# Ampol Digital - Mobile Technical

## 2 Hours

Welcome to the technical take home exercise for the mobile team at Ampol. Our aim of this exercise is to observe how you approach real-world coding challenges and allow you to demonstrate your style, syntax and method in a practical setting. We want to give you the freedom to showcase your ability throughout this exercise, whilst not necessarily prescribing you to set frameworks or implementations, and we encourage you to use your own judgement and creativity to solve the given problem.

Most importantly, we want this task to be enjoyable. We love what we work on, and are excited by the challenges we tackle every day at Ampol. We understand that a take home exercise can take a lot of time out of a busy schedule, so we ask that you complete only as much as you are comfortable with. Consider time boxing the exercise to around 2 hours.

Best of luck, and we look forward to seeing your work!

### Objective

With your selection of mobile frontend (Native iOS or Android), your assignment is to implement a minimal dashboard interface to serve as the entry point for the three Ampol products: *FuelPay, AmpCharge, and Ampol Energy.*

### Brief

Ampol operates across a range of business streams, including fuel, convenience, and electricity. To provide customers with a seamless experience across these product offerings, Ampol is looking to create a dashboard that combines these offerings into a single app experience.

For the purpose of this take home, you have the creative freedom to choose how you approach the following requirements, completing as many or as few as you would like. Whether implementing a single subset in detail, or completing multiple in less detail.

**Requirements**

- The app's main page should be a dashboard that displays key information to the user in the form of blocks or modules. Examples of this can include:
    - Most recent fuel transactions
    - Recent charge sessions
    - In store offers
    - Upcoming Energy Bills
    - Home energy usage
- Blocks or modules should show high level detail that the user might want immediately, whilst giving them the option to navigate into more detail. Examples of this can include:
    - Recent fuel transactions shows only the 2 most recent transactions. A "See all" button on the module could take the user to a detail view with all their previous transactions.
    - Recent charge session shows only the location and price of the charge, tapping into the module could take the user to a detail view that shows further details about their charge including but not limited to:
        - Price
        - Amount of time and Charge rate (kWh)
        - Discount
    - In store offers could show a targeted offer, whilst tapping into the module takes them to show all offers available.
- With the goal of creating a seamless experience for 3 products in a single application, blocks and modules should have complementary design and colours, and should communicate a single experience across the differing products.
- Upload your source code to a public repo and provide a link to the exercise, please include how long you spent on this exercise.

**Tips**

- Your code does not have to be production ready, a good implementation of the requirements will demonstrate how you develop in the real world.
- You have creative freedom in how you design and colour the app. You may focus more or less time on this aspect to your liking,
- Code the way you like to code.
- Take a few notes about decisions you make along the way, so that you may talk about them briefly in the technical interview.
- You don't have to cater to all screen sizes for this exercise, stick to your favourite and just let us know which one, so we can view it the way you intended!
- We like to use some of the latest technologies in the mobile world at Ampol. So feel free to target a minimum version of iOS 15 or Android 12 in your implementation.
- Networking is not required, all data can be mocked locally, or handled however you see fit.
- We love code that is clean, readable, maintainable and performant.

**Additional Guidelines**

If your solution requires any external dependencies or setup to run, please include a README on how to get it up and running. (Here is where you can include preferred devices to run it on, if you have any!)

**A little bit about our stack**

**iOS**

- Pure SwiftUI 3+ (iOS 15 minimum target) application
- Combine + Swift Modern Concurrency (async await)
- SwiftUI application lifecycle
- Swift Package Manager
- In house design system and component library
- Functional design and immutable state

**Android**

- Jetpack Compose for all new functionality
- Legacy code in XML
- In house design system and component library
- Dagger 2
- Retrofit
- Flows + Coroutines