Even Semester (2023)



**BINUS UNIVERSITY**

**BINUS INTERNATIONAL**

**Assignment Cover Letter**

**(Individual Work)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | |  | |  | |  |
| **Student Information**: **Surname** | | | | | | **Given Names**    **Jocelyn** | | **Student ID Number**  **2301900454** | |  |
| 1. | | | **Thiojaya** |  | |  |
|  |  | |  |
| **Course Code** | **: COMP6510** | |  |  | | **Course Name** | | **: Introduction to Programming** | |  |
| **Class** | **: L2AC** | |  |  | | **Name of Lecturer(s)** | | **: Jude Joseph Lamug Martinez** | |  |
|  |  | |  |  | |  | |  | |  |
| **Major** | **: Comp. Science** | |  |  | |  | |  | |  |
| **Title of Assignment**  (if any) | : Inventory Management App | | |  |  | |  | |  | |
| **Type of Assignment**    **Submission Pattern** | **: Final Project** |  | |  | |  | |  | |  |
| **Due Date** | **: 20-06-20** |  | |  | | **Submission Date** | | **: 17-01-20** | |  |

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer’s instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

# Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

# Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)

1. Jocelyn Thiojaya

**“Inventory Management App”**

**Name : Jocelyn Thiojaya**

**ID : 2301900454**

1. **PROJECT SPECIFICATIONS**
2. **Introduction**

Online shops are becoming more popular recently, with increasing amounts of interest in shopping online. The types of online shops may vary, but they all need the same basic things, such as a platform for the online shop to run in, and an inventory system.

A problem that is often encountered when running an online shop is miscommunication, or deviation of item stock numbers. This happens due to having more than one person manage the shop and send the items. Online shops usually have a database, but one that must be accessed through computer. Updating stock and changing information also requires the user to log into the database, making it quite a time-consuming process. This opens room for mistakes. One person might have taken the items to be sent for today and forget to update the stock. A second person might look at the stock and think that today’s items have not been processed, then take out the items again. This gets worse when there is more than one storehouse, and they fail to communicate with each other.

Regardless, it is impossible for an online shop to consist of only a single person, because of the amount of work to be done. This project’s idea is to make a more accessible way of managing stock, orders, and items by making a mobile application. Nearly everyone always brings their phone with them, so they could make changes to a shop inventory easily.

This project will focus on building a simple inventory management app, aiming to help small scale online shops to keep track of their inventory. The code will focus on making the basic interface and features to manage item and order data. For now, the data will be stored locally for this app.

1. **Features**

This inventory management app will consist of a main menu, where users can choose to either view their inventory or manage customer orders. They will see a list of their current items or orders and have the choice to add or remove from their list. For items, they will have the option to edit the stock count of each item. For orders, they may add a list of items linked to each order.

1. **Android Studio**

This project will be made using Android Studio. Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA.

Project Structure in Android Studio contains one or more modules with source code files and resource files. Types of modules include android app modules, library modules, and Google App Engine modules.

Each app module contains the following folders: manifests: Contains the AndroidManifest.xml file. java: Contains the Java source code files, including JUnit test code. res: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

This project will use of multiple activities connected to each other. Each activity is an XML layout file, which is connected to a java file that controls that activity. Additional layout resource files and java class files will be used to assist the activities.

1. **Recycler View in Android Studio**

In the RecyclerView model, several different components work together to display data. The overall container for user interface is a RecyclerView object that you add to your layout. The RecyclerView fills itself with views provided by a layout manager that you provide. You can use one of our standard layout managers (such as LinearLayoutManager or GridLayoutManager), or implement your own.

The views in the list are represented by view holder objects. These objects are instances of a class you define by extending RecyclerView.ViewHolder. Each view holder is in charge of displaying a single item with a view. For example, if your list shows music collection, each view holder might represent a single album. The RecyclerView creates only as many view holders as are needed to display the on-screen portion of the dynamic content, plus a few extra. As the user scrolls through the list, the RecyclerView takes the off-screen views and rebinds them to the data which is scrolling onto the screen.

The view holder objects are managed by an adapter, which you create by extending RecyclerView.Adapter. The adapter creates view holders as needed. The adapter also binds the view holders to their data. It does this by assigning the view holder to a position, and calling the adapter's onBindViewHolder() method. That method uses the view holder's position to determine what the contents should be, based on its list position.

