Justification of Task Hawk Architecture

- Jacob Thieret

Description of overall architecture

The chosen architecture for our project is the Flutter framework using Dart. It is a Google backed, widget-based architecture, so it is very modular, and many portions of code are reusable. It provides a very well supported environment for mobile applications. This architecture is optimized for maintainability, performance, and a built-in consistent user experience across different platforms.

1. Justify the chosen architecture over alternatives.
    a. Justification of Flutter with Dart over Kotlin or Java:
        – Official support. Dart is the primary language supported by the Flutter team, Kotlin and Java are not. Means more updates, better documentation, and better support from the community.

        – Hot Reload. Quick development times due to developers real time updates to the changes in the code base. Kotlin and Java also allow this, but it is more prone to errors.

        – Ahead-of-Time (AOT) and Just-in-Time(JIT) compilation. Dart supports AOT and JIT, which may provide slight performance in a production environment. While Kotlin and Java are based on the Java Virtual Machine(JVM), and may not be at the same level of optimization

            1. AOT Advantages. Faster startup time. Compiled code can be optimized for the target platform during the building process. Smaller memory footprint because it does not require a runtime compiler. Faster component rendering. Basically better performance during runtime, because less work is being done in that step.

            2. JIT Advantages. Changes to the code don't require recompilation of the entire application, making debugging a much more efficient process. JIT compilers can also optimize the code based on runtime behavior.

        – Single codebase for IOS and Android. This framework comes with built-in cross compatibility. That's pretty nice... It is more challenging to implement this in Kotlin and Java.

        – Dart's direct integration with Flutter. It's the default language so there aren't any extra steps to getting started coding, unlike Kotlin and Java which require extra tools/bridges to get started.

- Large Community. Since it's the default language, it's generally going to be very well documented with a very large support community.

b. Justification of Flutter with Dart over React Native

- Arguably better performance. Flutter uses the Skia Graphics Engine to draw UI components directly on the screen, whereas React Native relies on a JavaScript bridge to communicate with native components, which can lead to performance bottlenecks.
- Consistent UI across both IOS and Android. Built-in cross compatibility for UI and internal interactions. Flutter uses its own set of widgets to create the UI, which ensures consistency across Android and iOS.This is different from React Native, which relies on native UI components and may result in platform-specific UI differences.
- Google backing. Flutter is backed by Google, which provides excellent support and updates to the framework.
- Strong typing and language features. Dart is a statically typed language with strong typing and advanced language features like null safety and asynchronous programming support. React Native uses JavaScript, which is dynamically typed and might not offer the same level of robustness.

c. Justification of Visual Studio Code over Android Studio

- Visual Studio Code is lightweight and has fast performance, unlike Android Studio which can consume more resources
- Cross platform development is better supported in VSCode, however Android Studios is primarily focused on Android development.
- Git support, integrated terminal, intuitive UI, thousands of extensions, advanced debugging capabilities, etc. It’s a much more versatile option over android studios.

2. Interesting technology choices

a. ‘getX’ Flutter package. What does it do?

- Automatically manages the state of widgets, updating them when necessary. Reduces the need for boilerplate and makes state management infinitely easier.
- Can navigate to new pages, return to previous pages, and pass data between pages with minimal code.
- Has utilities for form validation, making it easier to build robust localized applications like ours, and reducing the amount of manual coding needed

greatly. We will have a couple forms in our application and have the form information be validated with minimal setup if very nice.

- It is designed to be highly efficient and consumes fewer resources than other options, which may allow the application to scale more effectively in the future.

b. ‘table_calendar’ Flutter package. What does it do?

- Provides a highly customizable calendar widget that can be used to display, navigate, and manipulate dates.
- We will use it to implement a calendar view that displays dots for each task due on that day.
    1. Allows you to associate events with specific dates, making it easy to display and manage tasks within in the calendar
    2. Also has event handling, making it easy to handle user interactions like selecting a date, tapping a task, or swiping to navigate months.
- Has a very customizable UI with support for customization of colors, text styles, and layout. It also has different calendar formats like month and week views.

3. Other technologies used:

a. Android Studios Android Virtual Device(AVD) Manager.

- Allows us to develop on a virtual device as opposed to needing a physical device.
- Some of the team has iPhone and use windows and those two do not play well together. Since none of us are using a Mac, it made sense for us to use an Android emulator.
- Visual studio code allows for integration with the virtual devices in the AVD, allowing us to start the entire system, emulator and all with one click.

b. Google Drive(imports files)

- For the external importing of files, we may end up using google drive because it’s a Google service and will likely be well supported in our environment. This is not yet certain.