SOFTENG 306 Project 1

Version 3.3

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

Table of Contents

1 **Assessment Conditions** **2**

2 **Project Management Tool** **2**

3 **GitHub Classroom** **2**

4 **The App Specifications** **2**

4.1 *Activity 1: MainActivity* *2*

4.2 *Activity 2: ListActivity* *3*

4.3 *Activity 3: DetailsActivity* *4*

4.4 *Search Functionalities (or SearchActivity):* *4*

4.5 *Database* *5*

4.6 *App Design and Wireframe* *5*

5 **Your Tasks** **5**

6 **GitHub Usage Policy** **6**

7 **Deliverables** **6**

7.1 *Design Doc (15 Individual Score)* *7*

7.2 *Project Demo (15 Score – individual and team components)* *9*

7.3 *Project Code and Quality Report (20 Score)* *10*

8 **Testing** **11**

9 **Project Code Submission** **11**

10 **Resources** **11**

11 **Project Schedule** **12**

12 **Links** **12**

13 **Feedback** **12**

14 **FAQs** **12**

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 software engineers, in teams of three, 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. Your ability to apply best software design principles, clean code, and GUI design will mainly be assessed in this competition.
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 in this document. At the end of this competition, you will demo your application over Week 6, submit the source code and other deliverables via Canvas for detailed analysis, and will receive a score out of 50.

1 Assessment Conditions

- To be done in teams of 3 students with both team and individual score components.
- 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 GitHub Classroom Assessment Link to setup the 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 members 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 to your UoA UID. Please do not use your personal GitHub account.
Admin access to your repository is enabled for all team members. It is the team's responsibility to maintain the repo, keep it private, and not provide access to anyone outside the team.

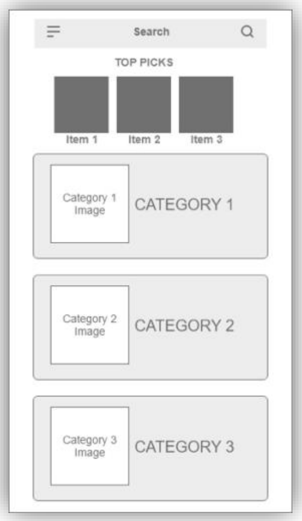
4 The App Specifications

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

4.1 Activity 1: MainActivity

This activity enables users to select a category of the items to be shown. For example, if your

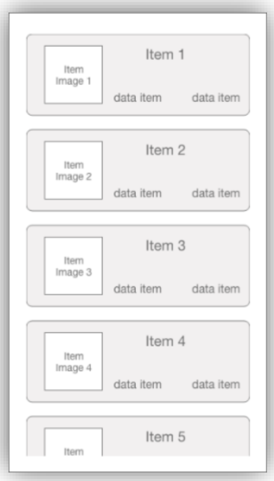
app sells cars, MainActivity provides different categories of cars for the user to browse such as Sedan, SUV, Hatchback, MPV, etc. You need to provide at least three categories.
This activity also enables users to search for specific items using the name of the item, and provides a panel for specific 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.



The model classes need to follow a proper inheritance structure and enable **dependency injection** via interfaces, and ensuring SOLID principles are not violated, as discussed during the lectures. Your abilities to maximize software maintainability will be holistically assessed in this project.

