COMPSYS 302 PROJECT 2

Version 2.5.2

Dr. Seyed Reza Shahamiri
Department of Electrical, Computer, and Software Engineering
Faculty of Engineering
The University of Auckland

You work as an Android developer at YouSee Soft, a company specialized in engineering mobile applications. This year, the company is running a competition in which all of its Android developers, in teams of two, are asked to participate in developing a native android app that can be used to showcase/sell some products and items. You can get inspired by similar apps like Trade Me and AliExpress. To be eligible to participate in this competition, you need to follow the instructions given in this document and ensure your app meets the specifications given here.

Dr. Reza Shahamiri and his team will be your coach in this competition. They will assess and monitor your progress, identify your final score and, ultimately, the winner(s) of this competition based on the criteria provided here. At the end of this competition, you will demo your application for the judges and other participants over Week 12, submit the source code via Canvas for detailed analysis, and will receive a score out of 45.

1. ASSESSMENT CONDITIONS

- To be done in teams of 2 students with both team and individual score components. Your partner cannot be the same as the first project.
- Open book
- No time limits (please refer to the due dates for each deliverable)

2. PROJECT MANAGEMENT TOOL

It is recommended that you use electronic project management tools, such as Trello, for task allocation, tracking, etc. Please ensure that you provide access to the coaching team.

3. GITHUB CLASSROOM

All developers need to use GitHub Classroom to work on the project collaboratively. **One of the developers** to follow this link to setup your team and repo. Your team name must be the same as the name used on Canvas. Once you create your team, kindly inform your other team member to follow the link and join your Classroom team. Please do not follow the GitHub Classroom invitation link ***before*** your partner invites you to do so, and then ensure that you join the correct team.

You must use the Git account associated with your UoA UID. Please do not use your personal GitHub account if it is not linked to your UOA UID.

4. THE APP SPECIFICATIONS

Your app should consist of at least three activities explained below and meet the required specifications.

a. Activity 1: MainActivity

This activity enables users to select a category of the items to be shown. For example, if your app sells cars, MainActivity provides different categories of cars for the user to browse such as Sedan, SUV, Hatchback, MPV, etc. **You need to provide at least three categories**.

This activity also enables users to search for specific items using the name of the item, and provides a panel for specific items such as bestselling, most viewed, etc. This panel is expected to populate via a RecyclerView – using fixed-coded views in this panel imposes a 40% penalty. You need to implement a logic in which the items in this panel get updated as the results of the user interacting with the app. A mock-up design of this activity is provided below.

In case the data items supplied for the "Top Picks" panel come from different model classes, you may need to properly implement the inheritance hierarchy of the model classes and use dependency injection via interfaces. You can read more about this on the net or refer to here.

b. Activity 2: ListActivity

Upon selecting one of the categories in MainActivity, the app shows ListActivity, in which a list of the items, with respect to the selected category, will be provided. In this activity, you need to properly use a ListView or RecyclerView to dynamically present the information from a data source that stores all of the items (i.e. a DataProvider class). The user should be able to scroll through this list and select one of the items. You need to provide **at least ten items per category**. Each item is presented as an object of its model class.

Only develop multiple versions of ListActivity to list different categories of items if they require different designs. Otherwise, use only one activity, adaptor, and ListView/RecyclerView for all items in every category. That means do not add a separate activity for, for example, SUVs, another activity for Hatchbacks, etc., unless you have a good reason – this is to promote maintainability.

A mock-up design of ListActivity is provided below.

c. Activity 3: DetailsActivity

Once users select one of the items of ListActivity (or SearchActivity, explained below), the app should bring up DetailsActivity, which provides all detailed information about the selected item. This activity enables the user to navigate through multiple images of the item (at least three images) using an image slider, ViewPager, etc. If present only one image for each item, a 20% penalty will apply.

Ensure proper information about each item is provided and the GUI properly categorizes related information. You can also add extra information, such as related items, etc. Ensure this page includes sufficient information about the item and is properly structured.

