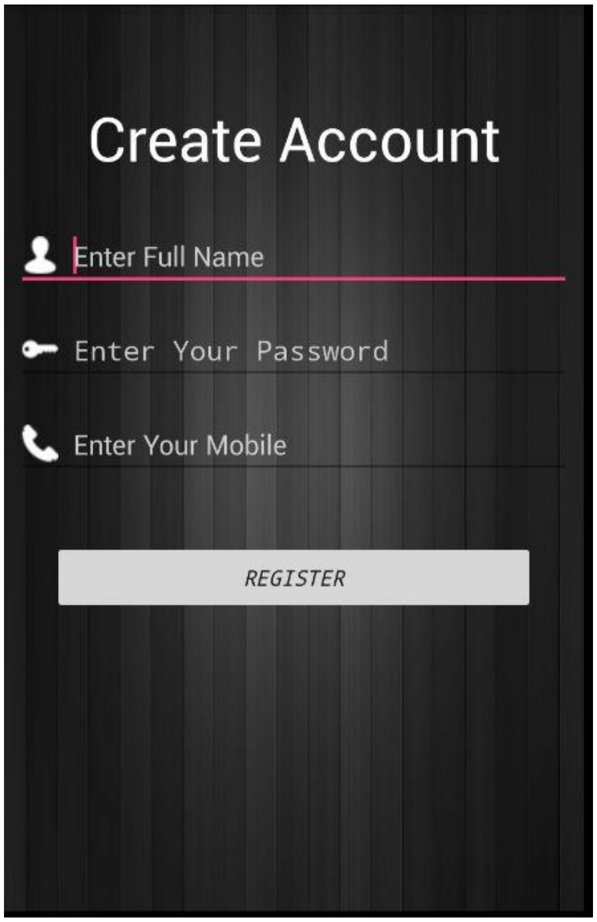
BookShop App

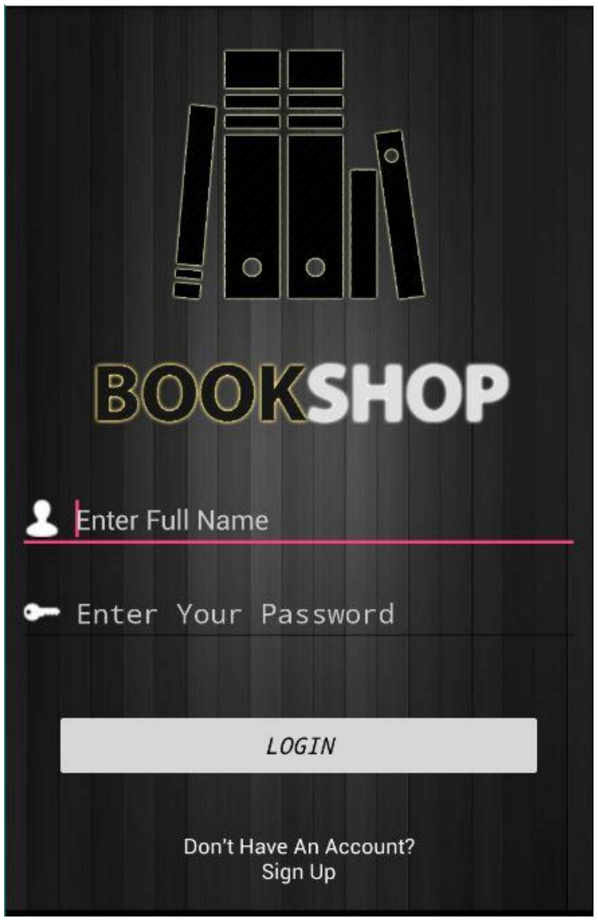
LAP 4

Team Members:
* Mahmoud Mohammed Hammed.
* Mostafa Ashraf Maamoun.
* Mostafa Ali Mohammed Ali.
* Mostafa Morsy Ismail.

Under the Guidance Of:
*Marah Mostafa.

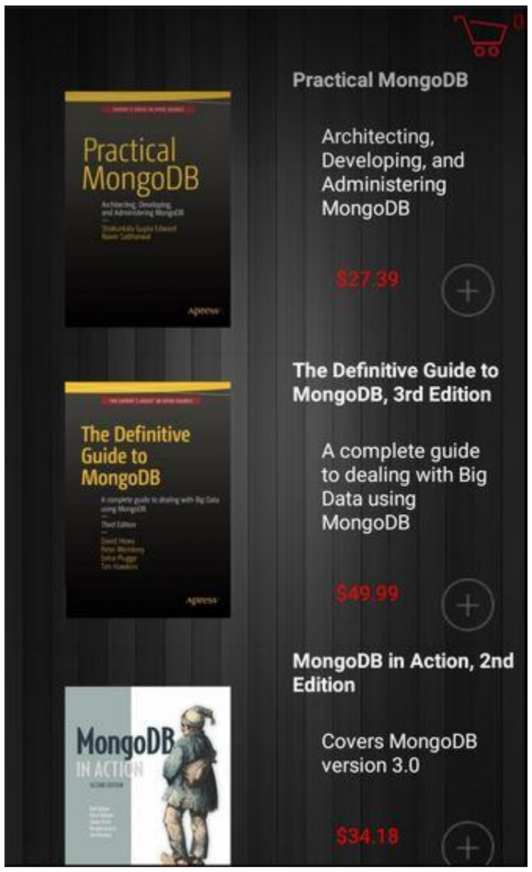
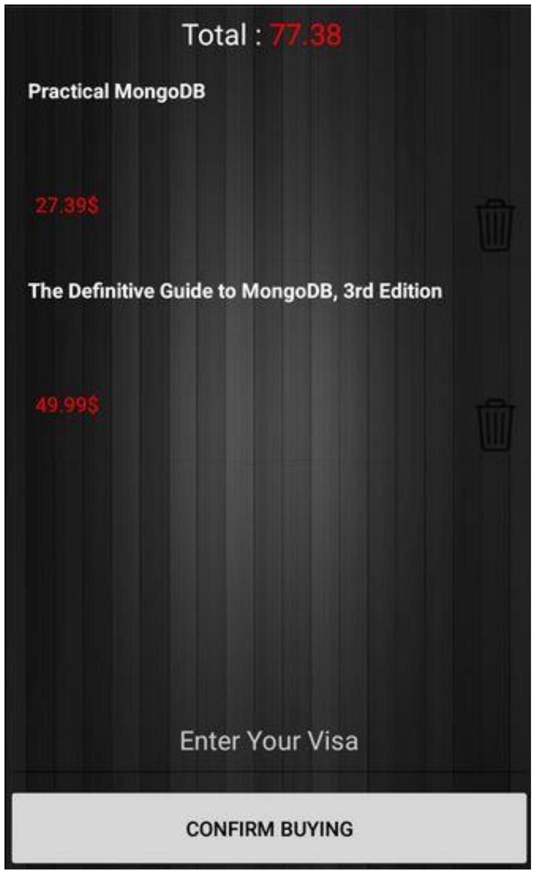
App Activities





1 -Login Activity.

2- Sign Up Activity.


0
Practical MongoDB
Architecting, Developing, and Administering MongoDB
$27.39
The Definitive Guide to MongoDB, 3rd Edition
A complete guide to dealing with Big Data using MongoDB
$49.99
MongoDB in Action, 2nd Edition
Covers MongoDB version 3.0
$34.18


3- BookList Activity.

4- Cart & Confirmation Activity.