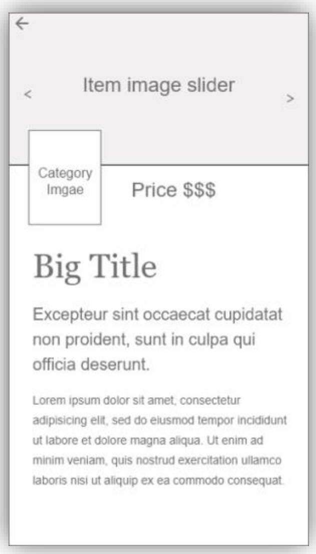
4.2 Activity 2: ListActivity

Upon selecting one of the categories in MainActivity, the app shows ListActivity in which a list of the items, in 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 the database (explained later) that stores all of the items. 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 each category. Each item is presented as an object of its model class.
For model classes, you need to use a different class for each category, and properly use dependency injection and inheritance hierarchy with one custom adaptor to populate all this activity. It is expected to see slightly different layout design for each category to showcase this requirement. This is further explained in lesson "*14 - Android - Working with Database*".
It is also expected that one custom adaptor for these activities is used and designed in a way to accommodate different item layouts – this is to promote maintainability. Please provide sufficient justification if you decide to implement more than one adaptor for these activities and explain why a single adaptor could not be used. Otherwise, a penalty of up to 30% may apply to your project.
A mock-up design of ListActivity is provided below:



4.3 Activity 3: DetailsActivity

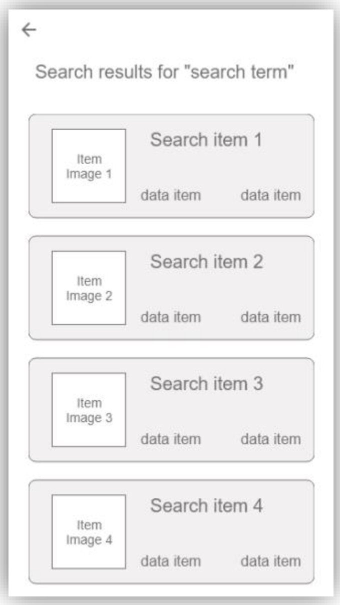
Once users select one of the items of ListActivity (or SearchActivity explained below), the app should bring up DetailsActivity that provides all detailed information about the selected item. This activity enables the user to navigate through multiple images of the item (at least 3 images) using an image slider, ViewPager, etc. If only one image for each item is presented, a 20% penalty will apply.
Ensure proper information of each item is provided, and the GUI properly categorize related information. You can add extra information as well, such as related items, etc.
A simple design mock-up of this activity is provided below:


Item image slider
Category Imgae
Price $$$
Big Title
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt.
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

4.4 Search Functionalities (or SearchActivity)

Searching functionalities are invoked by a search request from MainActivity 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 so that it can also show the search results, or alternatively design a separate activity called SearchActivity. You need to ensure all possible exceptions are properly handled.
The search operation should follow a similar procedure to other well-known apps in the Android

platform. Do not impose unnecessary learning curve for your users.
A design mock-up for this activity is shown below:



4.5 Database

You are required to store your app data in Firestore collections. Your app should properly interact with the database to capture and manipulate data as needed.

4.6 App Design and Wireframe

All given design wireframes 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.
A possible Wireframe of this app is provided. You can also refer to the previous implementations of a similar project to get inspired:
Sample 1
Sample 2
Sample 3

5 Your Tasks

To be eligible for this competition you are required to:

1. 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 the Firestore database for your app,
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 for ListActivity (or ListActivities),

8. Improve the UX of your app by adding smooth animations and transitions,
9. Research well-known apps in the context of your project to understand the type of animations/transitions expected. Your ability to self-train is being evaluated here,
10. Ensure your app is responsive-resize and has appealing graphical user interface,
11. Ensure good coding and software design practices and principles are followed,
12. Ensure your app does not violate SOLID principles,
13. Ensure your app follows Material Design Guidelines,
14. Use GitHub Classroom from the beginning of the project for versioning and frequently commit changes with proper description 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 I) all the way to the submission day,
15. All pull requests must be reviewed by the other team member,
16. All engineers to demo the project,
17. One of the dev engineers to compress the entire project as a Zip file including all project files and submit 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 as part of assessing individual scores, meeting deadlines, team commitment, etc. Please be advise that your GitHub performance is not the only indication of your commitment and participation, and we rely on other factors to identify your score. Kindly pay attention to the followings:

- 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.
- 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 partners' scores too.
- Use your git account associated with your UID.
- Lack of continuous and meaningful commit history from the beginning of the project, any attempt to manipulate your participation, or using your personal git account results in, at least, 30% deduction of your overall score if you fail to supply a proper justification.