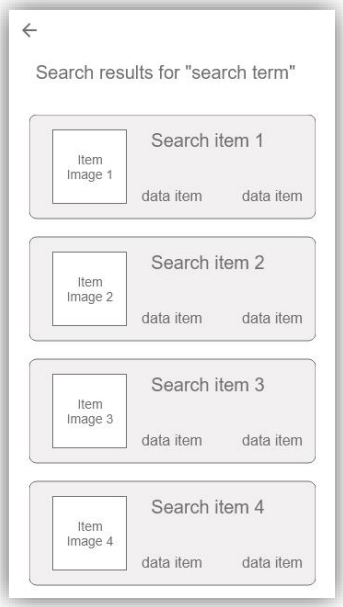
A design mock-up of this activity is provided below.

d. Search functionalities (or SearchActivity):

A search request from MainActivity invokes searching functionalities and shows the results of the search term. If no item is found, a proper message should be shown. To design this page, you can either implement ListActivity in a scalable manner to show the search results or alternatively design a separate activity called SearchActivity. **You need to ensure that all possible exceptions are properly handled.**

The search operation should follow a similar procedure to other well-known apps on the Android platform. Do not impose an unnecessary learning curve on your users.

A design mock-up for this activity is shown below.



e. App Design and Wireframe

All given design mock-ups are only simple samples, and you are expected to adopt different design ideas that best match the scope and users of your app as long as the specifications are met. You also need to properly follow Material Design guidelines.

A possible wireframe of this app is provided here. **You can also refer to the previous implementations of a similar project to get inspired.**

5. YOUR TASKS

To be eligible for this competition you are required to:

1. Working in pairs, select the context of your app. For example, your app can list cars, books, houses, cell phones, etc.,
2. Complete and submit the design document as explained in the deliverable section,
3. Implement a DataProvider class that is responsible for "simulating" fetching the data from a database. This class creates all model objects that the app needs. You can also use proper databases instead of a DataProvider class, which is considered one of the criteria for exceeding expectations. A sample of the DataProvider class is provided here,
4. Implement the app that provides the user experience (UX) explained above with your chosen design,
5. Use Android Studio and Java only,
6. Present your model classes via a proper inheritance association,
7. Try to use only one adaptor and ListView/RecyclerView for ListActivity if listview items design is similar among all categories,
8. Use animations/transitions. Research well-known apps in the context of your project to understand the type of animations/transitions expected. Your ability to self-learn is being evaluated here,
9. Ensure your app is responsive-resize and has an appealing graphical user interface,
10. Ensure good coding and software design practices and principles are followed,

11. Ensure your app follows Material Design Guidelines,
12. Use GitHub Classroom from the beginning of the project for versioning and frequently commit changes with proper descriptions so we can measure and evaluate each developer's participation (explained below). You must show commits and code addition from the beginning of the competition (i.e. the beginning of Part II) all the way to the submission day,
13. All pull requests must be reviewed by the other team member,
14. All engineers to demo the project,
15. Participate in peer-reviews,
16. One of the dev engineers compresses the entire project as a Zip file, including all project files, and submits the zip file on Canvas.

6. GITHUB USAGE POLICY

You are required to properly set GitHub Classroom and commit changes as you go through the project. We refer to each engineer's commit and code participation history to assess individual scores, meet deadlines, team commitment, etc. Please be advised that your GitHub performance is not the only indication of your commitment and participation, and we rely on other factors to identify your score. Please pay attention to the followings:

1. Do not commit all your changes at the end of the project, or you may receive no score (i.e., a zero mark!) if you cannot provide us evidence of your ongoing commitment to the competition and team.
2. Do not push fake commits or add and then delete bulk code to increase your commit and code participation metrics. This may result in a zero score and may affect your partner's score, too.
3. Lack of continuous and meaningful commit history from the beginning of the project or any attempt to manipulate your participation results in, at least, 30% deduction of your overall score if you fail to supply a proper justification.

Please note that the team repo is not an indication of submission, and all deliverables must be submitted via Canvas as well since GitHub is an external system for our organization. Your repo must be private and only be shared with the coaching team and no other participants. You cannot share anything about this project with other students if they are not your project partners **even** after course completion, and it is your responsibility to ensure your submissions are not available in any public domain for five years after you complete this course.

7. DELIVERABLES

The deliverables include 15 scores allocated individually and 30 scores allocated in teams. Deliverables are assessed holistically – we are interested in how you meet and exceed the expectations, the overall quality of your code and app, the design, GUI, and coding practices you adopted, the process, and your entire experience. Meeting the expectations does not guarantee you a full score – it is the quality of which the expectations are met. Securing a full score requires you to exceed expectations.

