PGR208 Android Programming exam
Autumn 2023

Practical information:

- This exam can be delivered as individual students or as a group of up to 3 students.
- The delivered project folder must be compressed (zipped) before being uploaded to exam system WISEFlow.
- You are allowed to discuss the exam with the teacher and TAs during the exam period, but should not expect to get any direct solutions to problems, but rather hints/guidance/tips on how to best proceed.
- Keep in mind that extra functionality in addition to the minimum requirements is expected for groups of two or more students. This is outlined further in the “app requirements”-section.
- The last page of this document contains an assessment guide outlining how the parts in the submitted exam are weighed in the total evaluation of the exam.

Report/documentation:

You should submit a short report describing and documenting your implementation (UI elements, code, frameworks, etc.) and techniques you have chosen to use in your app.
The report is expected to contain between 500 - 1500 words.

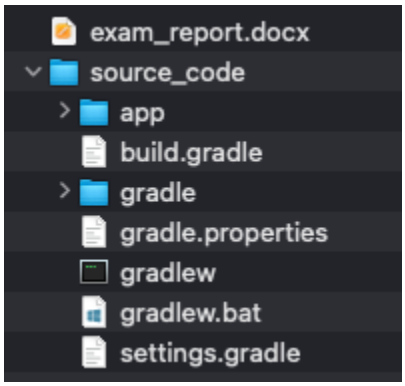
This report is a tool for you to - in addition to the code and comments therein - indirectly explain your implementation and thought process(es) throughout the exam work to the assessor.

Reflect on your decisions:

- What would you do differently a second time around?
- What did you spend a lot of time on?
- What part of the delivery are you most satisfied with?
- Etc..

Submission:

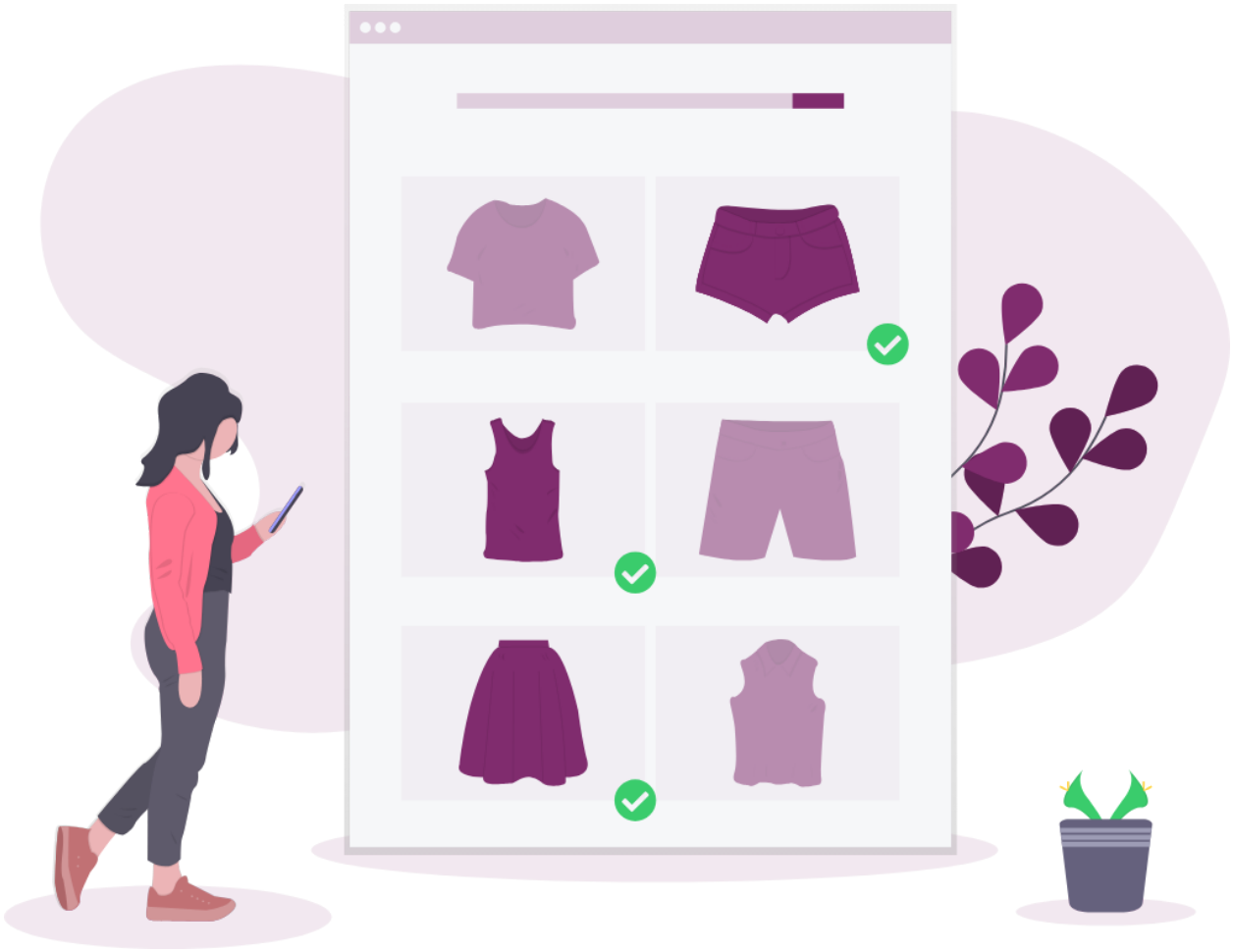
You should submit your source code, report and any other necessary documentation in a .zip file.
The delivery (contents of the .zip file) should be organized in the following manner:



NOTE: The submission should contain only the files listed above. Any other files/folders within the project-folder are not required in the delivery (locally cached/intermediate files).

Case:

You've been approached by a startup that offers various design products. They have an existing online platform, but given the increasing mobile usage trends, they want to expand their reach by introducing a mobile application.
You've been given the task of developing a **prototype** of their new app which will use the **Fake Store API**, tailored to the customer's needs and preferences. The app should offer a seamless experience from browsing products to checking out and viewing previous orders.



App requirements:

Below is an example overview of the different screens that the app is expected to contain, along with minimum requirements and an example UI for each screen.

On top of the minimum requirements, additional functionality/logic/UI is encouraged for all students, but is ***expected*** for groups of two or more students.
You will find some suggestions to additional logic/functionality below, but you are encouraged to come up with your own ideas too.

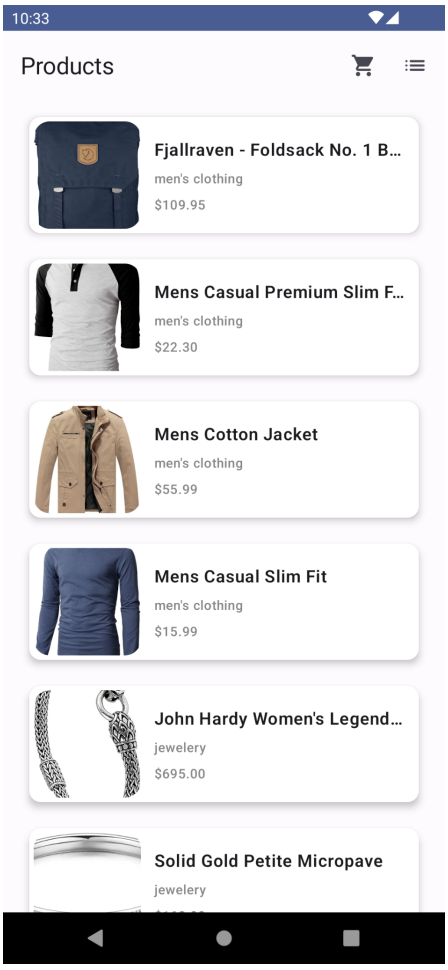
Screen #1 - Product Overview:

Minimum requirements:

- A showcase of all available products.
- Should be scrollable in case list/grid of products overflow the screen boundaries.
- Each product displayed should contain a thumbnail image, product name (“title”) and the price of the product.
- A way to navigate to the Shopping Cart-, Order History- and Product Details screens.