Please note 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 to our organisation. 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 developers if they are not your project partner.

7 Deliverables

The deliverables include 21 score allocated individually and 29 score 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, design, GUI, and coding practices you adopted, the process, and your entire experience. Meeting the expectations does not guarantee

you full score – it is the quality of which the expectations are met. Securing full score requires you to exceed expectations.
The deliverables for this competition are:

7.1 Design Doc (15 Individual 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, ensuring 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, etc.) 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. Ensure SOLID principles are considered properly.
4. Design Analysis: This section is composed of the following sub-sections:
    a. Explain what design smell(s) you identified in your initial attempts of the class diagram.
    b. For each SOLID principle, explain how it is applied in the class diagram. If a principle was not considered, explain why and justify.
    c. How the smell(s) identified in 4.a are removed or improved, or even got worse in the class diagram.
    d. Analyse, compare, and explain your design with respect to maintainability, reusability, coupling, and coherency. Your judgment should be objective and based on metrics: justify your judgment by properly measuring software metrics and argue software quality attributes.
5. GUI Mockups: the UI mocks for each activity and the entire app. The UI mocks should include colours, typographies, photos, other visual design elements, 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 score,
6. The data schemes (the collections and documents in Firestore, or in RDBMS terms, the database and tables),
7. Project Schedule (moving forward) that includes:
    e. Role of each software engineer moving forward with the development of the app
    f. Gantt chart (with breakdown responsibilities for each member)

This document needs to be compiled by all team members; however, each developer will receive individual scores based on roles and responsibilities mentioned in the table below. **It is**

mandatory for the teams to follow this table precisely – the task allocations are designed 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 to submit the complete report for this deliverable on behalf of the team. Please note that handwritten diagrams and/or photos of handwritten drawings are not acceptable; it is your responsibility to ensure every figure and information supplied is easily readable. Kindly ensure that you mention in your report the name of the developers who completed the individual sections per table below:

Section	**Developer 1**	**Developer 2**	**Developer 3**
1. Introduction			x
2. System Modelling	x		
3. System Design		x	
4.a Initial Design smells		x	
4.b: SOLID principles	x		
4.c: Software Design Smell Analysis			x
4.d: Metrics and quality attributes		x	
5. GUI Mocks			x
6. Data schemes			x
7.a Role of each software engineer	x		
7.b: Gantt chart	x		

Some tips for the design docs:

1. Every use case that you supply in your use case diagram must be implemented in the app. You need to ensure that you conduct enough research and feasibility study so that the proposed use cases will be realized.
2. The class diagram that you design at this stage is a "conceptual class diagram", that means it is not the final class diagram and it is likely that you change it during implementation once you have more technical knowledge about your project. These changes are acceptable and expected, but you need to explain them at the end of the project.
3. A class diagram does not necessary contain all types of associations, i.e., you do not have to use all four types of class relationships. For example, your class diagram for this project may not have any composition/aggregation association.
4. The class diagram of this project is more complex than its use case diagram since you already are familiar with OOP and classes but use cases might be new to you that imply a steeper learning curve. The increased complexity of the class diagram is to compensate for the additional learning required for use case diagrams.
5. Ensure your use case diagram, class diagram, design mocks, etc. are consistent - they all present the same product after all.
6. While each developer has responsibilities for specific sections indicated in the table, all 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 other members. You should not complete your sections in isolation. We recommend that you all work together as a team, but each team member take responsibility and lead the discussions of your own sections according to the table presented above.
7. The roles and responsibilities mentioned in the above table only applies to the Design

Doc and teams can decide among themselves on the members' roles for the remaining of the project.

7.2 Project Demo (15 Score – individual and team components)

All team must be ready to demo their app by Monday of Week 6, but submission of the source code can be towards the end of Week 6.

