```
@Entity
public class Artwork  {

    @PrimaryKey(autoGenerate = true)
    public int local_id;

    @ColumnInfo(name = "title")
    public String title;
    @ColumnInfo(name = "artist")
    String artist_display;
    String id;
    String image_id;
    int source;

    @Ignore
    String medium_display;
    String dimensions;
    String description;

    public Artwork() { }

    public Artwork(String id, String title,String image_id,int source) {
        this.id = id;
        this.title = title;
        this.image_id = image_id;
        this.source = source;
    }

    //constructor for list display
    public Artwork(String id, String title,int source) {
        this.id = id;
        this.title = title;
        this.source = source;

    }

    public Artwork(String image_id) {
        this.image_id = image_id;
    }

    public Artwork(String id, String title, String artist_display, String medium_display, String
dimensions, String image_id, String description,int source) {
        this.id = id;
        this.title = title;
        this.artist_display = artist_display;
        this.medium_display = medium_display;
        this.dimensions = dimensions;
        this.image_id = image_id;
        this.description = description;
        this.source = source;
    }
}

public class myApp extends Application {
```

```
    public JsonService getJsonService() {
        return jsonService;
    }
    public NetworkingService getNetworkingService() {
        return networkingService;
    }
    private JsonService jsonService = new JsonService();
    private NetworkingService networkingService = new NetworkingService();
    private DatabaseManager databaseManager =new DatabaseManager();
    public DatabaseManager getDatabaseManager() {
        return databaseManager;
    }
}

public class MainActivity extends AppCompatActivity implements
ResultAdapter.artClickListener,
        NetworkingService.networkingListener {
    ResultAdapter adapter;
    NetworkingService networkingService;
    //    ArrayList<Artwork> artListC;
    ArrayList<Artwork> artListR = new ArrayList<>(0);
    //    ArrayList<Artwork> fullList;
    JsonService jsonService;
    RecyclerView recyclerView;
    FavDatabase db;
    DatabaseManager dbManager;
    Bundle savedInstanceState;
    ProgressBar loading;
    String myQuery;
    ImageView cover_img;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        networkingService = ((myApp)getApplication()).getNetworkingService();
        jsonService = ((myApp)getApplication()).getJsonService();
        db = DatabaseManager.getDBInstance(this);
        dbManager = ((myApp)getApplication()).getDatabaseManager();
        loading=findViewById(R.id.progressBar);
        cover_img = findViewById(R.id.cover);
//        networkingService.fetchAllArtListData();
//        networkingService.fetchArtListData(myQuery);
//TODO:show last query result when come back from detail page
//        if (savedInstanceState != null) {
//            myQuery = savedInstanceState.getString("query");
//            System.out.println(myQuery);
//            networkingService.fetchArtListData(myQuery);
//        }
        //combine two arraylist for display
//        fullList.addAll(artListC);
//        fullList.addAll(artListR);
```

```
        recyclerView = findViewById(R.id.resultGrid);
        int numberOfColumns = 2;
        recyclerView.setLayoutManager(new GridLayoutManager(this,numberOfColumns));
//        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        adapter = new ResultAdapter(this,artListR);
        recyclerView.setAdapter(adapter);
        adapter.listener=this;
        networkingService.listener = this;
    }

//    @Override
//    public void onSaveInstanceState(Bundle savedInstanceState) {
//        super.onSaveInstanceState(savedInstanceState);
//        savedInstanceState.putString("query",myQuery);
//    }
    @Override
    protected void onResume() {
        super.onResume();
        networkingService = ( (myApp)getApplication()).getNetworkingService();
        jsonService = ( (myApp)getApplication()).getJsonService();
        networkingService.listener = this;
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu,menu);
        MenuItem searchItem = menu.findItem(R.id.search);
        SearchView searchView = (SearchView) searchItem.getActionView();
        String keyword = searchView.getQuery().toString();
        if(!keyword.isEmpty()){
            searchView.setQuery(keyword,false);
            searchView.setIconified(false);}
        searchView.setQueryHint("Search for specific subjects");
        searchView.setSubmitButtonEnabled(true);
        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String query) {