The deliverables for this competition are:

a. Design Docs (10 Score):

The design doc includes the following sections:

1. *Introduction* to the system that includes the context of your app

2. *System Modelling:* Design a high level "use case diagram" to explain the primary functionalities of the software. Ensure that all relationships are identified correctly, and the diagram sounds both semantically and grammatically. Supplying association multiplicities for your use case diagram is optional. Furthermore, briefly explain the use cases in your report.
3. *System Design:* Provide a high-level class diagram for your app properly showing all intended classes including the Activity classes, associations and their types and cardinality (one-to-one, one-to-many, etc.) are correct, and classes include all their fields/attributes, properties, and methods with their access type that you can identify at this early stage of development. For simplicity, we recommend that you consider any class dependencies, if any, (for example, when one class object uses a field or calls a method of another class object, or when an object is used in another class as a method input or output argument) as a simple association between two classes. The cardinality of all associations, aggregations, and composition relationships needs to be presented, but not for dependency relationships in case you decide to use them.
4. *GUI Design Mocks:* the UI mocks for each activity and the entire app. The UI mocks should include the overall theme, colouring, etc. that means the mocks should provide visual details as you intend to implement them. Supplying wireframes similar to the ones provided in this document does not have any mark,
5. *Project Schedule* (moving forward) that includes:
    a. Role of each software engineer moving forward with the development of the app
    b. Gantt chart (with breakdown of responsibilities for each member)

This document needs to be compiled by both team members; however, each developer will receive individual scores based on the roles and responsibilities mentioned below. In particular, both developers should compile sections 1 and 5 collaboratively, and then one of the developers is responsible for sections 2 and 4, and the other developer for section 3. **It is mandatory for the teams to follow this table precisely** – the task allocations are done based on the estimated effort required for each section and to ensure fairness and equal work distribution among team members.

Please compile all sections in one document, and one of the developers will submit the complete report for this deliverable on behalf of the team. Please note that handwritten diagrams and/or photos of handwritten drawings are unacceptable; you are responsible for ensuring that every figure and information supplied is easily readable. **Kindly ensure that you mention in your report the name of the developer who completed the individual sections per the table below:**

Section	Developers
1. Introduction	Both developers
2. System Modelling	Developer 1
3. System Design	Developer 2
4. GUI Mocks	Developer 1
5. Project Schedule	Both Developers

Some tips for the design docs:

1. Every use case that you supply in your use case diagram must be implemented in the app. Some of you may overthink the project and add use cases for user authentication and user management, inventory management, shopping capabilities, etc. You can add all these functionalities to your app, but they are not required for this competition unless you intend to

have these extra functionalities to exceed expectations. You need to ensure that you conduct enough research and feasibility study so that the proposed use cases will be realized.

2. Starting from lesson 12 you will develop an app called Learn Māori with similar functionalities and use cases with this competition that helps you learn skills that can be easily transferred to your app. You can have a look at the complete version of this app here to better understand the expected user experience of this app.
3. The class diagram that you design at this stage is a "conceptual class diagram", which means it is not the final class diagram, and it is likely that you will change it during implementation once you have more technical knowledge about your project - these changes are acceptable and expected. Conceptual class diagrams are not as detailed as implementation class diagrams.
4. A class diagram doesn't necessarily contain all types of associations, i.e., you don't have to use all four types of class relationships. For example, your class diagram for this project may not have any composition/aggregation association.
5. The class diagram of this project is "slightly" more complex than its use case diagram since you are already familiar with OOP and classes, but use cases might be new to you, which implies a steeper learning curve. The increased complexity of the class diagram is to compensate for the additional learning required for use case diagrams.
6. Ensure your use case and class diagrams are consistent - they present the same app after all.
7. **While each developer has responsibilities for specific sections indicated in the table, both team members need to collaborate with each other and coordinate to ensure the consistency of this document, your designs, and your product, and that everyone is aware of the sections complied by the other member. You should not complete your sections in isolation. We recommend that you all work together as a team, but you take responsibility and lead the discussions of your own sections.**
8. The roles and responsibilities mentioned in the above table only apply to the Design Doc, and teams can decide among themselves on the members' roles for the remainder of the project.

b. Project Demo (20 Score):

All team must be ready to demo their app by Monday of Week 12, but submission of the source code can be towards the end of Week 12.