Here, you are required to present and demo your app in ten minutes slots (2-3 min each developer, eight minutes overall, plus 2 min for Q&A and handover), and you 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 6's both lecture sessions in front of the judges and other participants. Additionally, extra demo sessions are scheduled. The demo schedules are provided in the table below:

Demo Session	Date and Time	Location	Attendance	Note
1	Tuesday August 22nd 1-3pm	Clock Tower - South, Room 039	Attendance is mandatory for all students	Mandatory peer-review
2	Wednesday August 23th 10am-12pm pm	B206-203 (Humanities Building)	All teams can attend	All developers presenting or observing to be in the venue before the first presentation to avoid disturbing the presenting teams
3	Wednesday August 23st, 1-2pm	Clock Tower - South, Room 039	Attendance is mandatory for all students	Mandatory peer-review
4	Thursday August 24th 10am-2pm	B423-340 (Conference Centre)	Only presenting developers	All developers presenting to be in the lab 15 minutes before their schedules

All teams must book their demo via Demo Booking Sheet by the end of week 4. A 20% penalty applies to teams without any booking. Please fill the mandatory sessions first. **Do not add new slots.**

During your app demo please ensure that you demonstrate the followings (App Demo Criteria – same for all developers). You need to ensure that your demo provides all the requested information. Any missing information will impose penalties.

The requested UX is implemented as explained in this document. This includes the followings: 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
App is functional and does not break down
Roles and responsibilities are explained
Consistency with the Design Doc (planned schedule, SOLID principles and design smells, any diversion from the plan, etc.)
Length of presentation properly fits the assigned time frame

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:

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.
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 (mandatory) lecture time, and complete Demo Peer-Review Form for each team separately (one form for all developers of a team). 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 are expecting 13-14 submissions per each developer. Lack of participation in judging your peers results in 30% penalty.

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

7.3 Project Code and Quality Report (20 Score)

You need to submit the code for inspection. Additionally, each team to prepare a short report explaining the final realization of SOLID principles and any other good coding practices the team applied.

We will consider the following criteria in addition to our holistic evaluation mentioned before to compile the score for this deliverable:

MainActivity
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
One adaptor is used for listing all categories' 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 captures and handled properly.
If no item is found, a proper message should be shown instead.
Misc.
Proper usage of ViewHolders in activities, adaptors, etc.
All app data is supplied via Firestore.
Proper inheritance structure for the model classes implemented.
Quality Report
SOLID principles are properly applied and realized (both structurally and within the classes) as proposed in the Design Doc, and any inconsistencies are properly explained.
Other indication of good coding practices such as consistent naming convention, commenting, proper usage of packages, etc.

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

This deliverable is submitted as a team. One of the developers to compress the code and report into one file and submit it 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. Applications that crash during our assessment or do not run will lose a considerable amount of score. However, this penalty does not apply during your demo as you may not demo your final product and still have the opportunity to fix the bugs before you submit the final code.

9 PROJECT CODE SUBMISSION

Please compress your completed code and report into a zip file or download the source code (zip) on the release page (with se306-202X-Java-groupID-submission as the file name) and upload to Project 1 code release in the 'Assignments' section on Canvas. Please ensure the zip file contains all project files and folders and the report.

10 RESOURCES

1. For the images and icons, you can design them yourself or download them from the internet (providing adequate licensing is provided).
2. For Information about Android animations and transitions please refer to developer.android.com, androiddesignpatterns.com, and this YouTube video. You can also inspire by studying the Material Design Guidelines and browsing the Learn Māori app.
3. For information about how to use RecyclerView refer to developer.android.com and

guides.codepath.com. We have also developed a RecyclerView Demo App to show you how to use RecyclerView in the context of this project.