                networkingService.fetchArtListData(query);
                searchView.clearFocus();
                myQuery=query;
                loading.setVisibility(View.VISIBLE);
                cover_img.setVisibility(View.INVISIBLE);
                return true;
            }

            @Override
            public boolean onQueryTextChange(String newText) {
                return false;
            }
        });
        return true;
```

```
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        super.onOptionsItemSelected(item);
        switch(item.getItemId()){
            case R.id.fav:{
                Intent toFav = new Intent(getApplicationContext(),FavActivity.class);
                startActivity(toFav);
                break;
                }
        }
        return true;
    }

    @Override
    public void artSelected(Artwork selectedArt) {
        Log.d("select","ArtSelected!#@######");
        Intent toDetail = new Intent(this,DetailActivity.class);
        toDetail.putExtra("selectID",selectedArt.id);
        startActivity(toDetail);
    }

    @Override
    public void APIlistener(String jsonString) {
//          artListC = jsonService.parseArtListJsonChi(jsonString);
        artListR = jsonService.parseArtListJsonRijks(jsonString);
        for (int i = 0; i < artListR.size(); i++) {
            networkingService.getImageDatafromRijks(artListR.get(i).image_id);
        }
        System.out.println("Size:"+artListR.size());
        //if no result is found
        if (artListR.size() == 0) {
            Toast toast = Toast.makeText(this,"No result available",Toast.LENGTH_LONG);
                toast.setGravity(Gravity.CENTER,0,0);
            toast.show();
        }
        adapter.artList = artListR;
        adapter.notifyDataSetChanged();
        loading.setVisibility(View.GONE);
    }

    @Override
    public void APIImgListener(Bitmap image) {
    }
}

public class ResultAdapter extends RecyclerView.Adapter<ResultAdapter.myViewHolder>{

    Context c;
    public ArrayList<Artwork> artList;

    //Adapter for homeActivity's recyclerview
```

```
    public ResultAdapter(Context c, ArrayList<Artwork> artList) {
        this.c = c;
        this.artList = artList;
    }

    interface artClickListener {
        void artSelected(Artwork selectedArt);
    }

    public artClickListener listener;

    public class myViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener{

        private final TextView txtView;
        private final ImageView imgView;
        public TextView getTxtView() {
            return txtView;
        }
        public ImageView getImgView() {
            return imgView;
        }

        public myViewHolder(@NonNull View itemView) {
            super(itemView);
            imgView = itemView.findViewById(R.id.result_img);
            txtView = itemView.findViewById(R.id.result_title);
            itemView.setOnClickListener(this);
        }

        @Override
        public void onClick(View v) {
            Artwork artwork = artList.get(getAdapterPosition());
            listener.artSelected(artwork);
        }
    }

    @NonNull
    @Override
    public ResultAdapter.myViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view = LayoutInflater.from(c).inflate(R.layout.result_cell,parent,false);
        return new myViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull myViewHolder holder, int position) {
        holder.getTxtView().setText(artList.get(position).title);
        Glide.with(c)
                .load(artList.get(position).image_id)
                .centerCrop()
                .into(holder.getImgView());
        holder.itemView.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
                listener.artSelected(artList.get(position));
            }
        });
    }

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


public class FavActivity extends AppCompatActivity implements

DatabaseManager.DatabaseListener,NetworkingService.networkingListener,FavRadapter.artCli
ckListener{

    ArrayList<Artwork> favDBList;
    DatabaseManager dbManager;
    RecyclerView recyclerView;
    FavRadapter rAdapter;
    NetworkingService networkingService;
    JsonService jsonService;
    AlertDialog.Builder warning;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_fav);
        recyclerView = findViewById(R.id.fav_view);
        registerForContextMenu(recyclerView);
        networkingService = ((myApp)getApplication()).getNetworkingService();
        jsonService=((myApp)getApplication()).getJsonService();

        dbManager =((myApp)getApplication()).getDatabaseManager();
        dbManager.listener = this;
        dbManager.getAllArt();

