottom="8dp"  
 android:paddingLeft="@dimen/activity\_horizontal\_margin"  
 android:paddingRight="@dimen/activity\_horizontal\_margin"  
 android:paddingTop="8dp"  
 tools:context="com.skapps.android.todolist.MainActivity">  
  
  
 <TextView  
 android:id="@+id/priorityTextView"  
 android:layout\_width="28dp"  
 android:layout\_height="28dp"  
 android:layout\_gravity="center\_vertical"  
 android:background="@drawable/priority\_circle"  
 android:gravity="center"  
 android:layout\_weight="1"  
 android:textAlignment="center"  
 android:layout\_marginEnd="16dp"  
 android:textColor="@android:color/white"  
 tools:text="1" />  
  
 <LinearLayout  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1135"  
 android:orientation="vertical">  
  
  
 <TextView  
 android:id="@+id/taskDescription"  
 style="@style/TextAppearance.AppCompat.Medium"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textColor="@color/list\_item\_text\_color"  
 tools:text="Description"/>  
  
 <TextView  
 android:id="@+id/taskUpdatedAt"  
 style="@style/TextAppearance.AppCompat.Small"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:textColor="#D2AFAFAF"  
 tools:text="Note" />  
  
 </LinearLayout>  
  
 <CheckBox  
 android:id="@+id/checkBox"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:scaleX="1.40"  
 android:scaleY="1.40"  
 android:layout\_weight="1" />  
  
</LinearLayout>