4. In case you need to refresh your memory with the requested UML diagrams, you can refer to your previous courses or online materials, or the following lessons: Use Case Diagram, Class Diagram

11 PROJECT SCHEDULE

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

	To Do	Deliverables
Week 1	Team formations finalized by Friday	
Week 2	Git repos finalized by Friday with all developers joined	
Week 3		Design Doc
Week 4	All teams to book their demo	
Week 5		
Week 6	Demo	Project Code and Quality Report

12 LINKS

Android Animations and Transitions:

- developer.android.com
- androiddesignpatterns.com
- YouTube video
- Learn Māori

Class Diagram

Demo Booking Sheet

Demo Peer-Review Form

GitHub Classroom Assessment Link

Material Design Guidelines

RecyclerView:

- developer.android.com
- guides.codepath.com
- RecyclerView Demo App

Sample 1

Sample 2

Sample 3

Trello

Use Case Diagram

Wireframe

13 FEEDBACK

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

14 FAQS

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. 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.
3. 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 the primary functions of your app from each actor's point of view are, then these main functionalities become your use cases, plus other use cases that you may need to service the main use cases.
4. 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.
5. 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.
6. Should we show cardinalities for all association types?
   Instead of inheritance (and dependencies if used), cardinalities for other associations are needed.
7. 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.
8. 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.

9. If two classes are associated with a strong relationship (e.g. 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.

10. What metrics should we calculate?
    It depends on how you decide to formulate your arguments. As long as your arguments are valid and the metrics you select to measure indicate them, we will accept. Having said that, a collection of object-oriented metrics that clearly rationalize your argument wrt the requested quality attributes is recommended.

11. Do we calculate CBO for classes with only static functions?
    Yes, static-only associations are also counted.

12. Are constructors one of the methods included in the $WMC_{Fan\text{-}In}$ calculation?
    Yes, they are.

13. When calculating Fan-in, should methods that invoke other methods in the same class be counted? For example, if MethodA and MethodB are in the same class and MethodA invokes MethodB, would that be counted when calculating the Fan-in for MethodB?
    Yes, both internal and external calls are to be considered.

14. Do I need to add functionalities to sell items?
    This is not needed to meet the expectations. For simplicity, this app only showcases items. However, you can simulate selling functionalities or other extra features if you intend to exceed expectations.

15. 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.

16. 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.

17. 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. You need to ensure that your DetailsActivity is realistic and supply proper information.

18. 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 in the dataset or

any other mechanisms you prefer. Please be advised that this panel is expected to show some items even in the first app run.

19. Do we need to add all folders of the project to our repo?
    Yes, we need all android project files and folders on your repo and also submitted on Canvas to properly run your project. If you have other files and folders that may be needed such as any documentation, please add them to your repo to.

20. 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.

21. 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.

22. 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.

23. 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.

24. 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.

25. Can we use fragments instead of activities?
    Yes, you can replace activities with fragments when necessary, for example, if you use the Bottom Navigation Activity template.

26. In our application it makes almost no sense to have multiple types of list activity, and I believe doing so would add confusion to the user. That being said the project document states that the ListAcitivty between categories has the be "different". Could we please get some clarification on how "different" we are required to make our ListActivity?
    What you need to do here is to slightly modify the items you show in the ListView/RecyclerView of ListActivity. The ListActivity and its adaptor is an easy opportunity to show SOLID and dependency injection are applied; otherwise, you need to find other ways to prove their applications in your project.

You can ask yourselves if there are different data attributes that can be considered for different category items, slightly different design for the ListView items for each category like a different background shade, relocation of views, etc. Be creative here.

Make sure that you properly watch Android Lesson 14 lecture to see how we addressed this requirement in the Learn Māori app.

27. Can we use a glide dependency to get images from the internet?
    Yes, this is acceptable.

28. 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.

29. 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.

30. 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 setup 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.

31. Do we have to follow the same structure as the design doc for the quality report?
    No, while the Design Doc has a specific structure, your Quality Report is flexible. You have complete flexibility to deliver your narrative addressing the requirements set out for the Quality Report. Use any structure you feel best.

32. 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 from 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.