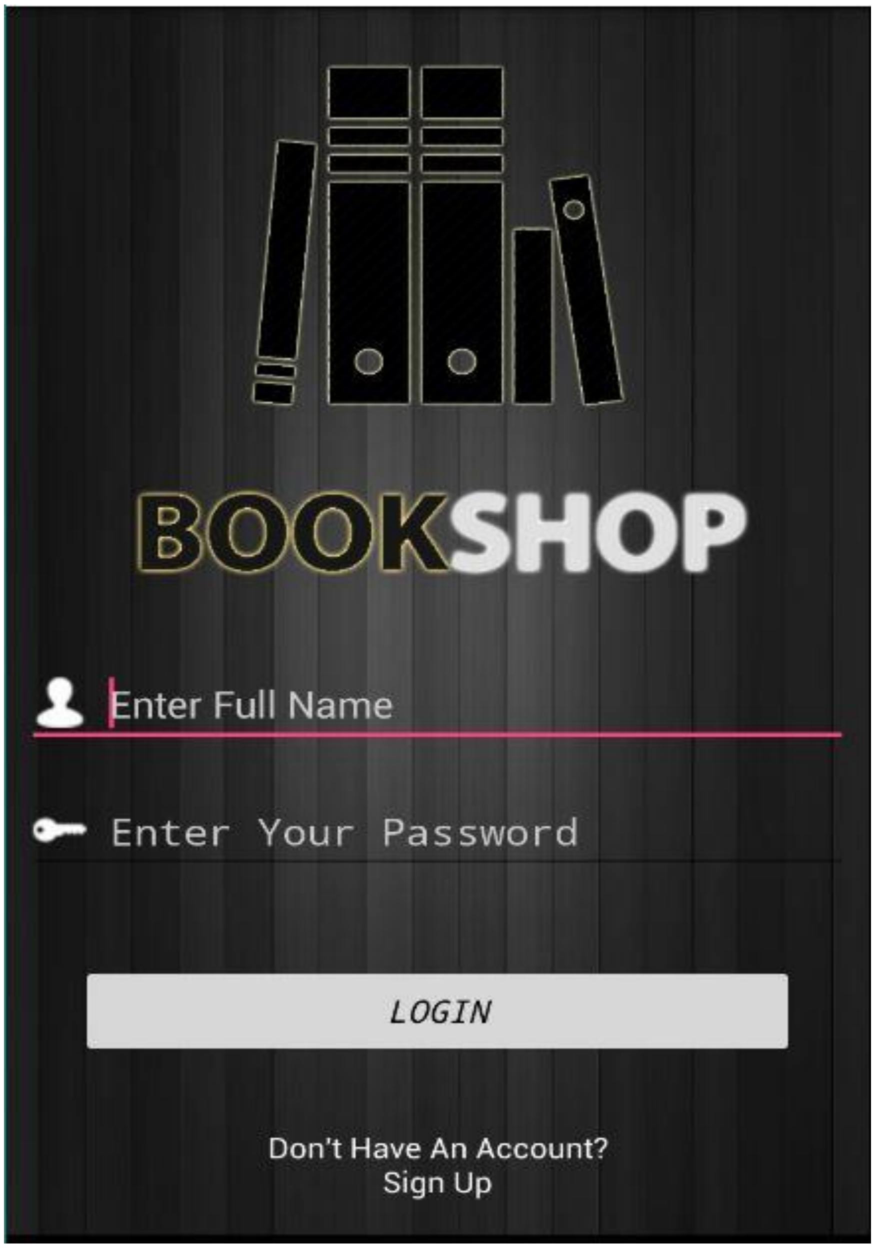
Project Classes:

- MainActivity (Login).
- NewAccount.
- DBAdapter (Database Adapter).
- Books.
- BookActivity.
- BookAdapter.
- CartActivity.
- CartAdapter.

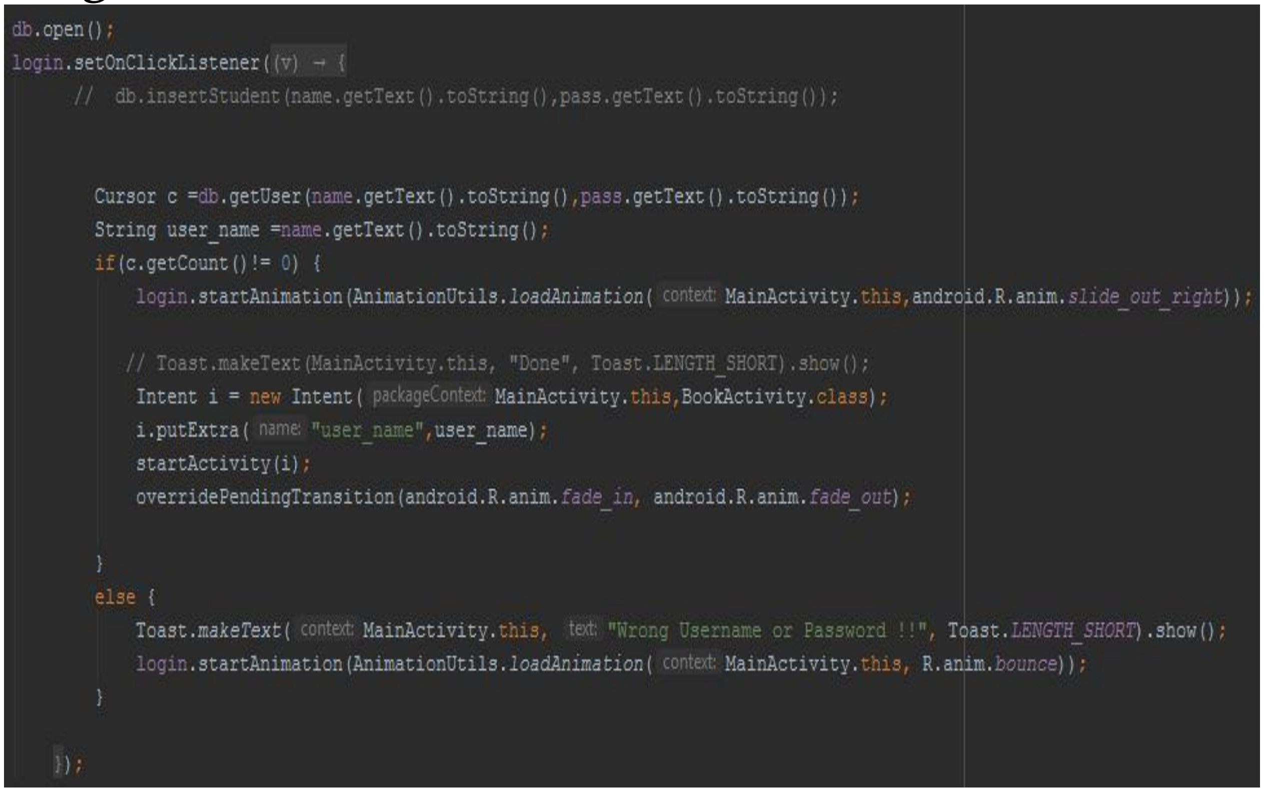
1 – Login Activity:

-Login Button

Using the database, it checks for the username the was given in the EditText and the password to take us to the books list activity.



-Login Button Code:

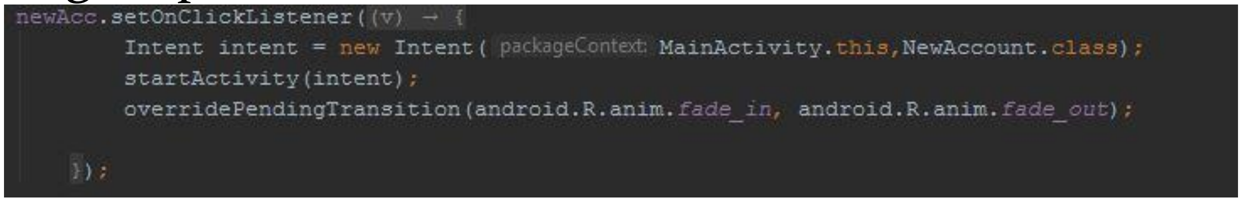


```
db.open();
login.setOnClickListener((v) → {
    //  db.insertStudent(name.getText().toString(),pass.getText().toString());


        Cursor c =db.getUser(name.getText().toString(),pass.getText().toString());
        String user_name =name.getText().toString();
        if(c.getCount()!= 0) {
            login.startAnimation(AnimationUtils.loadAnimation( context: MainActivity.this,android.R.anim.slide_out_right));

           // Toast.makeText(MainActivity.this, "Done", Toast.LENGTH_SHORT).show();
            Intent i = new Intent( packageContext: MainActivity.this,BookActivity.class);
            i.putExtra( name: "user_name",user_name);
            startActivity(i);
            overridePendingTransition(android.R.anim.fade_in, android.R.anim.fade_out);

        }
        else {
            Toast.makeText( context: MainActivity.this,  text: "Wrong Username or Password !!", Toast.LENGTH_SHORT).show();
            login.startAnimation(AnimationUtils.loadAnimation( context: MainActivity.this, R.anim.bounce));
        }

    });
```

First thing it open the database, then it gets the text from the EditText of name and password, if it were correct it return's true so it'll open the next activity passing the username to it, so we can use it later to save data on database.