1. **Card View in Android Studio**

Apps often need to display data in similarly styled containers. These containers are often used in lists to hold each item's information. The system provides the CardView API as an easy way for you show information inside cards that have a consistent look across the platform. These cards have a default elevation above their containing view group, so the system draws shadows below them. Cards provide an easy way to contain a group of views while providing a consistent style for the container.

1. **Shared Preferences in Android Studio**

If you have a relatively small collection of key-values that you'd like to save, you should use the SharedPreferences APIs. A SharedPreferences object points to a file containing key-value pairs and provides simple methods to read and write them. Each SharedPreferences file is managed by the framework and can be private or shared.

You can create a new shared preference file or access an existing one by calling these method:

getSharedPreferences() — Use this if you need multiple shared preference files identified by name, which you specify with the first parameter. You can call this from any Context in your app.

To write to a shared preferences file, create a SharedPreferences.Editor by calling edit() on your SharedPreferences. Pass the keys and values you want to write with methods such as putInt() and putString(). Then call apply() or commit() to save the changes.

1. **SOLUTION DESIGN**
2. **A screenshot of a social media post

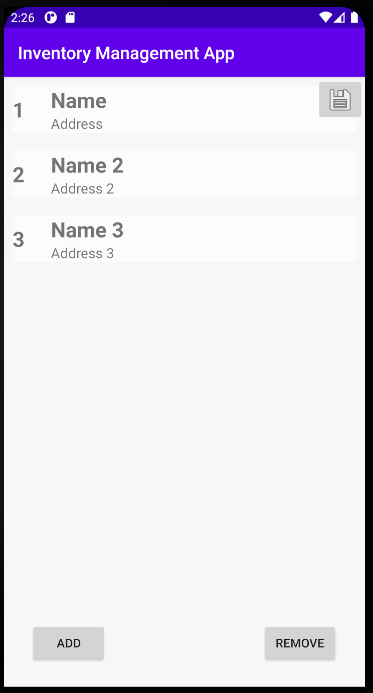
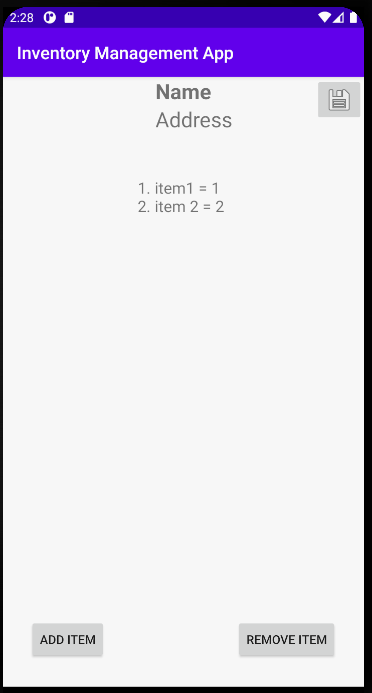
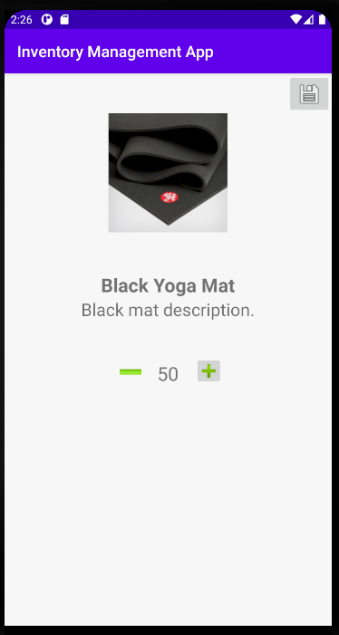
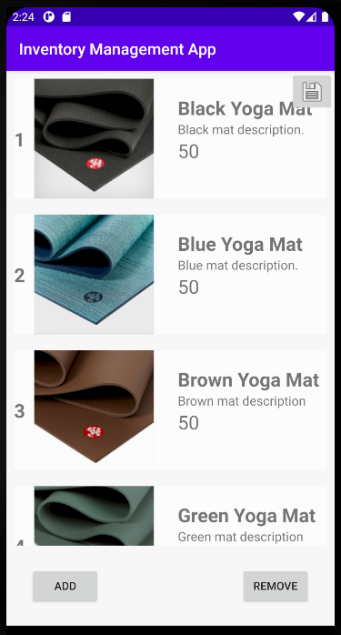
   Description automatically generatedUML Class Diagram**

Note: the diagram might appear blurry but zooming in will make it clear.