Here, you are required to present and demo your app in pairs in 8 minutes slots **(3 min each developer, 2 min for Q&A)** and may need to answer some technical questions. You also need to show 1) the implementation of the project is consistent with your project schedule and roles and responsibilities supplied in your Design Doc, and 2) in case of any inconsistencies, explain why your initial design/plan had to be changed. A 30% penalty applies to developers who do not supply this information. **You need to prepare a set of slides and present them alongside your app. The slides are to be submitted on Canvas.**

Please ensure that your BYOD is ready for demo as your time starts on schedule, and you have proper adaptors if your BYOD doesn't have any standard HDMI port. For app demo, please use the emulator in Android Studio as the demo location may not have document cameras.

The project demos will happen across Week 12's both lecture sessions in front of the judges and other participants. Additionally, extra demo session(s) is scheduled as shown below:

Demo Session	Date and Time	Location	Attendance	Note
1	Wednesday June 4, 12-2 pm	Biology Building, Room 204	All teams	All developers presenting or observing to be in the venue before the session start time to avoid disturbing the presenting teams. Attendance is mandatory for all students during the official lecture hours and all teams presenting in the same sessions. Attendance is optional for non-presenting teams during extra sessions, but they are highly encouraged to participate in peer reviews. All attendees are required to complete the peer reviews.
2	Thursday, June 5, 10 am – 5 pm	423-340	All teams presenting	
3	Friday, June 6, 10 am – 12 pm	423-340	All teams presenting	

Attendance by all students is mandatory for demos scheduled during the normal lecture time. However, attendance during the extra demo sessions is optional for non-presenting teams but mandatory for teams presenting during the session.

All teams, please book your demo via this sheet **by the end of week 10**. A 20% penalty applies to teams without any booking. **Please fill out the mandatory sessions first.** Otherwise, we will move your session if any slot is available. Likewise, complete the "MUST COMPLETE" tab once your booking is done. We advise that you demo in the earlier slots to have more time to action feedback before you submit your final code.

During your app demo, please ensure that you show the following. You need to ensure that your demo provides all the requested information. Any missing information will impose penalties.

App Demo Criteria (same for each team member)

The requested UX is implemented as explained in this document. This includes the followings: Misc.: • App is functional and does not break down MainActivity: • At least three categories of items are provided • Tapping on each category properly opens ListActivity and passes over information about the selected category ListActivity: • At least ten items per each category is presented in ListActivity • Tapping on each item properly opens up DetailsActivity and passes over the information of the selected item DetailsActivity: • All detailed information about the selected item is shown Search Functionalities: • Clicking each item properly opens up DetailsActivity and passes over the information of the selected item
The UI is resize-responsive
The UI design is professional and appealing
Animations and Transitions are properly used

Consistency with the Design Doc
Roles and responsibilities are explained

Additionally, your individual presentation performance based on the criteria below will be considered to identify your individual score. Each developer needs to equally present and ensure the presentation enables us to assess the presenter properly:

Individual Presentation Criteria

Speaker maintains good eye contact with the audience and is appropriately animated (e.g., gestures, moving around, etc.), and is properly dressed.
Speaker uses a clear, audible voice.
Length of presentation properly fits the assigned time frame.
Information was well communicated, complete, and good language skills and pronunciation are used.

One of the developers to submit the demo slides on Canvas.

Judging and Peer-Review: You are required to judge **all** app demos during the **official lecture time**, and complete this form for each team separately (one form for both developers of a team). We also request that all teams attending the demos to complete the peer-reviews during the extra demo sessions. The form must be completed immediately after each demo is done and submissions outside the lecture time will be deleted. Your UoA UID and timestamp are captured when you submit the form so please ensure that you logged in using your uid@aucklanduni.ac.nz Google account.

We expect at least **17** submissions per student, but we encourage you to peer-review as many teams as possible. You can attend any other demo sessions (in addition to mandatory sessions) that fit your schedule to complete the minimum number of peer-review submissions. **To submit a peer review, you must be present during the session**. Lack of participation in judging your peers results in a 30% penalty. The form must be completed immediately after each demo; otherwise, the submission will be deleted and will not count toward your minimum peer-reviews completed. Fake reviews will be deleted, and the penalty will be applied.