If we enter wrong username or wrong password there will be a toast appearing in the screen telling us that it's wrong username or password.

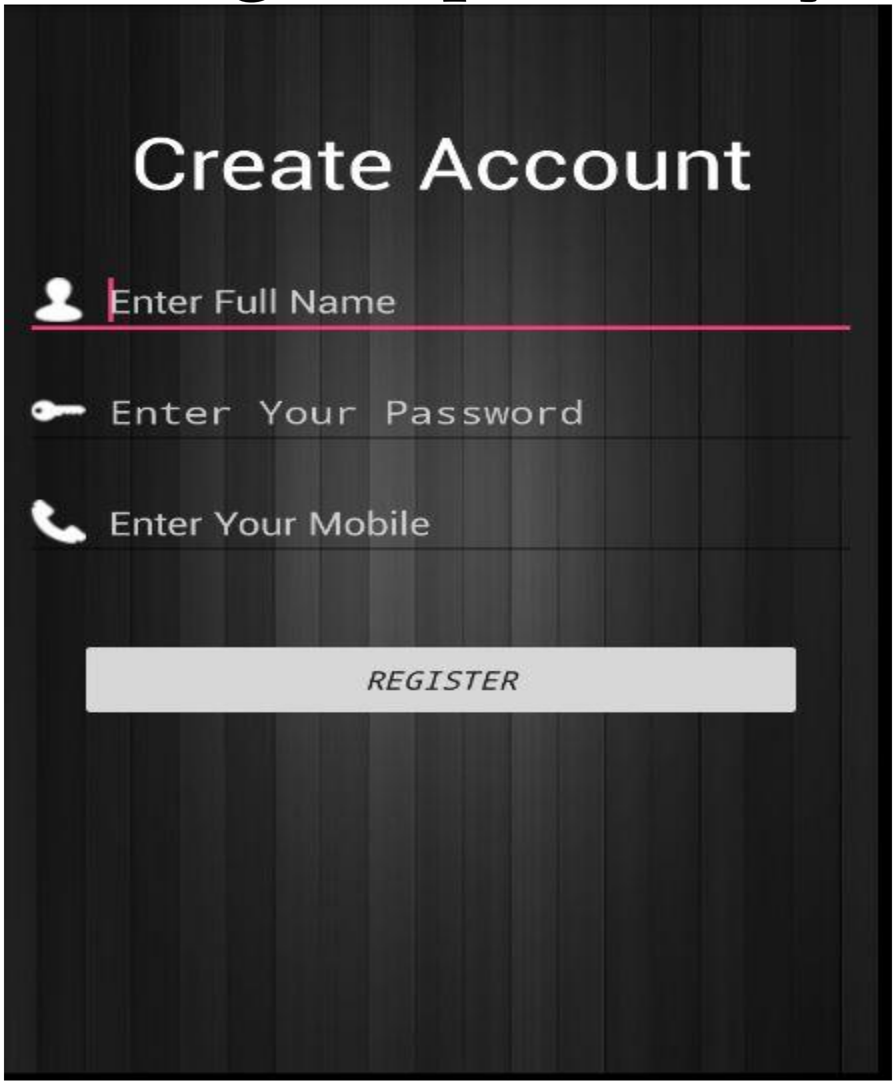
-Sign Up text:



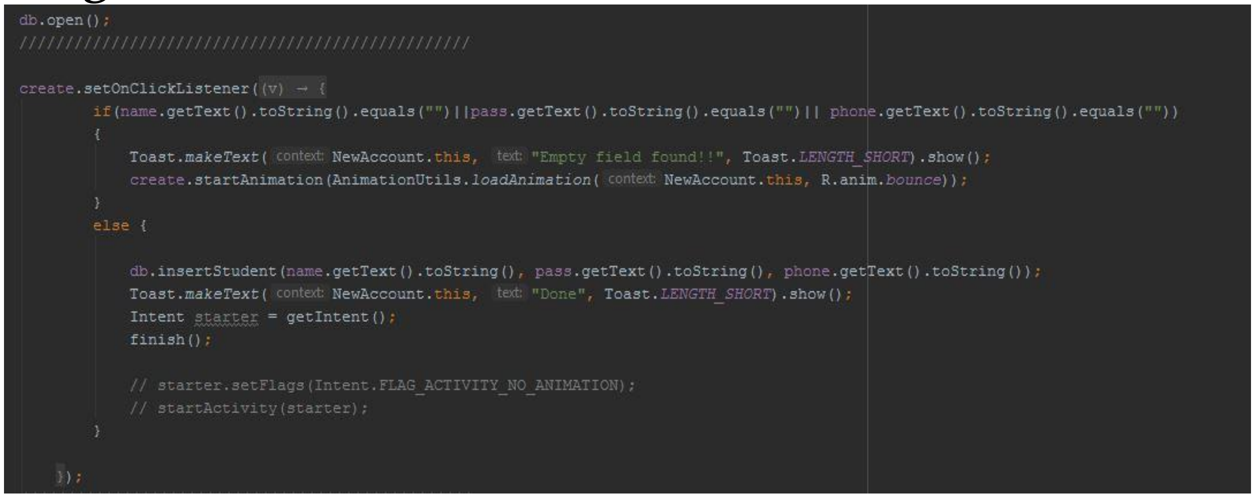
```
newAcc.setOnClickListener((v) → {
        Intent intent = new Intent( packageContext: MainActivity.this,NewAccount.class);
        startActivity(intent);
        overridePendingTransition(android.R.anim.fade_in, android.R.anim.fade_out);

    });
```

It moves the app to a sign-up screen where the user make a new account and save it to the app's database.

1 – Sign-Up Activity:



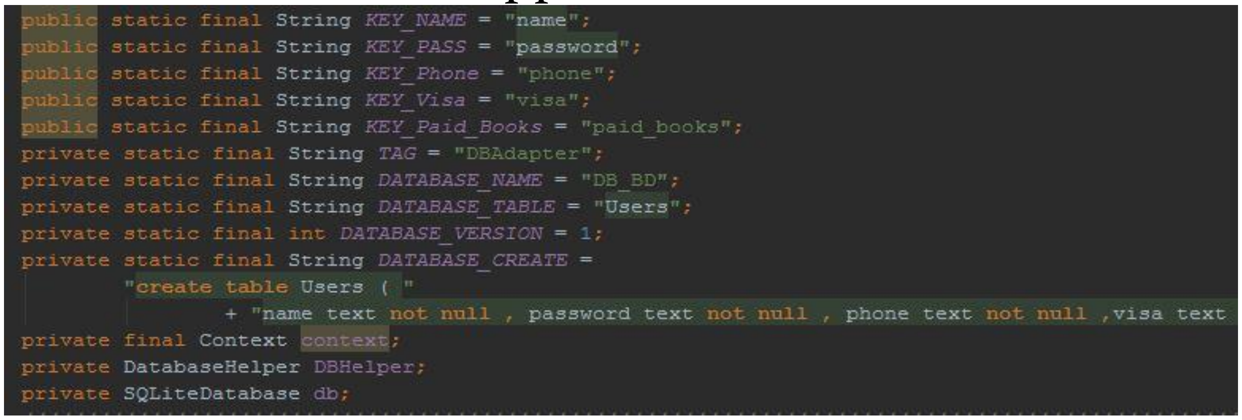
-Register Button:



```java
db.open();
/////////////////////////////////////////////////

create.setOnClickListener((v) → {
        if(name.getText().toString().equals("")||pass.getText().toString().equals("")|| phone.getText().toString().equals(""))
        {
            Toast.makeText( context: NewAccount.this,  text: "Empty field found!!", Toast.LENGTH_SHORT).show();
            create.startAnimation(AnimationUtils.loadAnimation( context: NewAccount.this, R.anim.bounce));
        }
        else {

            db.insertStudent(name.getText().toString(), pass.getText().toString(), phone.getText().toString());
            Toast.makeText( context: NewAccount.this,  text: "Done", Toast.LENGTH_SHORT).show();
            Intent starter = getIntent();
            finish();

            // starter.setFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
            // startActivity(starter);
        }

    });
```