        warning = new AlertDialog.Builder(this);
    }

    @Override
    public void databaseAllFavArtList(List<Artwork> list) {
        favDBList = new ArrayList<>(list);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        rAdapter = new FavRadapter(favDBList,this);
        recyclerView.setAdapter(rAdapter);
        rAdapter.listener=this;
    }

    @Override
```

```
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.fav_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        super.onOptionsItemSelected(item);
        switch (item.getItemId()) {
            case R.id.remove_all: {
                showAlert();
                break;
            }
        }
        return true;
    }
//TODO: delete individual saved work with context menu
//    @Override
//    public void onCreateContextMenu(ContextMenu menu, View v,
//                                    ContextMenu.ContextMenuInfo menuInfo) {
//        super.onCreateContextMenu(menu, v, menuInfo);
//        MenuInflater inflater = getMenuInflater();
//        inflater.inflate(R.menu.context_menu, menu);
//    }
//
//    @Override
//    public boolean onContextItemSelected(MenuItem item) {
//        AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo)
item.getMenuInfo();
//        switch (item.getItemId()) {
//            case R.id.delete:
////                dbManager.deleteArt(favDBList.get(info.position));
//                dbManager.deleteArt(favDBList.get(info.position));
//                rAdapter.notifyDataSetChanged();
//                break;
//        }
//        return super.onContextItemSelected(item);
//    }

    @Override
    public void APIlistener(String jsonString) {
    }

    @Override
    public void APIImgListener(Bitmap image) {
    }

    @Override
    public void favArtSelected(Artwork selectedArt) {
//        TODO:Fullscreen activity???
//         Intent toFull = new Intent(this,FullimgActivity.class);
//        toFull.putExtra("img_url",selectedArt.image_id);
```

```
//        startActivity(toFull);
        Intent i = new Intent(Intent.ACTION_VIEW);
        i.setData(Uri.parse(selectedArt.image_id));
        startActivity(i);
    }

    private void showAlert(){
        warning.create();
        warning.setMessage("You will delete all saved artwork.");
        warning.setPositiveButton("Confirm", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dbManager.deleteAll();
                Intent backHome= new Intent(getApplicationContext(),MainActivity.class);
                startActivity(backHome);
            }
        });
        warning.setCancelable(true);
        warning.show();
    }
}
public class FavRadapter extends RecyclerView.Adapter<FavRadapter.viewHolder> {

   ArrayList<Artwork> listOfArt;
    Context fav_context;

    interface artClickListener {
        void favArtSelected(Artwork selectedArt);
    }

    public FavRadapter.artClickListener listener;

    public FavRadapter(ArrayList<Artwork> listOfArt, Context list_context) {
        this.listOfArt = listOfArt;
        this.fav_context = list_context;
    }

    public class viewHolder extends RecyclerView.ViewHolder implements
View.OnCreateContextMenuListener,View.OnClickListener {
        public ImageView getImg() {
            return imgView;
        }

        public TextView getTitle() {
            return titleView;
        }

        private final ImageView imgView;
        private final TextView titleView;

        public viewHolder(@NonNull View itemView) {
            super(itemView);
            imgView = itemView.findViewById(R.id.fav_img);
```

```
            titleView = itemView.findViewById(R.id.fav_title);
            itemView.setOnClickListener(this);


        }

        @Override
        public void onCreateContextMenu(ContextMenu menu, View v,
ContextMenu.ContextMenuInfo menuInfo) {
            menu.add(Menu.NONE, R.id.delete,0,"REMOVE");
        }

        @Override
        public void onClick(View v) {
            Artwork artwork = listOfArt.get(getAdapterPosition());
            listener.favArtSelected(artwork);
        }
    }

    @NonNull
    @Override
    public FavRadapter.viewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View v = LayoutInflater.from(fav_context).inflate(R.layout.fav_cell,parent,false);
        return new viewHolder(v);
    }

    @Override
    public void onBindViewHolder(@NonNull viewHolder holder, int position) {
        holder.getTitle().setText(listOfArt.get(position).title);
        Glide.with(fav_context)
                .load(listOfArt.get(position).image_id)
                .centerCrop()
                .into(holder.getImg());

        //trigger context menu by long press