1. **Basic Application Flow and Mockup**

This Inventory Management application will consist of a few activities (or screens), which are Homepage, Item list page, Item list details, Order list page, Order list details. Homepage will only have 2 buttons, one to take the user to the item list page, and one to take the user to the order list page. Order list and Item list pages is a RecyclerView, and each row is a CardView. Clicking a row will take the user to the corresponding details page.

The following is the app mockup:



1. **Item class and Order class**

Each product in this inventory is an instance of the Item class. The item class has 4 attributes, *imageResource* and *stock* which are integers, *name* and *description* which are strings. There is one constructor for this class that requires the input of all 4 attributes. The methods in Item class are all setters and getters for each attribute. This will come in useful for manipulating an item’s attribute later in our code.

Each order in this inventory is an instance of the Order class. The order class has 3 attributes, *name* and *description* which are strings, and *items* which is an arraylist of Items. So, each instance of Order will be able to have an arraylist of Items, representing the items that a customer had ordered. There is one constructor for this class that requires the input of *name* and *description*, the *items* arraylist will be made default as a new empty arraylist. The methods in this class are setters and getters for *name* and *description*, additionally there are a getter, addItem(), and removeItem() methods for the *items* arraylist,

1. **Code for the RecyclerViews**

To make a RecyclerView, we need a total of two java classes, one is main file that is connected to the activity with the RecyclerView, and one is the Adapter file that manages the data coming in and out of the RecyclerView.

There are two activities with RecyclerViews in this project, the first one for Inventory (filled with a list of Items) and one for Orders (filled with a list of Orders). The main file for the Inventory RecyclerView is called ViewInventory.java, and the adapter is called MyAdapter.java. The main file for the Order RecyclerView is called ManageOrders.java, and the adapter is called MyAdapterOrders.java.

The adapter classes are going to be managing the data displayed on its corresponding RecyclerView. The adapters are responsible for a few things:

* Detecting an item click, managing information provided by that item click.

//below code is in ViewHolder class

itemView.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (listener != null){  
 int position = getAdapterPosition();  
 if (position != RecyclerView.*NO\_POSITION*){  
 listener.onItemClick(position);

* Making a View Holder, which connects items to their layout ID.

public static class MyViewHolder extends RecyclerView.ViewHolder{  
 public ImageView image;  
 public TextView name, description, position, stock;  
  
 public MyViewHolder(@NonNull View itemView, final OnItemClickListener listener) {  
 super(itemView);  
 image = itemView.findViewById(R.id.*imageView*);  
 name = itemView.findViewById(R.id.*textViewTitle*);  
 description = itemView.findViewById(R.id.*textViewDescription*);  
 position = itemView.findViewById(R.id.*textViewPosition*);  
 stock = itemView.findViewById(R.id.*textViewStock*);

* Connecting the CardView to each row of RecyclerView (OnCreateViewHolder method)
* Setting the correct text for each item (OnBindViewHolder method).

The main classes have one method to build the RecyclerView. This method is responsible for a few things:

* Connecting the instance of RecyclerView in this class to the layout ID.
* Giving the adapter all the information it needs (which includes itemList, an arraylist of Items)
* Making an onItemClickListener for clicks on the rows, and determining what clicking a row does.

private void buildRecyclerView(){  
 myRecyclerView = findViewById(R.id.*myRecyclerView*);  
 myRecyclerView.setHasFixedSize(true);  
 myLayoutManager = new LinearLayoutManager(this);  
 myAdapter = new MyAdapter(itemList);  
  
 myRecyclerView.setLayoutManager(myLayoutManager);  
 myRecyclerView.setAdapter(myAdapter);  
  
 myAdapter.setOnItemClickListener(new MyAdapter.OnItemClickListener() {  
 @Override  
 public void onItemClick(int position) {  
 Intent startIntent = new Intent(getApplicationContext(), InventoryDetails.class);  
 startIntent.putExtra("position", position);  
 startActivity(startIntent);  
 }  
 });

1. **Data for the Item/Order lists**

All the data management are in the ViewInventory.java and ManageOrders.java files. This includes loading and saving data to a file, adding and removing data to the respective arraylists. Since the code is the same for both files, the only difference being ViewInventory manages data for itemList, and ManageOrders manages data for orderList, I am only going to explain one for the itemList.