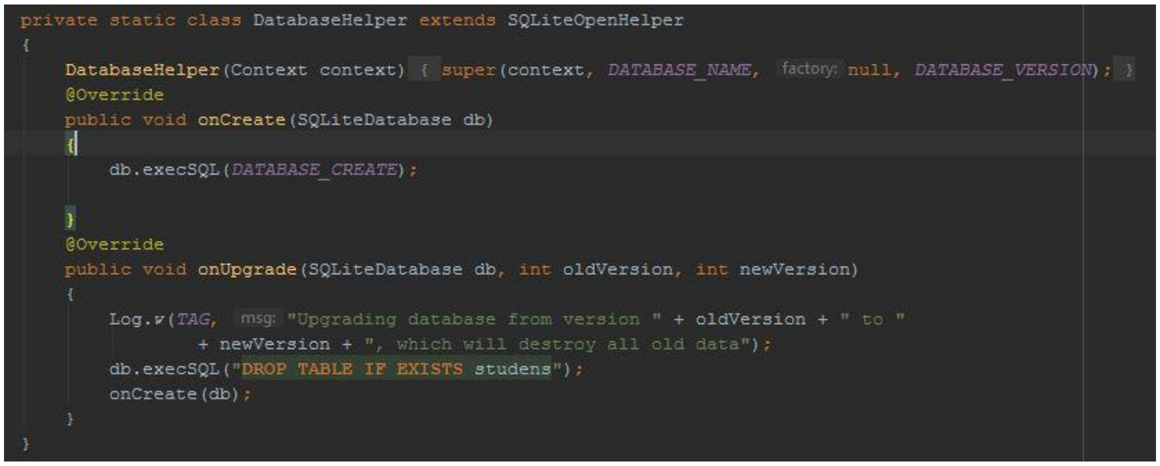
First, we open the database, so we can access it, then when we click on Register button with any of the fields is empty it'll show a toast with it, else it'll take the text from the fields and use the insert method from the database's adapter to insert it into the database, then close this activity and send us back to the Login Activity.

(**) Database Adapter Class Code:
To use the database, we had to create an adapter for it to connect the app with it.



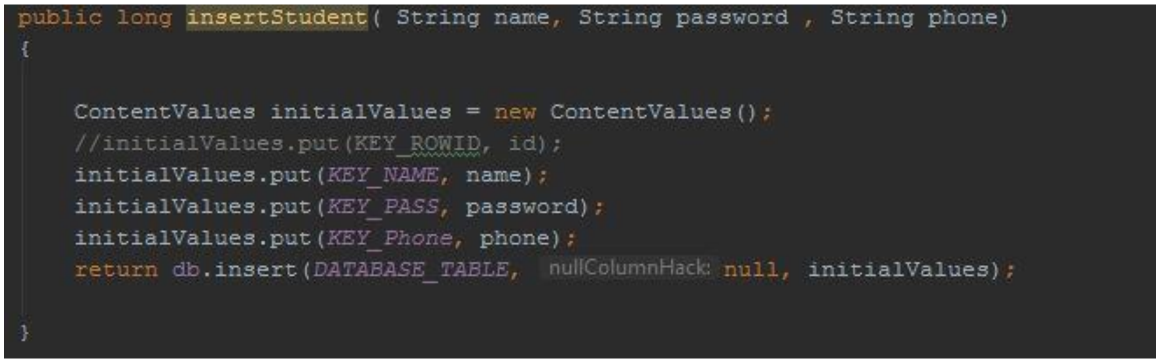
```java
public static final String KEY_NAME = "name";
public static final String KEY_PASS = "password";
public static final String KEY_Phone = "phone";
public static final String KEY_Visa = "visa";
public static final String KEY_Paid_Books = "paid_books";
private static final String TAG = "DBAdapter";
private static final String DATABASE_NAME = "DB_BD";
private static final String DATABASE_TABLE = "Users";
private static final int DATABASE_VERSION = 1;
private static final String DATABASE_CREATE =
        "create table Users ( "
                + "name text not null , password text not null , phone text not null ,visa text
private final Context context;
private DatabaseHelper DBHelper;
private SQLiteDatabase db;
```

First, we create variables that has the names of the Database's columns, name and a create query so we can use it directly.



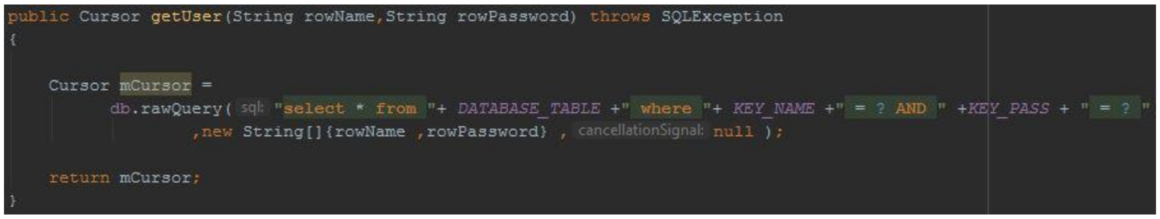
```
private static class DatabaseHelper extends SQLiteOpenHelper
{
    DatabaseHelper(Context context) { super(context, DATABASE_NAME, factory: null, DATABASE_VERSION); }
    @Override
    public void onCreate(SQLiteDatabase db)
    {
        db.execSQL(DATABASE_CREATE);

    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
    {
        Log.w(TAG, msg: "Upgrading database from version " + oldVersion + " to "
                + newVersion + ", which will destroy all old data");
        db.execSQL("DROP TABLE IF EXISTS studens");
        onCreate(db);
    }
}
```

We also have a class the extends the SQLiteOpenHelper that is used to perform a create query and upgrade too.



```
public long insertStudent( String name, String password , String phone)
{


    ContentValues initialValues = new ContentValues();
    //initialValues.put(KEY_ROWID, id);
    initialValues.put(KEY_NAME, name);
    initialValues.put(KEY_PASS, password);
    initialValues.put(KEY_Phone, phone);
    return db.insert(DATABASE_TABLE, nullColumnHack: null, initialValues);

}
```

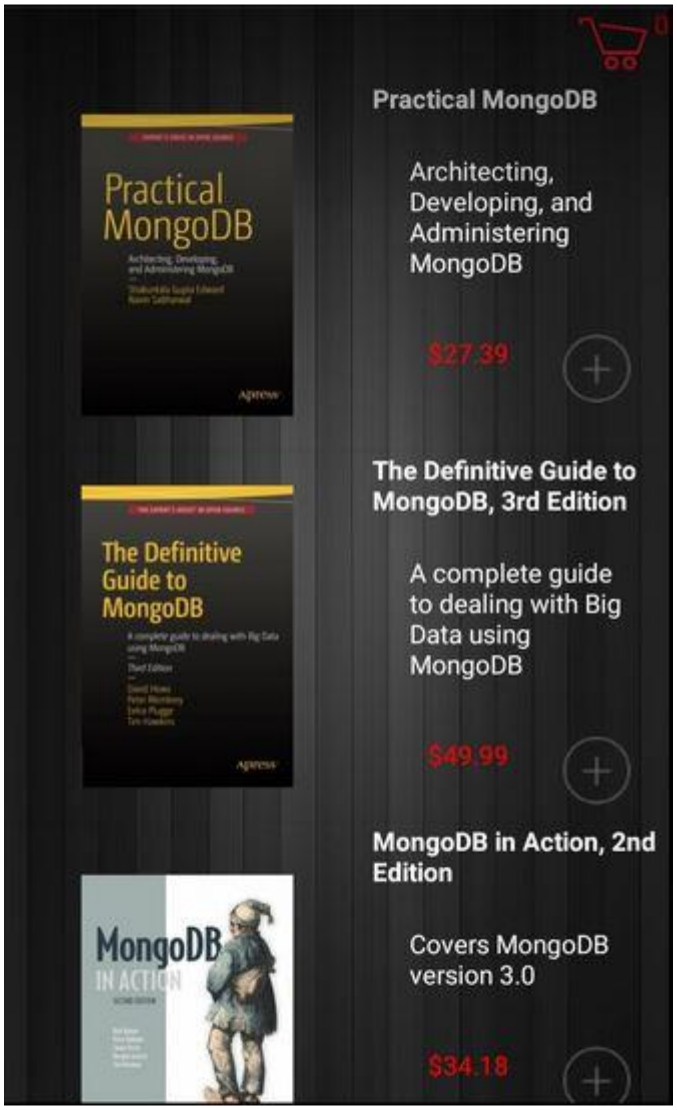
There's an insert method that takes the parameters from the user and save it to the database's columns.



```
public Cursor getUser(String rowName,String rowPassword) throws SQLException
{

    Cursor mCursor =
            db.rawQuery( sql: "select * from "+ DATABASE_TABLE +" where "+ KEY_NAME +" = ? AND " +KEY_PASS + " = ? "
                    ,new String[]{rowName ,rowPassword} , cancellationSignal: null );

    return mCursor;
}
```