        holder.itemView.setOnLongClickListener(new View.OnLongClickListener() {
            @Override
            public boolean onLongClick(View v) {
                holder.getAdapterPosition();
                return false;
            }
        });
        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                listener.favArtSelected(listOfArt.get(position));
            }
        });
    }
    @Override
    public int getItemCount() {
        return listOfArt.size();
```

```
    }
}

public class JsonService {
    ArrayList<Artwork> fullArtList = new ArrayList<>(0);
    //keyword search result: a list of object with basic info from Chicago museum, only image
need to be displayed
    //json structure see:jsonArtList - CHI
//    public ArrayList<Artwork> parseArtListJsonChi(String jsonThumbStr) {
//        ArrayList<Artwork> fullArtListChi = new ArrayList<>(0);
//        try {
//            JSONObject jsonObject = new JSONObject(jsonThumbStr);
//            JSONArray artArray = jsonObject.getJSONArray("data");
//            for(int i=0;i<artArray.length();i++) {
//                JSONObject thumbObj = artArray.getJSONObject(i);
//                int id = thumbObj.getInt("id");
//                String title = thumbObj.getString("title");
//                int source = 1;
//                fullArtList.add(new Artwork(String.valueOf(id),title,source));
//            }
//        } catch (JSONException e) {
//            e.printStackTrace();
//        }
//        return fullArtListChi;
//    }

    //json structure see:jsonArtList - RIJKS
    public ArrayList<Artwork> parseArtListJsonRijks(String jsonThumbStr) {
        ArrayList<Artwork> fullArtListR = new ArrayList<>(0);
        try {
            JSONObject jsonObject = new JSONObject(jsonThumbStr);
            JSONArray artArray = jsonObject.getJSONArray("artObjects");
            for(int i=0;i<artArray.length();i++) {
                JSONObject thumbObj = artArray.getJSONObject(i);
                String id = thumbObj.getString("objectNumber");
                String title = thumbObj.getString("title");
                JSONObject imgObj = thumbObj.getJSONObject("webImage");
                String image_url = imgObj.getString("url");
                int source = 2;
                Artwork artResult = new Artwork(id,title,image_url,source);
                fullArtListR.add(artResult);
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return fullArtListR;
    }

//    public ArrayList<Artwork> parseListJsonMet(String jsonSearchStr) {
//        ArrayList<Artwork> fullArtListMet = new ArrayList<>(0);
//    }

//detail info for called after click one artwork, displayed on detailActivity
    //json structure: jsonArtDetail-CHI
```

```
//
//    public String parseImgIDJsonChi(String jsonImgStr) {
//        Artwork artThumbnail = new Artwork();
//        JSONObject jsonObject = null;
//        try{
//            jsonObject = new JSONObject(jsonImgStr);
//            JSONObject detailObj = jsonObject.getJSONObject("data");
//            String imageID = detailObj.getString("image_id");
//
//        } catch (JSONException e) {
//            e.printStackTrace();
//        }
//        return imageID;
//    }
//
//    public Artwork parseArtDetailJsonChi(String jsonArtStr) {
//        Artwork artDetailData = new Artwork();
//        JSONObject jsonObject = null;
//        try {
//            jsonObject = new JSONObject(jsonArtStr);
//            JSONObject detailObj = jsonObject.getJSONObject("data");
//            int id = detailObj.getInt("id");
//            String title = detailObj.getString("title");
//            String artist = detailObj.getString("artist_display");
//            String dimensions = detailObj.getString("dimensions");
//            String medium = detailObj.getString("medium_display");
//            boolean zoomable = detailObj.getBoolean("is_zoomable");
//            String image = detailObj.getString("image_id");
//            int source = 1;
//            artDetailData = new
Artwork(String.valueOf(id),title,artist,medium,dimensions,image,null,source);
//        } catch (JSONException e) {
//            e.printStackTrace();
//        }
//        return artDetailData;
//    }

    public Artwork parseArtDetailJsonRijks(String jsonArtStr) {
        Artwork artDetailData = new Artwork();
        try {
            JSONObject jsonObject = new JSONObject(jsonArtStr);
            JSONObject detailObj = jsonObject.getJSONObject("artObject");
            String id = detailObj.getString("objectNumber");
            String title = detailObj.getString("title");
            JSONObject imageObj = detailObj.getJSONObject("webImage");
            String img_url = imageObj.getString("url");