Suggested additional functionality:

- Filter products by category
- Search through list of products
- Make it possible for the user to also browse products while not connected to the internet (offline)



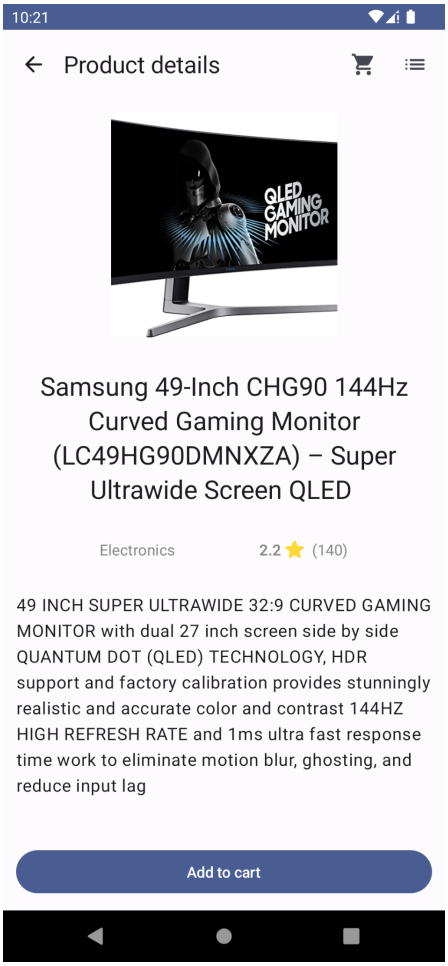
Screen #2 - Product Details:

Minimum requirements:

- A detailed view of the product selected in Screen #1.
- Should provide an in-depth description and other vital information about the product, along with a larger image and an option to add the product to the cart.
- This screen should provide a way to navigate back to Screen #1.

Suggested additional functionality:

- *Leave a rating/comment/review for the product*
- *Add product as a "favorite"*



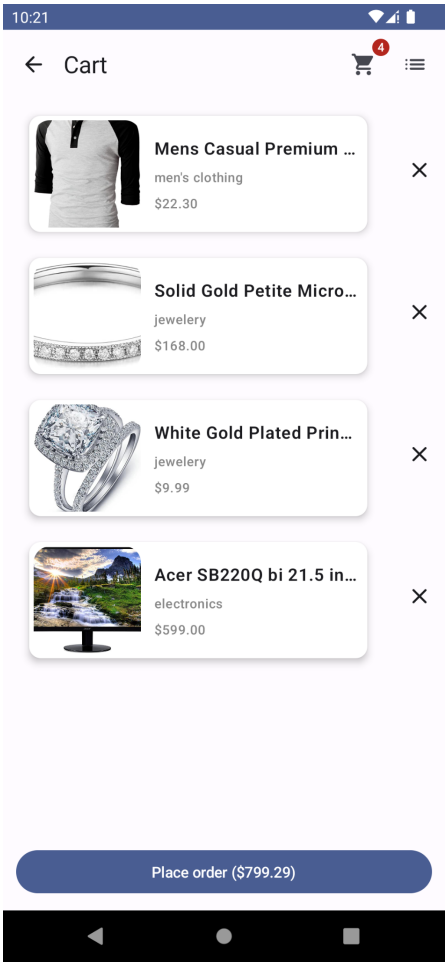
#3 - Shopping Cart:

Minimum requirements:

- A collection of all items that the user has currently added to the cart.
- The user should be able to remove items from the cart.
- The total price of all products in the cart should be clearly visible.
- The screen should provide a way for the user to purchase the items currently in the cart (i.e. “Proceed to checkout”, “Place order”, etc.)
- This screen should provide a way to navigate back to Screen #1.

Suggested additional functionality:

- Let the user adjust the quantity of each product added to the cart.