At last we have a select method that we use to check for the username and the password for login.

3 – Book List Activity:



Practical MongoDB

Architecting, Developing, and Administering MongoDB

$27.39

The Definitive Guide to MongoDB, 3rd Edition

A complete guide to dealing with Big Data using MongoDB

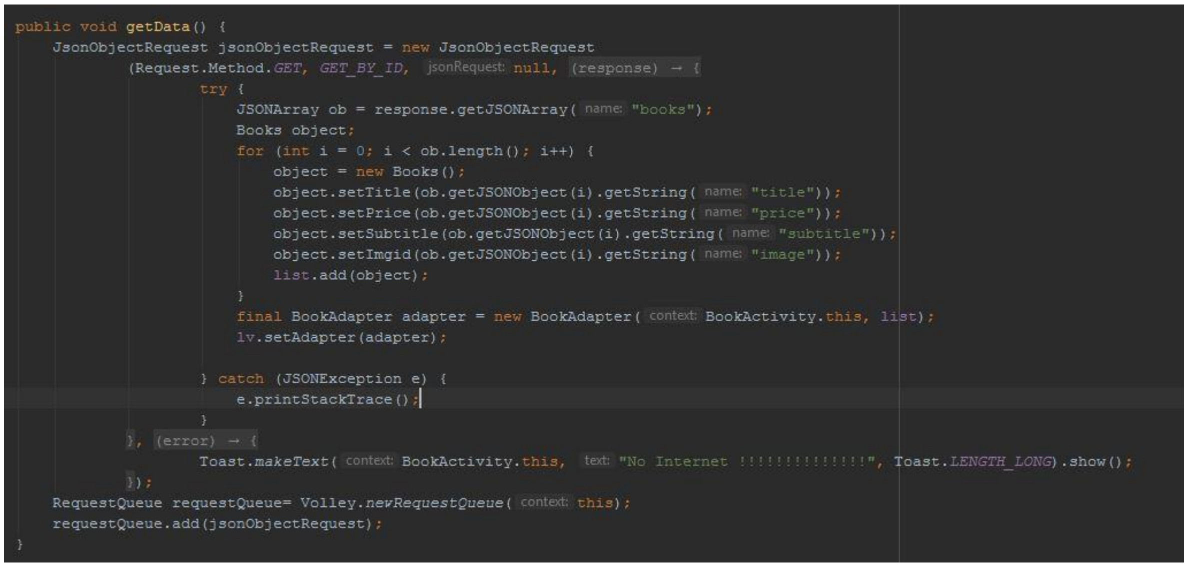
$49.99

MongoDB in Action, 2nd Edition

Covers MongoDB version 3.0

$34.18

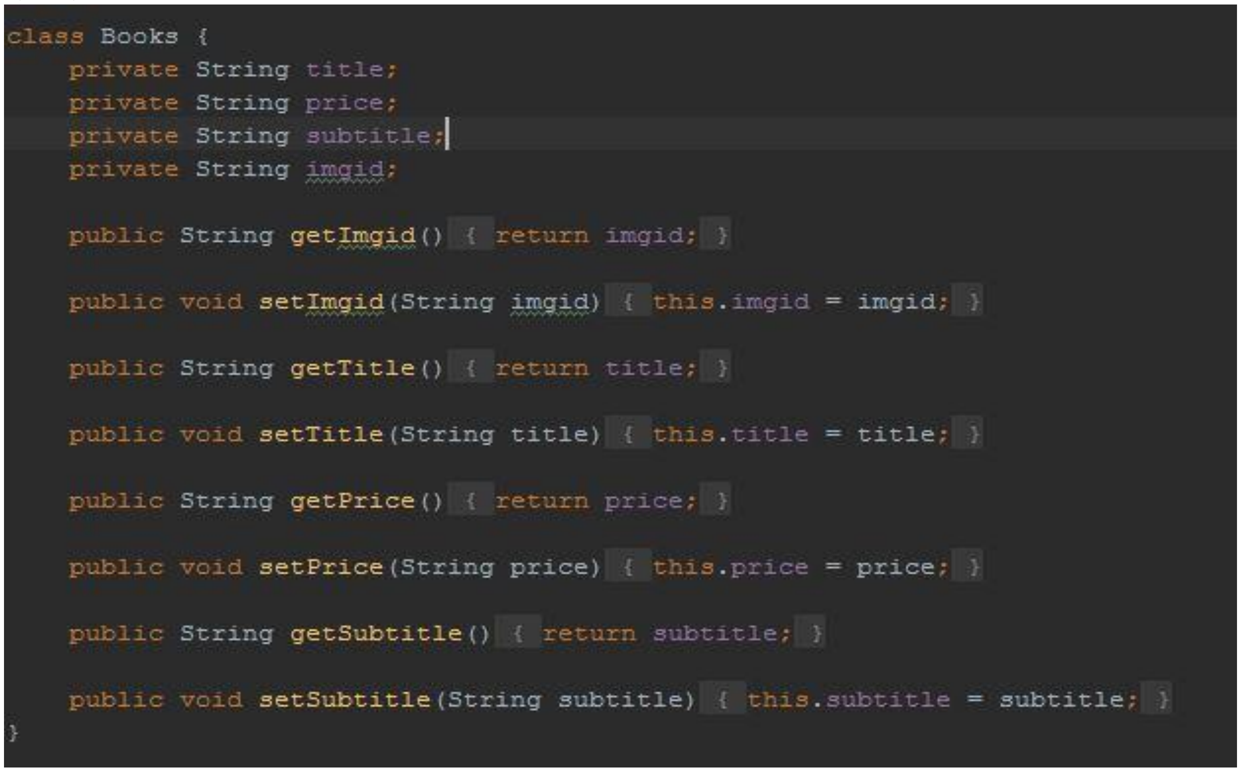
-On this activity first thing onCreate it'll load (GetData () ;) method the brings the information from the following URL:
" https://api.itbook.store/1.0/search/mongodb"



```
public void getData() {
    JsonObjectRequest jsonObjectRequest = new JsonObjectRequest
            (Request.Method.GET, GET_BY_ID,  jsonRequest: null, (response) → {
                try {
                    JSONArray ob = response.getJSONArray( name: "books");
                    Books object;
                    for (int i = 0; i < ob.length(); i++) {
                        object = new Books();
                        object.setTitle(ob.getJSONObject(i).getString( name: "title"));
                        object.setPrice(ob.getJSONObject(i).getString( name: "price"));
                        object.setSubtitle(ob.getJSONObject(i).getString( name: "subtitle"));
                        object.setImgid(ob.getJSONObject(i).getString( name: "image"));
                        list.add(object);
                    }
                    final BookAdapter adapter = new BookAdapter( context: BookActivity.this, list);
                    lv.setAdapter(adapter);

                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }, (error) → {
                Toast.makeText( context: BookActivity.this,  text: "No Internet !!!!!!!!!!!!!!", Toast.LENGTH_LONG).show();
            });
    RequestQueue requestQueue= Volley.newRequestQueue( context: this);
    requestQueue.add(jsonObjectRequest);
}
```

We first make a jsonObject request that requests the server for the information, then we need the "books" Json array from the URL so we create new JsonArray and add the 4 elements (title, price, sub_title, image) we need from the URL, we get it by creating a new object of a class That we created named as "Books"-see image below for code- it set and get these values, add the elements to the Books object then add it to an arrayList object named as List, then we create an object of the adapter that will attach the arrayList to the ListView on the layout, then create a ListView object and set its adapter to the BookAdapter class' object we created.