            JSONArray makerArray = detailObj.getJSONArray("principalMakers");
            JSONObject makerObj = makerArray.getJSONObject(0);
            String artist = makerObj.getString("labelDesc");
            String dimensions = detailObj.getString("subTitle");
            String description = detailObj.getString("plaqueDescriptionEnglish");
            String medium = detailObj.getString("physicalMedium");
            int source = 2;
```

```
            artDetailData = new
Artwork(String.valueOf(id),title,artist,medium,dimensions,img_url,description,source);
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return artDetailData;
    }
}
public class NetworkingService {

    //TODO:add more museum APIs
//    String chiURL = "https://api.artic.edu/api/v1/artworks/";
//    String chiSearchURL1 = "search?q=";
//    String chiSearchURL2 = "&limit=20";
//
//    String chiImgURL1 = "https://www.artic.edu/iiif/2/";
//    String chiImgURL2 = "/full/843,/0/default.jpg";
//    String chiImgURLs = "/full/200,/0/default.jpg";
//
//    String ClevelandURL = "https://openaccess-api.clevelandart.org/api/artworks/?
has_image=1&q=";

    String rijksQURL = "https://www.rijksmuseum.nl/api/en/collection?";
    private String apiKeyR = "key=VaUtnjGb";
    String rijksSearch = "&imgonly=true&ps=10&what=";
    String rijksDetailURL = "https://www.rijksmuseum.nl/api/en/collection/";

    public static final ExecutorService networkingExecutor = Executors.newFixedThreadPool(4);
    static Handler networkHandler = new Handler(Looper.getMainLooper());

    interface networkingListener {
        void APIlistener(String jsonString);
        void APIImgListener(Bitmap image);
    }

    networkingListener listener;

    //fetch search results
    public void fetchAllArtListData() {
//        String completedURLChi = chiURL+chiSearchURL1 + text + chiSearchURL2;
        String completedURLRijks = rijksQURL+ apiKeyR + rijksSearch;
//        connect(completedURLChi);
        connect(completedURLRijks);
    }
    public void fetchArtListData(String text) {
//        String completedURLChi = chiURL+chiSearchURL1 + text + chiSearchURL2;
        String completedURLRijks = rijksQURL+ apiKeyR + rijksSearch + text;
//        connect(completedURLChi);
        connect(completedURLRijks);
    }

    public void fetchDetailData(String id) {
//        String completedURLChi = chiURL + id;
```

```
        String completedURLRijks = rijksDetailURL + id + "?" + apiKeyR;
//        connect(completedURLChi);
        connect(completedURLRijks);
    }

    //second call with object id to get img_id
//    public void getImgID(String id) {
//        String completedURL = chiURL + id;
//        connect(completedURL);
//    }

//    public void getThumbnailImageDatafromChi(String img_id) {
//        //parse img_id from object in model
//        String completedURL = chiImgURL1 + img_id +chiImgURLs;
//        //only get the image of smaller size
//        networkingExecutor.execute((new Runnable() {
//            @Override
//            public void run() {
//                URL urlObj = null;
//                try {
//                    urlObj = new URL(completedURL);
//                    InputStream in = ((InputStream) urlObj.getContent());
//                    Bitmap imageData = BitmapFactory.decodeStream(in);
//                    networkHandler.post(new Runnable() {
//                        @Override
//                        public void run() {
//                            listener.APIImgListener(imageData);
//                        }
//                    });
//                } catch (MalformedURLException e) {
//                    e.printStackTrace();
//                } catch (IOException e) {
//                    e.printStackTrace();
//                }
//            }
//        }));
//    }
//    public void getImageDatafromChi(String img_id) {
//        String completedURL = chiImgURL1 + img_id +chiImgURL2;
//        networkingExecutor.execute((new Runnable() {
//            @Override
//            public void run() {
//                URL urlObj = null;
//                try {
//                    urlObj = new URL(completedURL);
//                    InputStream in = ((InputStream) urlObj.getContent());
//                    Bitmap imageData = BitmapFactory.decodeStream(in);
//                    networkHandler.post(new Runnable() {
//                        @Override
//                        public void run() {
//                            listener.APIImgListener(imageData);
//                        }
//                    });
//                } catch (MalformedURLException e) {
```

```
//                    e.printStackTrace();