Presenters, please ensure that you show your team number and your names clearly at the beginning and end of your demo slides.

c. Project Code (15 Score)

You need to submit the code for inspection. We will consider the following criteria in addition to our holistic evaluation mentioned before to compile the code score:

MainActivity
MainActivity enables users to search for specific items
The "Top Picks" (or similar) panel in MainActivity is populated via a RecyclerView and gets updated as the app is being used. Selecting an item shows its details via DetailsActivity.
ListActivity
The data items are populated via a ListView or RecyclerView
Each data item is presented as an object of its model class
One Activity and adaptor are used for ListActivity. This criterion is not applied if different layout designs are required for each category of items.
DetailsActivity
This activity enables the user to navigate through at least three images of the selected item.
Search Functionalities
Searching functionality is invoked by a search request from MainActivity and

shows the results of the search term. All possible exceptions were captured and handled properly.
If no item is found, a proper message should be shown instead.
Misc.
The class diagram presented in Design Docs is implemented properly. In case of inconsistencies, sufficient rationale is provided.
Proper usage of ViewHolders
All data preparation operations are provided via DataProvider class.

We will also check for the overall code and UI quality, UX, proper usage of software principles and practices, etc. Poor quality may affect the assessment of other project deliverables as well.

This deliverable is submitted as a team. One of the developers to submit the code on behalf of the team.

8. TESTING

You are required to properly test and debug your app to ensure it runs without a fault and produces the required results.

9. PROJECT CODE SUBMISSION

Please compress your completed code into a zip file or download the source code (zip) on the release page (with cs302-2022-Java-teamID-submission as the file name) and upload it to the Project 2 code release in the 'Assignments' section on Canvas. Please ensure the zip file contains all project files and folders.

10. RESOURCES

- For the images and icons, you can design them yourself or download them from the internet (providing adequate licensing is granted). You can also use generative AI to generate the images.
- For Information about Android animations and transitions, please refer to here, here, and here. You can also inspire by studying the Material Design Guidelines and browsing the app here.
- For information about how to use RecyclerView refer to here and here. We have also developed a demo application to show you how to use RecyclerView in the context of this project.
- Designing the Use Case and Class Diagram is self-learning. We expect that you conduct your research and study how to design and practice them. However, the following resources are a good start for your self-study:
    - Use Case Diagram: Slides, Tutorial
    - Class Diagram: Slides, Tutorial

11. Links

To aces the SharePoint links, ensure that you are singed in with your UoA UID.

Trello:
https://trello.com/
GitHub Classroom Project Enrolment:
https://classroom.github.com/a/BGseP8Hn
Dependency Injection and SOLID Slides:
https://uoa-my.sharepoint.com/:p:/g/personal/ssha631_uoa_auckland_ac_nz/EQOBhGV8CfZEkp5bi-XamiQBqaGM0BBQVKSZvils1S_W_Q?e=1fvPFh

DataProvider Class Sample:
https://github.com/university-of-auckland-cs-302/COMSYS302Exercises/blob/master/Android/RecyclerViewDemo/app/src/main/java/com/example/recyclerviewdemo/DataProvider.java
Demo Booking Sheet
COMPSYS302 Demo Booking Sheet.xlsx
ViewPager:
https://developer.android.com/training/animation/screen-slide
Material Design :
https://material.io/develop/android
Project Wireframe Sample:
https://xd.adobe.com/view/972b0ab8-d288-4de1-6b66-fa4006ee51a3-e11a/?fullscreen
Project samples:
https://uoa-my.sharepoint.com/:f:/g/personal/ssha631_uoa_auckland_ac_nz/EhuHp2TMenVDsWRnxNAGOn4BA8DWflC4VqW_1hc99a2PUA?e=mWtpytLinks to an external site.
Learn Māori app:
https://play.google.com/store/apps/details?id=com.rezanet.learnmaori
Peer Review Form:
https://forms.gle/NGt95YXpYXWZEL6y9
Android Animation and Transition:

- https://developer.android.com/training/animation/index.html
- http://www.androiddesignpatterns.com/2014/12/activity-fragment-transitions-in-android-lollipop-part1.html
- https://www.youtube.com/watch?v=K3yMV5am-Xo

Recycler View:

- https://developer.android.com/guide/topics/ui/layout/recyclerview
- https://guides.codepath.com/android/using-the-recyclerview
- Demo App: https://github.com/university-of-auckland-cs-302/COMSYS302Exercises/tree/master/Android/RecyclerViewDemo

Use Case Diagram:
Slides: https://uoa-my.sharepoint.com/:p:/g/personal/ssha631_uoa_auckland_ac_nz/EWt8SXr3w2ZNl_2e1dzHQ90B7JLYvd5DFz1bXgtkdxdVCQ?e=fv6Zzf
Tutorial Video: https://youtu.be/4emxjxonNRI
Class Diagram:
Slides: https://uoa-my.sharepoint.com/:p:/g/personal/ssha631_uoa_auckland_ac_nz/EWt8SXr3w2ZNl_2e1dzHQ90B7JLYvd5DFz1bXgtkdxdVCQ?e=fv6Zzf
Tutorial Video: https://youtu.be/6XrL5jXmTwM

12. PROJECT 2 SCHEDULE

The project schedule is provided below. The deliverables due dates are supplied on Canvas.