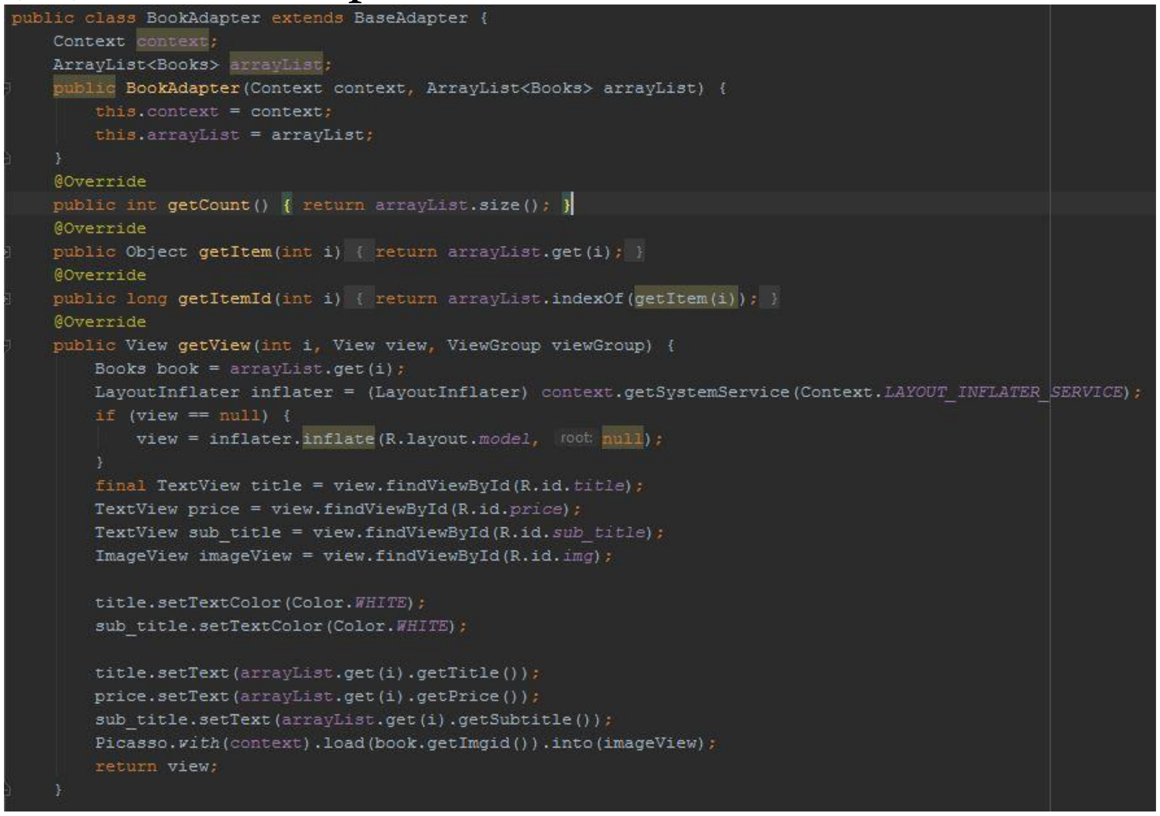
(**) Books Class Code:



```
class Books {
    private String title;
    private String price;
    private String subtitle;
    private String imgid;

    public String getImgid() { return imgid; }

    public void setImgid(String imgid) { this.imgid = imgid; }

    public String getTitle() { return title; }

    public void setTitle(String title) { this.title = title; }

    public String getPrice() { return price; }

    public void setPrice(String price) { this.price = price; }

    public String getSubtitle() { return subtitle; }

    public void setSubtitle(String subtitle) { this.subtitle = subtitle; }
}
```

As we said above it set and gets the 4 elements we need from the URL of the JsonObject.

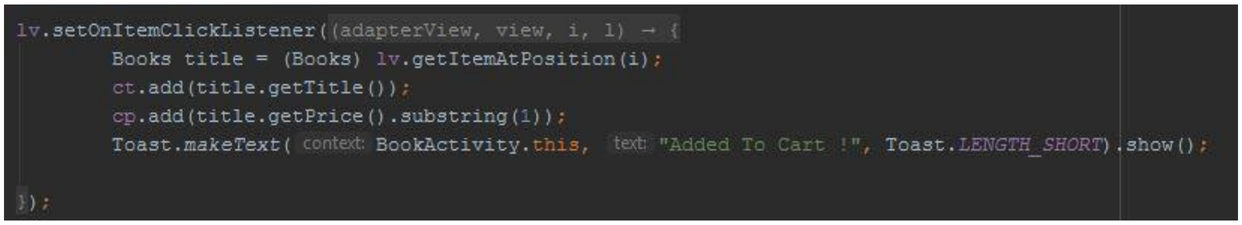
(**) Books Adapter Class Code:



```
public class BookAdapter extends BaseAdapter {
    Context context;
    ArrayList<Books> arrayList;
    public BookAdapter(Context context, ArrayList<Books> arrayList) {
        this.context = context;
        this.arrayList = arrayList;
    }
    @Override
    public int getCount() { return arrayList.size(); }
    @Override
    public Object getItem(int i) { return arrayList.get(i); }
    @Override
    public long getItemId(int i) { return arrayList.indexOf(getItem(i)); }
    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        Books book = arrayList.get(i);
        LayoutInflater inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        if (view == null) {
            view = inflater.inflate(R.layout.model,  root: null);
        }
        final TextView title = view.findViewById(R.id.title);
        TextView price = view.findViewById(R.id.price);
        TextView sub_title = view.findViewById(R.id.sub_title);
        ImageView imageView = view.findViewById(R.id.img);

        title.setTextColor(Color.WHITE);
        sub_title.setTextColor(Color.WHITE);

        title.setText(arrayList.get(i).getTitle());
        price.setText(arrayList.get(i).getPrice());
        sub_title.setText(arrayList.get(i).getSubtitle());
        Picasso.with(context).load(book.getImgid()).into(imageView);
        return view;
    }
```