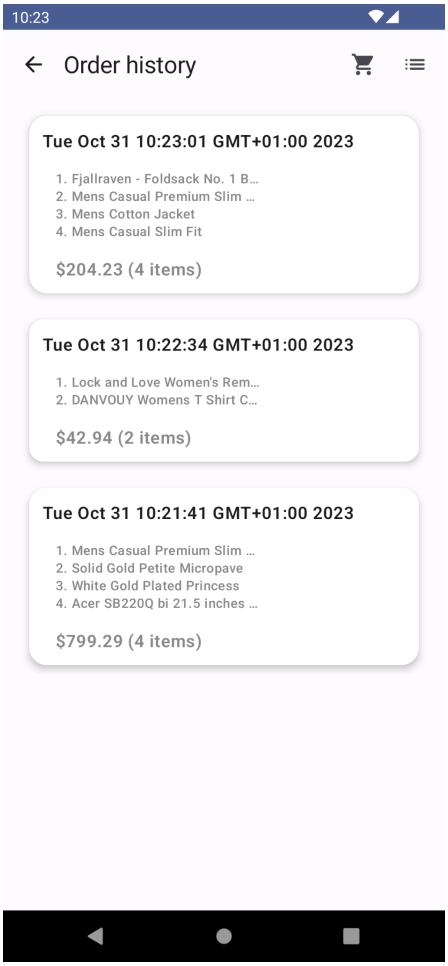
#4 - Order History screen:

Minimum requirements:

- A list of previous orders made by the user.
- This screen should provide a way to navigate back to Screen #1.

Suggested additional functionality:

- Let the user see more details about an order – i.e. navigate to a new Order History Details screen that displays all products contained in the order, price, date of order, etc.



Expected functionality/techniques:

- Use Kotlin as the primary programming language, Android Studio as your IDE.
- Make HTTP requests to the FakeStoreAPI to retrieve products available in the store.
    - https://fakestoreapi.com/products
- Products added to the cart and the user's order history data should be stored in a local database.
- All screens should retain their data through configuration changes (i.e. device rotation)
- Requirements for each screen given in the "app requirements"-section are the minimum requirements for the logic/functionality along with example images of the UI for each screen. Your app/implementation may incorporate different designs and added functionality of your choosing.
- We expect you to implement good code structure/tidiness and good use of techniques learnt throughout the course.

Tips:

- Plan out your app (screens, UI, code/file structure, etc.) ahead of developing it. This will help you keep a good overview of your solution.
- Talk with other students about solutions you've made or problems you've encountered. Even if you're submitting the exam as an individual student (not as a group) reviewing, talking about and explaining your solutions and thought processes to others will help a lot!
- When ready to deliver your submission, it is highly recommended that you try to open the .zip file with your delivery yourself, and see if you can open/run the project. This is to make sure that you have all necessary files added to your submission.
- ***Q:*** *"I'm working alone. Will I get an A by doing only the minimum requirements?"*
  A: This is very subjective and based on the solution's overall complexity, code quality, structure, etc.
  However, in theory, a very high quality implementation of only the minimum requirements with very well written and structured code containing all expected functionality and techniques, as well as a well documented report where you clearly show your skills and knowledge with the concepts used (i.e. a "perfect" solution/implementation), **should** land you an A.

Evaluation/assessment guide:

This assessment guide is for students, internal assessors and external assessors. The percentages shown are approximations of the weight they have in the exam. Depending on the extra functionality added, the weight of the points may vary some.

Code structure, tidiness, good use of techniques learnt throughout the course, etc. is an integral part of the overall assessment and is integrated into the weighting of the points below.

The subjective “beauty”/visual design of the app in itself (“beautiful design”) is not something that will be assessed in this exam. The UI/UX point below is limited only to how the user-interface and user-experience of the app facilitates good interaction methods and visual information for the end-user of the app.

	Note	Weight %
Screen #1		15%
Screen #2		10%
Screen #3		15%
Screen #4		10%
HTTP / API	General implementation and usage of HTTP to retrieve data from the provided third-party API	20%
Local storage	General implementation and usage of storing data locally (database/file)	15%
UI/UX	Overall UI and user experience implementation (i.e. error handling)	5%
Navigation between screens	Navigation and passing of data between screens where necessary	5%
Report/documentation	Self reflection, documentation	5%