	To Do	Deliverables
Week 7	Team formations finalized by Friday	
Week 8	Git repos finalized by Friday with both developers joined	Assignment 2 - Sunday
Week 9		Design Doc - Sunday
Week 10	Demo Bookings	
Week 11		
Week 12	Demo	Project Code - Sunday

13. FEEDBACK

Please submit your feedback via email to reza.shahamiri@auckland.ac.nz.

14. FAQ

1. *Can I use a different a programming language?*

   No, you can't. All projects must use Java and native Android platform. Any team using other programming languages or platforms will be disqualified for this competition and receive a zero score.

2. *Should we keep our GitHub repo updated with the final version of the code?*

   Yes, your repo and the code you submit on Canvas must be identical.

3. *What if we make changes to the version of the app we demo and the final code submitted on Canvas?*

   It is fine that your demoed app and final submission to be slightly different as you may use the feedback received during the demo and last week of the competition to polish your app, especially for teams that present on the first days of demo.

4. *Do we need to supply all information about each item in the DetailsActivity?*

   It is recommended that you provide all data for each of your model objects. However, the data can be mock data if you cannot find real information.

5. *How should I select the Top Pick items in MainActivity?*

   This is completely up to you. You can define different criterion such as number of views, popularity, or any other criterion you prefer. This criterion can be initialized via your DataProvider or any other mechanisms you prefer. Please be advised that this panel is expected to show some items even after the first app run.

6. *Do we need to add all folders of the project to our repo?*

   The important folder is *scr* so you can ignore the other folders if you want. If you have other files and folders that may be needed such as any documentation, please add them to your repo to.

7. *Do you use any special emulator to test our apps?*

   Your app should be resize-responsive. However, we exclude uncommon devices such as very small screen or very large screen devices. Ensure your app shows up properly on devices with screen sizes of 5-7 inch.

8. *Should we design our app for landscape mode too?*

   Your app is resize responsive, it should work on both portrait and landscape mode. You generally design the layout for portrait mode, but test it is properly shown on landscape mode as well. If an activity should not be shown in landscape mode, you need to lock that activity to portrait mode.

9. *Should we use an automatic image slider or is it essential to have one which the user can navigate through on DetaislActivity?*

   As the designer of the app it is your decision to have the images automatically slide, or the user be able to navigate them manually. However, automatic sliders are more popular on the landing

page of apps or websites while it is preferred for users to be able to navigate manually when they see item details.

10. *Can we output a toast if a search is invalid or does it have to be a message on the ListActivity interface?*

    For unsuccessful search enquires, users are generally informed via a persistent method. For other exceptions, it is your decision to select the way in which you inform your users, as long as your approach is generally acceptable to users and consistent with other frequently used apps.

11. *This document suggests we use a RecyclerView for the top picks section; is it okay to use a ListView and adapter since it performs about the same as a RecyclerView?*

    Using a RecyclerView is a requirement for the Top Picks panel of MainActivity since ListViews do not provide horizontal scrolling of list items. For other places, it is up to the team whether to use ListView or RecyclerView.

12. *Can we use a glide dependency to get images from the internet?*

    Yes, this is acceptable.

13. *Can we use Fragments?*

    You need to have the three activities. However, you could use fragments in the second activity in case you want to supply different designs for listing items in each categories.

14. *It seems pretty difficult to find multiple photos of the same item through officially licensed methods. Is it possible for us to gather images manually from another website (Amazon, AliExpress etc.) and then mention the source somewhere in the app or final project documentation?*

    Most of the times using resources for educational reasons is fine so if you are not publishing your app it should be ok. However, if publishing your app you need to check for the copyrights. You can also take your phone and capture the images if that's possible.