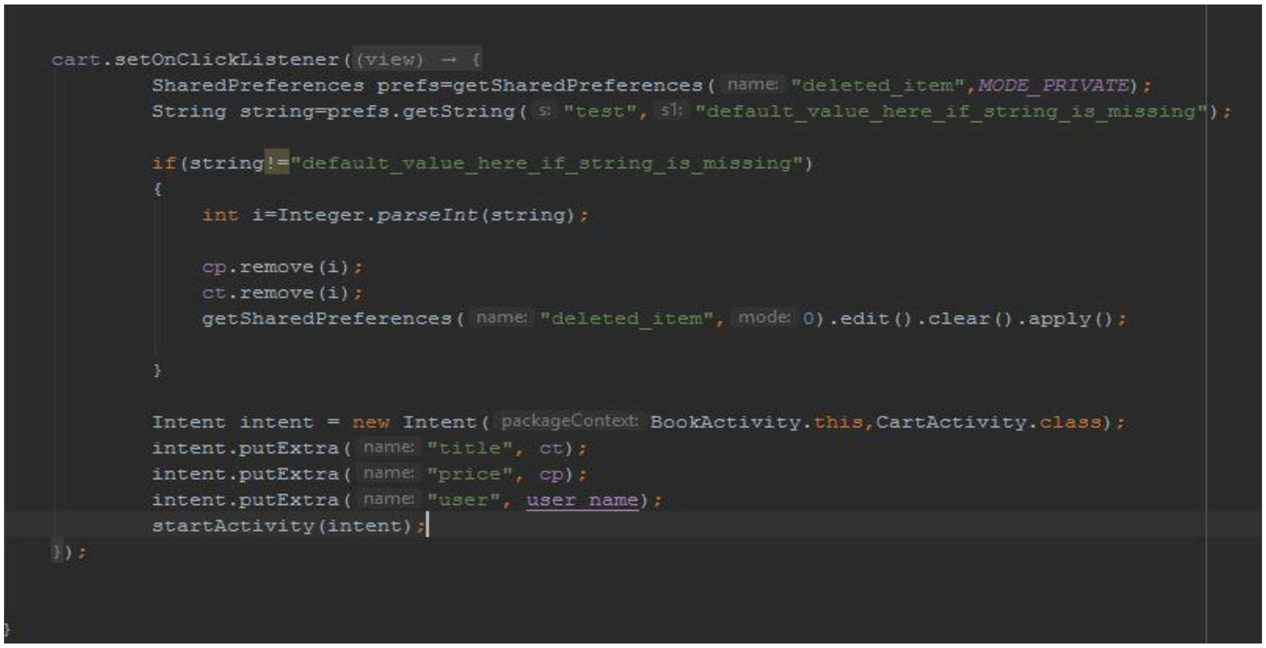
We create a new class and make it extend “BaseAdapter” so it adds Override methods such as “getItem” that return the item at given index, So, to put items in ListView we override “getView” and inside it we get the first object from the Books arrayList then declare each component of the list we need to pass a value to it then give it the value and return the view, then it’ll create the ListView on the layout and put the items on it.

-ListView item Click:


```
lv.setOnItemClickListener((adapterView, view, i, l) → {
        Books title = (Books) lv.getItemAtPosition(i);
        ct.add(title.getTitle());
        cp.add(title.getPrice().substring(1));
        Toast.makeText( context: BookActivity.this,  text: "Added To Cart !", Toast.LENGTH_SHORT).show();

});
```

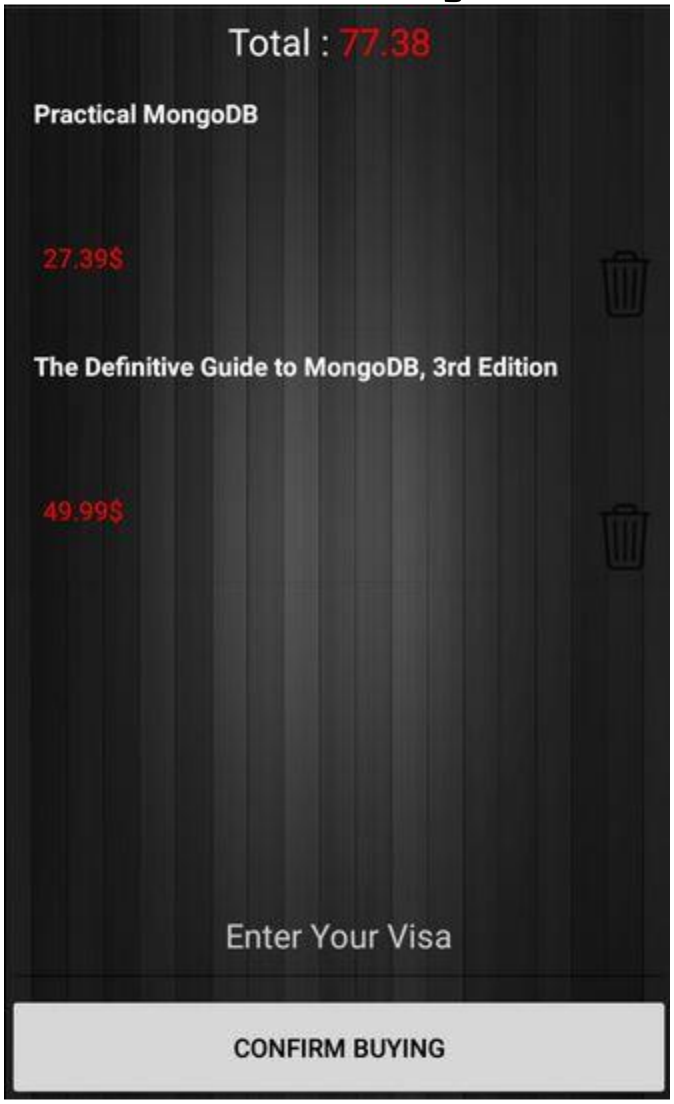
On the BookList Activity when we click on any item it'll be added to the Cart to confirm the buying process on it.

-On the top of the Activity there's "Add to Cart" button:


```
    cart.setOnClickListener((view) → {
            SharedPreferences prefs=getSharedPreferences( name: "deleted_item",MODE_PRIVATE);
            String string=prefs.getString( s: "test",  s1: "default_value_here_if_string_is_missing");

            if(string!="default_value_here_if_string_is_missing")
            {
                int i=Integer.parseInt(string);

                cp.remove(i);
                ct.remove(i);
                getSharedPreferences( name: "deleted_item",  mode: 0).edit().clear().apply();

            }

            Intent intent = new Intent( packageContext: BookActivity.this,CartActivity.class);
            intent.putExtra( name: "title", ct);
            intent.putExtra( name: "price", cp);
            intent.putExtra( name: "user", user_name);
            startActivity(intent);
    });


}
```

It takes the title and price of the selected book and move it to the cart.

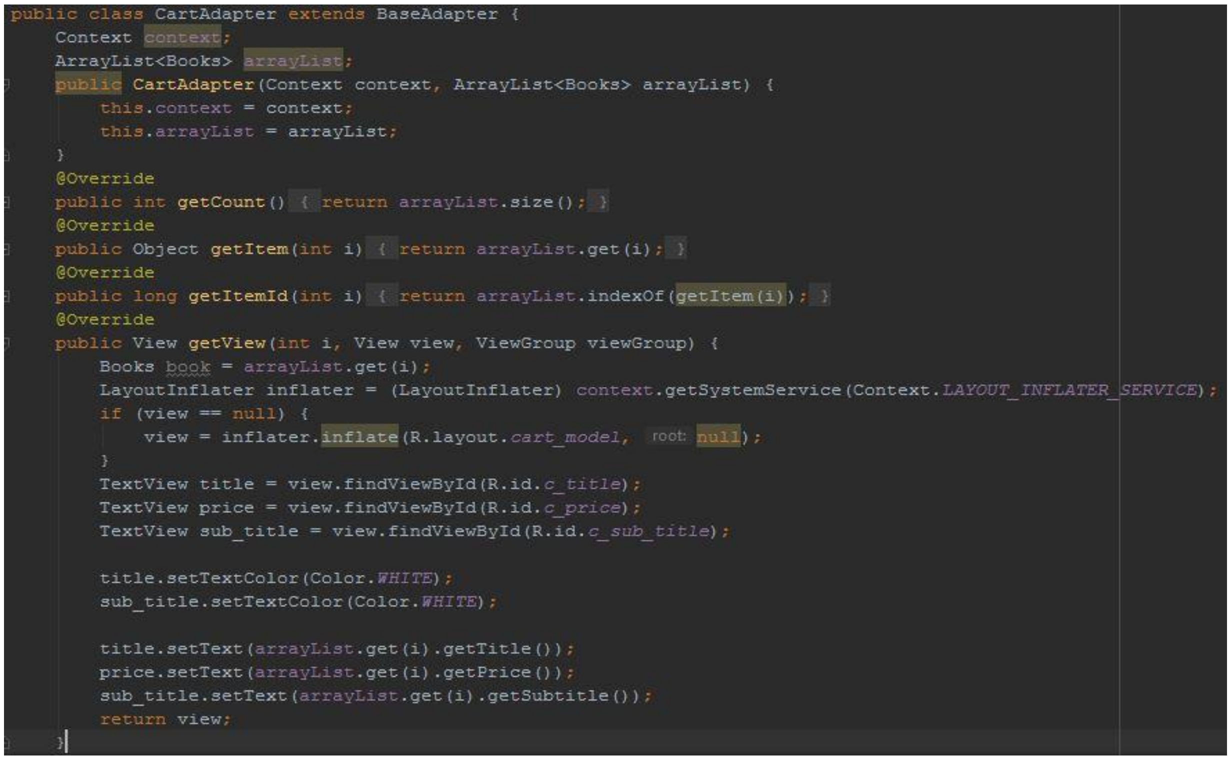
4 – Cart & Confirmation Activity:



In this Activity we have a ListView with a button and EditText.

