Coursework Spec Explanatory Notes:

App proposal

- Propose an app of suitable scale and complexity, with reasonable functionalities
- Keep the proposal as precise and concise as possible, if 1 page is enough, don't use 2!
- E.g. 1 paragraph explaining what the app does and what problem it solves, followed by a list of functionalities in bullet points, name which technical requirement a functionality meets in bracket, e.g. functionality xxx (use permissions)

Mock screens

- Be clear on the design and clear on navigation (what user interaction on a screen navigates to which other screen)

App Development

- Build the app from scratch
- 3rd party API is allowed but there should be enough of your own work! (3rd party API must be clearly referenced in code, should also explain in comments how it is used so that it's clear how much is your own work),
- Identify and apply best practices as much as you can

Meeting the technical requirements:

- Having a minimum of 2 distinct screens
    - They should be screens that allows some user interaction. E.g. Splash screen doesn't count.
    - They should have distinctly different layouts. If two screens have the same layout and can be done using the same composable function with different input parameters, then they count as one distinct screen.
- Use the navigation component to move between screens in your app.
    - Use NavHost, NavController, as necessary for navigation, not the explicit Intent
- Use Intent to move to an outside app.
    - Implicit Intent, E.g. show something on a map, uses the camera, etc.
    - You don't need permission for this! The system app has the necessary permission.
- Work properly with the app lifecycle (including rotate screen changes)
    - This means you need to override the lifecycle methods as necessary, save and restore UI state during the lifecycle methods appropriately, or use lifecycle-aware UI components
    - If data disappears after a screen rotation, then you have not met this requirement. Typical examples include: at screen rotation, input text may disappear, UI component that are dynamically added may disappear, timer may get reset, etc.
    - Locking the screen rotation is not the solution!
- Use permissions and use them responsibly.
    - This doesn't mean to simply request any permissions that you can think of.
    - You should not request any permission that your app doesn't need.
    - You should have a justifiable need to use permission, e.g. need permission to access the Internet for server-side API data, show notification, access location etc.

    - You should use it responsibly (handle runtime permission at appropriate time, show permission rationale, ethical and convenient use of it, consider the option to revoke permission)
    - When the user doesn't give permission, your app should still function (at least partially function).
- Make use of local storage.
    - Choosing the appropriate type of local storage according to the need, e.g. key-value pairs, database, or files
- Create and use your own ContentProvider
    - Creating your own class that extends ContentProvider, providing data to other apps.

Optional requirements:

- Implement your own Service
    - This means creating your own class that extends Service, not using the existing services.
- Capture touch gestures and make reasonable use of them
    - There should be a justifiable need to use the touch gesture. Not the standard onClick, but e.g. long press, scroll, drag, swipe, fling, multi-touch gestures.
    - This means capturing and handling gestures yourself. If the touch event is captured and handled by the UI component, not by you, then it doesn't meet this requirement. E.g. LazyColumn handles the scroll gesture, so it doesn't count.

Code style and readability tips:

- Use meaningful names and consistent naming conventions, indentation, formatting
- Clear and concise comment throughout your code
- Avoid duplicate code, hard-coding, consider modularity, reusability and organisation of code

Quality and smoothness of user interface:

- More on how well users can use your app to complete the tasks that it's intended to, less on artistic skills.
- The app should run smoothly, be robust. The UI should be easy to understand and navigate. E.g. You will lose marks if the app crashes, has performance issues, bugs or errors, or the UI is confusing and difficult to navigate.

Demo

It is entirely up to you whether to use a few presentation slides to introduce your app at the beginning and provide reflective comments at the end or just talk about it.

Demo of app functionalities can be done using a virtual device (or screen capture on a real device if you prefer).

Overview of implementation can include how you met the technical requirements, or other functionalities that you are proud of. Use evidence in your code to support your statements.

Reflective comments can include any changes made to proposal/implementation and justification of changes, any best practices applied in implementation, any lessons learned from testing and debugging, any critical analysis of your app development experience and how you would do things differently if there is time.