//                } catch (IOException e) {
//                    e.printStackTrace();
//                }
//            }
//        }));
//    }
    public void getImageDatafromRijks(String img_url) {
        networkingExecutor.execute((new Runnable() {
            @Override
            public void run() {
                URL urlObj = null;
                try {
                    urlObj = new URL(img_url);
                    InputStream in = ((InputStream) urlObj.getContent());
                    Bitmap imageData = BitmapFactory.decodeStream(in);
                    networkHandler.post(new Runnable() {
                        @Override
                        public void run() {
                            listener.APIImgListener(imageData);
                        }
                    });
                } catch (MalformedURLException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }));
    }

    private void connect(String url) {
        networkingExecutor.execute(new Runnable() {
            String jsonString = "";
            @Override
            public void run() {
                HttpURLConnection httpURLConnection = null;
                try {
                    URL urlObj = new URL(url);
                    httpURLConnection = (HttpURLConnection) urlObj.openConnection();
                    httpURLConnection.setRequestMethod("GET");
                    int status = httpURLConnection.getResponseCode();
                    if((status >=200) && (status <=299)) {
                        InputStream in = httpURLConnection.getInputStream();
                        InputStreamReader reader = new InputStreamReader(in);
                        int read = 0;
                        while((read = reader.read()) != -1) {
                            char c = (char)read;
                            jsonString += c;
                        }
                        final String finalJson = jsonString;
                        System.out.println("FinalJsonStr:" +finalJson);
                        networkHandler.post(new Runnable() {
```

```
                    @Override
                    public void run() {
                        listener.APIlistener(finalJson);
                    }
                });
            }
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        finally {
            httpURLConnection.disconnect();
        }
    }
});
    }
}
public final class MyGlideModule extends AppGlideModule {
    @Override
    public void applyOptions(Context c, GlideBuilder builder) {
        builder.setDefaultRequestOptions(new
RequestOptions().format(DecodeFormat.PREFER_ARGB_8888));
    }
}

@Database(version = 1,entities = {Artwork.class})
public abstract class FavDatabase extends RoomDatabase {
    abstract public DAO getDao();
}
@Dao
public interface DAO {
    @Query("SELECT * FROM Artwork")
    List<Artwork> getAll();

    @Insert
    void insertNewArt(Artwork artNew);

    @Delete
    void deleteArt(Artwork artDelete);

    @Query("DELETE FROM Artwork")
    public void delete();

//    @Query("DELETE FROM Artwork")
//    abstract void deleteArtQ(Artwork toDelete);
}

public class DatabaseManager {

    static FavDatabase db;
    ExecutorService databaseExecuter = Executors.newFixedThreadPool(4);
    Handler db_handler = new Handler(Looper.getMainLooper());
```

```
public interface DatabaseListener {
    void databaseAllFavArtList(List<Artwork> list);
}

public DatabaseListener listener;

private static void BuildDBInstance(Context c) {
    db = Room.databaseBuilder(c, FavDatabase.class, "art_db").build();
}

public static FavDatabase getDBInstance(Context c) {
    if (db == null) {
        BuildDBInstance(c);
    }
    return db;
}

public void addNewArt(Artwork art) {
    databaseExecuter.execute(new Runnable() {
        @Override
        public void run() {
            db.getDao().insertNewArt(art);
        }
    });
}

public void deleteArt(Artwork art) {
    databaseExecuter.execute(new Runnable() {
        @Override
        public void run() {
            db.getDao().deleteArt(art);
        }
    });
}

public void deleteAll() {
    databaseExecuter.execute(new Runnable() {
        @Override
        public void run() {
            db.getDao().delete();
        }
    });
}

public void getAllArt(){
    databaseExecuter.execute(new Runnable() {
        @Override
        public void run() {
            List<Artwork> list = db.getDao().getAll();
            db_handler.post(new Runnable() {
                @Override
                public void run() {
```

```
                    listener.databaseAllFavArtList(list);
                }
            });
        }
    });
  }
}
```