To make the items appear on the ListView we made a new adapter like the BookAdapter
But it only returns the view with title and price only, and we used the Books class too with the adapter.

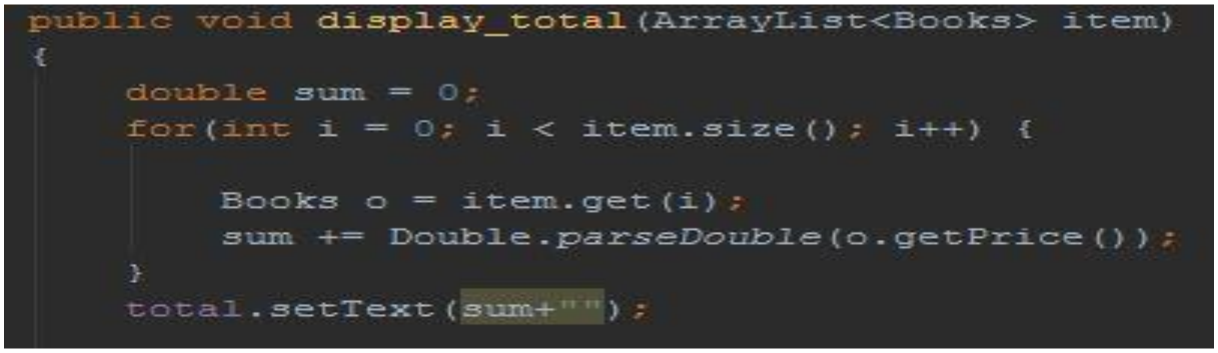
-Cart Adapter Code:



```
public class CartAdapter extends BaseAdapter {
    Context context;
    ArrayList<Books> arrayList;
    public CartAdapter(Context context, ArrayList<Books> arrayList) {
        this.context = context;
        this.arrayList = arrayList;
    }
    @Override
    public int getCount() { return arrayList.size(); }
    @Override
    public Object getItem(int i) { return arrayList.get(i); }
    @Override
    public long getItemId(int i) { return arrayList.indexOf(getItem(i)); }
    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        Books book = arrayList.get(i);
        LayoutInflater inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        if (view == null) {
            view = inflater.inflate(R.layout.cart_model,  root: null);
        }
        TextView title = view.findViewById(R.id.c_title);
        TextView price = view.findViewById(R.id.c_price);
        TextView sub_title = view.findViewById(R.id.c_sub_title);

        title.setTextColor(Color.WHITE);
        sub_title.setTextColor(Color.WHITE);

        title.setText(arrayList.get(i).getTitle());
        price.setText(arrayList.get(i).getPrice());
        sub_title.setText(arrayList.get(i).getSubtitle());
        return view;
    }
```

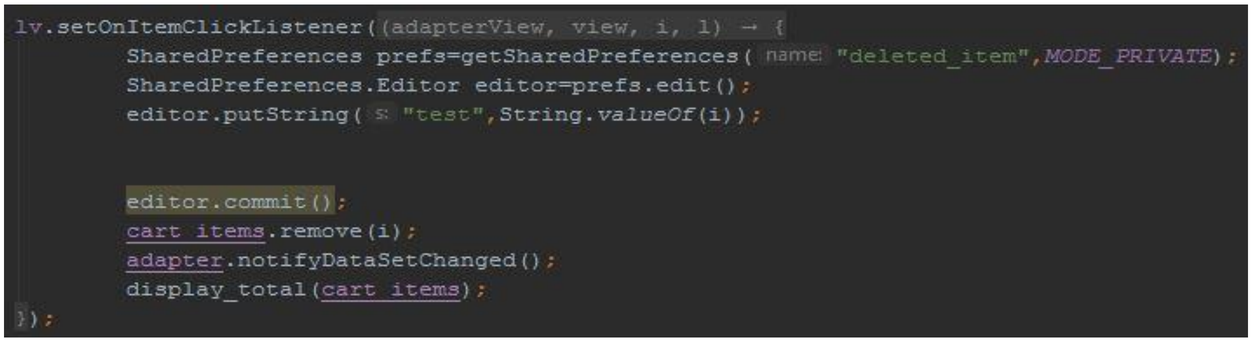
Everything we did in the BookAdapter class we did it here too, but we removed the images and sub-title from it, so it'll make a view with title and price only.

In this activity we used a method that takes the price of every existed book from the arraylist passed from the previous activity and get its total price and shows it on the top of the activity:



```
public void display_total(ArrayList<Books> item)
{
    double sum = 0;
    for(int i = 0; i < item.size(); i++) {

        Books o = item.get(i);
        sum += Double.parseDouble(o.getPrice());
    }
    total.setText(sum+"");
```

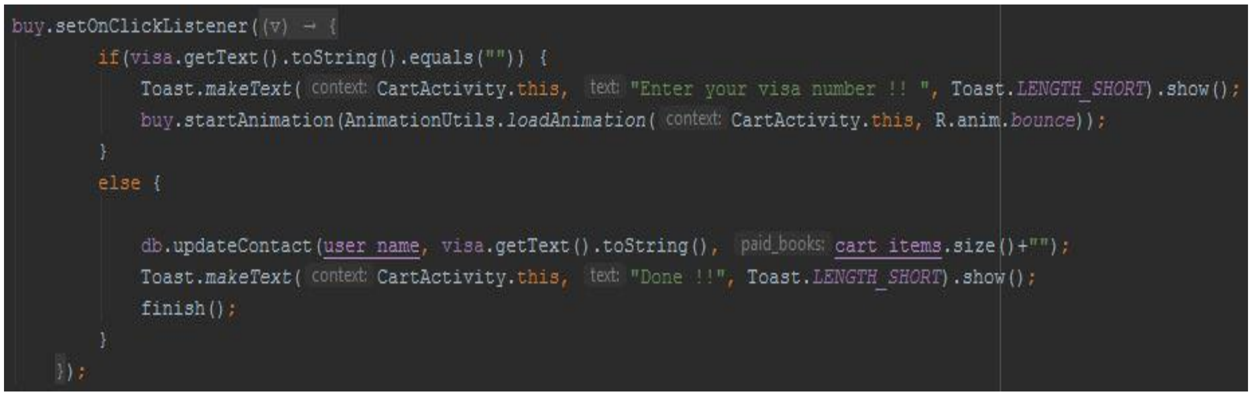
In this activity If the user added a book and before buying he doesn't want it he can easily click on it to remove it from the cart:



```
lv.setOnItemClickListener((adapterView, view, i, l) → {
        SharedPreferences prefs=getSharedPreferences( name: "deleted_item",MODE_PRIVATE);
        SharedPreferences.Editor editor=prefs.edit();
        editor.putString( s: "test",String.valueOf(i));


        editor.commit();
        cart_items.remove(i);
        adapter.notifyDataSetChanged();
        display_total(cart_items);
});
```

This code removes the selected item from the list and refreshes it.

-CartActivity Confirm Buying Button:



```
buy.setOnClickListener((v) → {
        if(visa.getText().toString().equals("")) {
            Toast.makeText( context: CartActivity.this,  text: "Enter your visa number !! ", Toast.LENGTH_SHORT).show();
            buy.startAnimation(AnimationUtils.loadAnimation( context: CartActivity.this, R.anim.bounce));
        }
        else {

            db.updateContact(user_name, visa.getText().toString(),  paid_books: cart_items.size()+"");
            Toast.makeText( context: CartActivity.this,  text: "Done !!", Toast.LENGTH_SHORT).show();
            finish();
        }
    });
```

If we press the button with a blank visa EditText it'll give us a toast with the error, else it'll take the username of the user, his visa number and the number of books he bought and save it to the database.