15. *What tools can we use to design the diagrams?*

    You are free to use any tool you prefer. Some recommendations are below:
    - UML Use Case Diagrams - Lucidchart
    - What is a Use Case Diagram - Visual Paradigm
    - UML Use Case Diagram - Java At Point
    - UML Class Diagram Tutorial - Visual Paradigm
    - UML Class Diagrams - Java At Point
    - UML Class Diagrams - Lucidchart
    - Regarding GUI design mocks, you can use Figma, inside which you can create a Prototype': an interactive wireframe that simulates how it will flow on the app. There is also a plugin for Material Design icons.

16. *We are wondering what should be considered a use case in the use case diagram. For example, assume a case study in which the user is prompted to enter information (student ID, course ID etc) - is that a use case? What about when the system parses the user's input and tells them if it was invalid?*

    Use cases capture the primary functionalities of the system. The level of detail you provide is with respect to the complexity and the context of the system. There also multiple levels of use case diagrams in which detailed use case diagrams can capture more detailed operations of the system, as actors see them. Here you only need to provide the high-level use case diagram. You need to act as actors to decide what functionalities are major and important.

17. *How detailed should the use case diagram be?*

    You only provide the major use cases in a high level diagram. You need to decide what major use cases in your app are and add them to this diagram. Ask yourselves what are the primary functions of your app from the actor's point of view, then these main functionalities become your use cases, plus other use cases that you may need to service the main use cases. Additionally, you only need to provide use cases for the functionalities of the app based on the given specifications.

18. *Do I need to add functionalities to sell items?*

    This is not needed. For simplicity, this app only showcase items.

19. *WRT the class diagram, assume "Method A of Class1 creates and returns an instance of Class2", and "Method B of Class1 uses this instance of Class2 as a parameter". Should we just have one association here between Class1 and Class2?*

    Yes, either there's one usage of an object or many, you only use one association to show that.

20. *Should we show cardinalities for all association types?*

    Instead of inheritance (and dependencies if used), cardinalities for other associations are needed.

21. *If a class accesses a static method of another class, what are the cardinalities of this relationship?*

    We do not create an object of a class if we only use static methods of the class. Cardinality is usually identified as how many instances of Class1 associate with how many instances of Class2, and vice versa. If the association between two classes only limits to static methods, it makes sense to consider it as a one-to-X relationship. Otherwise, the cardinality clause above applies.

22. *Should we add multiplicities to our dependencies?*

    This is a recommendation that you don't consider dependency for this project - to simplify it, but you still need to capture them in your class diagram. That is, when there's a relationship between two classes, you first decide whether it's inheritance, composition, or aggregation. If the relationship doesn't fall into any of these, then you consider it as a simple association, that makes your task easier rather than arguing whether it's a dependency or simple association. If you're adopting this approach, then yes, you need to supply the cardinality of all associations/compositions/aggregations but not for generalizations. However, it's up to you whether you'd still like to use dependencies.

23. *Should we add classes that we intend to import in the class diagram, such as Intent?*

    You only include classes that you intend to implement or override (such as Activity) in the class diagram. Otherwise, you do not show platform classes that you only import but not modify.

24. *If two classes are associated with a strong relationship (eg. association, aggregation, composition), can we omit a dependency relationship between those two classes?*

    You are required to do so in this project. It's either dependency OR other associations but not both. The preference is to use associations.

25. *I missed a due date. Do any penalties apply?*

    Unless you have a reasonable and acceptable reason for missing due dates, yes, penalties will apply. This applies to all deliverable due dates and other due dates such as team formations, git

repo setups, etc. Being able to deliver on time and follow instructions are among the criteria that we consider in our holistic assessment approach as both product's features and quality, and the process in which you follow will be assessed.

26. *Can I change my partner?*

    Changing partners after teams are setup creates a ripple effect. As such, changes in teams are not possible after teams are formed on Canvas. It is important that you develop skills to work with others who you do not know closely since you usually do not get to select your partner in a real professional environment. Your team working skills will be assessed in this competition.

27. *Can I use generative AI, such as ChatGPT, in this project?*

    It is important that your work is original and you can explain it completely. You can use tools to help you complete the project better. However, if you are using code or other materials you did not produce, you must disclose them and explain from where and to what extent you have used them in your project. You are still required to be able to explain any material you submitted, even if inspired by tools or other individuals, and prove you understand the project and course materials well. Failure to explain your submissions and demonstrate sufficient technical understating or disclose the use of generative AI or other tools and sources that helped you in your project may be considered plagiarism.