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


    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/resultGrid"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageView
        android:id="@+id/cover"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="centerCrop"
        app:srcCompat="@drawable/lion"
        />

    <ProgressBar
        android:id="@+id/progressBar"
        style="?android:attr/progressBarStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:indeterminate="true"
        android:visibility="invisible"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat xmlns:android="http://
schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
```

```
    android:background="@color/white"
    >

    <ImageView
        android:id="@+id/result_img"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="fitXY"

        />

    <TextView
        android:id="@+id/result_title"
        android:layout_width="match_parent"
        android:layout_height="30dp" />

</androidx.appcompat.widget.LinearLayoutCompat>

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

    <ImageView
        android:id="@+id/myImage"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:scaleType="centerCrop"
        app:layout_constraintBottom_toTopOf="@id/scroll"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ScrollView
        android:id="@+id/scroll"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        app:layout_constraintTop_toBottomOf="@id/myImage"
        >
    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:paddingStart="30dp"
        android:paddingEnd="30dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
```

```
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/myImage">

<TextView
    android:id="@+id/title"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:fontFamily="@font/nexa_bold"
    android:textColor="@color/white"
    android:textSize="20sp"
    android:textStyle="bold" />

<TextView
    android:id="@+id/artist"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:fontFamily="@font/source_italic"
    android:textColor="@color/white"
    android:textSize="20sp" />

<TextView
    android:id="@+id/dimension"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:fontFamily="@font/source_light"
    android:textColor="@color/white"
    android:textSize="17sp" />

<TextView
    android:id="@+id/medium"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:fontFamily="@font/source_light"
    android:textColor="@color/white"
    android:textSize="17sp" />

<TextView
    android:id="@+id/desc"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:fontFamily="@font/source_light"
    android:textColor="@color/white"
    android:textSize="17sp" />

<ImageButton
    android:id="@+id/save_btn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
```

```
            android:background="@android:color/transparent"
            android:onClick="save_item"
            app:srcCompat="@drawable/outline_favorite_border_white_24dp" />


    </LinearLayout>
   </ScrollView>
</androidx.constraintlayout.widget.ConstraintLayout>

<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat xmlns:android="http://
schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ImageView
        android:id="@+id/image_cover"

        android:layout_width="match_parent"
        android:layout_height="150dp"/>

    <TextView
        android:id="@+id/item_title"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:text="TextView"
        android:textColor="@color/white"
        android:background="@color/black" />

    <TextView
        android:id="@+id/item_source"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:textColor="@color/purple_500"
        android:text="TextView"
        android:background="@color/black" />
</androidx.appcompat.widget.LinearLayoutCompat>


<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/
android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".FavActivity"
    >
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/fav_view"
       android:layout_width="match_parent"
        android:layout_height="match_parent"
```

```
        android:background="@color/black"/>
</androidx.cardview.widget.CardView>
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat xmlns:android="http://
schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <ImageView
        android:id="@+id/fav_img"
        android:layout_width="match_parent"
        android:layout_height="140dp"

        />
    <TextView
        android:id="@+id/fav_title"
        android:layout_margin="5dp"
        android:fontFamily="@font/source_italic"
        android:textSize="17sp"
        android:textColor="@color/white"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
       />

</androidx.appcompat.widget.LinearLayoutCompat>

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

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <application
        android:name=".myApp"
        android:allowBackup="false"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ArtDictN"
        android:usesCleartextTraffic="true">
        <activity
            android:name=".FavActivity"
            android:exported="true">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.example.artdictn.MainActivity" />
        </activity>
        <activity
            android:name=".DetailActivity"
            android:exported="true">
            <meta-data
```

```
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.example.artdictn.MainActivity" />
        </activity>
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```