First, we need to initialize an arraylist in this class. Because this class is for the Items list, we make an arraylist called itemList which contains instances of Item.

private ArrayList<Item> itemList;

Now we need two methods to save the data and load the data. We will be using SharedPreferences to do this. As you can see, saveData() takes the current itemList and writes it into and saves it with the key “item list”. The loadData() method gets from sharedPreferences and searched the information with the key “item list”, then it writes the information to itemList.

If the itemList taken from loadData() has no content, it will add the default data to itemList. The default data is a sequence of items which I added by code, in the createItemList() method.

private void saveData(){  
 SharedPreferences sharedPreferences = getSharedPreferences("shared preferences", *MODE\_PRIVATE*);  
 SharedPreferences.Editor editor = sharedPreferences.edit();  
 Gson gson = new Gson();  
 String json = gson.toJson(itemList);  
 editor.putString("item list", json);  
 editor.apply();  
}  
  
private void loadData(){  
 SharedPreferences sharedPreferences = getSharedPreferences("shared preferences", *MODE\_PRIVATE*);  
 Gson gson = new Gson();  
 String json = sharedPreferences.getString("item list", null);  
 Type type = new TypeToken<ArrayList<Item>>() {}.getType();  
 itemList = gson.fromJson(json, type);  
  
 if (itemList == null || itemList.size() < 1){  
 itemList = new ArrayList<>();  
 createItemList();  
 }  
}

There are two more important methods to discuss, which are insertItem() and removeItem().

insertItem() takes user input from two EditText fields (name and description) provided in the screen. From these EditTexts, this method creates a new instance of Item according to the user input and adds it to the itemList.

removeItem() takes user input which is the position of the item desired to be deleted. This method will remove the item in that position. Here, we remove item position (user input - 1) because positions in arraylists start from 0, but the user would see a list with position starting from 1.

private void insertItem(EditText text1, EditText text2){  
 String name = text1.getText().toString();  
 String description = text2.getText().toString();  
 itemList.add(new Item(R.drawable.*black*, name, description, 50));  
 myAdapter.notifyItemInserted(itemList.size() + 1);  
}  
  
private void removeItem(EditText text){  
 int position = Integer.*parseInt*(text.getText().toString());  
 itemList.remove(position - 1);  
 myAdapter.notifyItemRemoved(position - 1);  
}

1. **Flowchart**

This is the overview of the program flow in ViewInventory.java and ManageOrders.java.

There are two more important screens, which are the Details screen for each Inventory and Orders (InventoryDetails.java and OrderDetails.java). These can be accessed by pressing an item row from its list. The way these Details code work is very similar to the Inventory and Order lists screens. The following is the flowchart if a user presses a row.

1. **Important notes**

Finally, I would like to add in some notes to my current program. This program currently works in local, which means it is not in any way connected to an online database. As with the program’s goal, it should be intended to be run online, but that has yet to be achieved.

In effect to this program not having an online database, this program does not give the option for users to upload their own photos when adding a new item. Currently, this program uses a default photo when a new item is added.

1. **EVIDENCE**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedHome screen, Inventory List, Add Inventory Item.**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedRemove Inventory Item, Item Details, Order List.**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedAdd Order, Remove Order, Order Details.**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedAdd Item to Order, Remove Item from Order, Save Warning Message.**

**REFERENCES**

Android Developers. User guide, Overview.

https://developer.android.com/studio/intro

Android Developers. Docs, Guides, Create a List with RecyclerView.

<https://developer.android.com/guide/topics/ui/layout/recyclerview>

Android Developers. Docs, Guides, Create a Card-Based Layout.

<https://developer.android.com/guide/topics/ui/layout/cardview>

Android Developers. Docs, Guides, Save key-value data.

<https://developer.android.com/training/data-storage/shared-preferences>

Coding in Flow. RecyclerView + CardView Tutorial - Android Programming.

<https://www.youtube.com/playlist?list=PLrnPJCHvNZuBtTYUuc5Pyo4V7xZ2HNtf4>

**CREDITS**

Yoga Mat Pictures:

[www.yogamatindonesia.com](http://www.yogamatindonesia.com)

[www.manduka.com](http://www.manduka.com)