### Project Phase 2: App Architecture

We're developing the iOS and Android versions of the Uber Eats app. With features including secure payment gateways, real-time order monitoring, location tracking, and interaction with restaurant inventory systems, the app aims to make ordering easier for users.

**Decision 1: UI Framework**

**Context:**

Given its influence on user engagement and happiness, the user interface (UI) of the Uber Eats app is critical to its success. Selecting a user interface framework is essential to ensuring a uniform and user-friendly experience on both the iOS and Android platforms.

**Options were considered:**

* Cross-platform user interface frameworks, like Flutter and React Native.
* To design iOS and Android platform-specific user interfaces, think about utilising Stylesheet API.
* An iOS and Android stylesheet API.

**Decision:** Select Stylesheet API for both iOS and android to create platform-specific user interfaces.

**Rationale:**

**Consistency Across Platforms:** Although Stylesheet API are platform-specific, they provide for uniformity in Android and iOS UI design and behaviour.

**Developer Productivity:** By providing a declarative and quick way to create user interfaces, the existing UI frameworks improve developer productivity and speed up feature delivery.

**Effective Maintenance:** Maintenance duties can be rendered faster with a standard Stylesheet API for both iOS and Android. The user interface (UI) can be changed or updated more easily, reducing the work required for developers and guaranteeing a simplified and consistent user experience across platforms.

**Consequences:**

**Platform Consistency:** The look and feel of an app may vary between iOS versions even if Stylesheet API allows for platform-specific UI designs.

**Learning Curve:** It could take some time for developers to get comfortable with Stylesheet API, which could have an impact on their early output.

**Adaptability Issues:** Stylesheet API may have issues changing to updates or changes made to the iOS and Android operating systems in the future. This can cause delays in installing new updated features or adjustments, which would make it harder for the app to maintain its advanced user experience.

**Follow-up actions:**

**Performance Monitoring:** To identify opportunities for UI enhancement, monitor user feedback and app performance on a regular basis.

**User Interface Consistency standards:** Create UI design standards to ensure uniformity across platforms while using each framework's special characteristics.

**Continuous Developer Training:** Provide training to ensure developers are proficient in utilizing the chosen UI framework effectively and can make improvements using the same framework.

**Decision 2: Permissions**

**Context:** To perform crucial features like push notifications and location tracking, the Uber Eats app needs access to certain device permissions. We need to decide about permission management so that user privacy and programme functionality can coexist in harmony.

**Options were considered:**

• Make use of the default permissions without requesting user input.

• Programme features provide the basis for requests for runtime permissions.

• Make sure you seek all rights during installation.

**Decision:** Implement runtime permissions based on app features and user activities.

**Rationale:**

**Granular Access:** We can provide a more effective and tailored experience while minimising unnecessary permissions by requesting access based on app features and user behaviours.

**Compliance:** Adhering to platform-specific authorization requirements ensures that you are compliant with the rules and standards of the app store, which lessens the likelihood of being denied or suspended.

**Efficient Utilization:** By getting permissions only when necessary, runtime permissions improve the use of available resources.

**Assurance of Compliance:** App store compliance is ensured by following platform-specific permission guidelines.

**Consequences:**

**User Trust:** Appropriate handling can lead to user frustration and mistrust, even though clearly outlining the requirement for rights and requesting them at runtime may boost user trust.

**User Engagement**: By the example of user-centric plans, appropriately planned and context-aware runtime permission requests can positively enhance user engagement and possibly boost user satisfaction and retention.

**Follow-up Actions:**

**Regular Privacy Tests:** To maintain a balance between functionality and privacy, do regular privacy checks to make sure that runtime permission requests meet changing app features and user behaviors.

**Feature-oriented Permission Optimization:** To improve the user experience, regularly test the features of the app to improve and optimize runtime permission requests. Make sure that these requests match the most appropriate and essential features.

**Platform Updates:** Keep up with permissions updates that affect your platform and implement any upgrades into your app to ensure ongoing security and the present user experience.

**Decision 3:** Hybrid Mobile Application

**Context:** We need to choose between developing the Uber Eats app as a web application or as a hybrid app for iOS and Android in order to ensure a seamless and responsive user experience.

**Options Considered:**

• Create apps for iOS and Android platforms.

• Only iOS devices—iPhone and iPad—are the target.

• Exclusively focus on Android smartphones and tablets.

**Decision:** Develop hybrid mobile apps for iOS and Android.

**Rationale:**

**Consistent User Experience:** We can offer a consistent user experience on both platforms by developing for both iOS and Android. We guarantee that consumers, irrespective of the device they are using, get a consistent and familiar experience by adhering to platform-specific design principles and best practices.

**User Preferences:** Depending on things like brand loyalty, ecosystem familiarity, and device price, users have different preferences for iOS and Android devices. We accommodate user choices by supporting both platforms, which raises user satisfaction and engagement.

**Market Share:** Around the world, both iOS and Android have substantial market shares. By developing for both platforms, we can take advantage of their sizable user populations and use the popularity of both to expand the app's market reach and adoption potential.

**Improving Market Reach:** Developing hybrid mobile applications that run on both iOS and Android guarantees a wider audience by using the huge user populations of both operating systems. This strategy maximizes the possibility of user acceptance and engagement by taking advantage of the broad adoption of iOS and Android smartphones worldwide.

**Consequences:**

**Resource Requirements:** More labour, money, and time are needed for cross-platform development and maintenance. Managing multiple platforms requires resources for testing, debugging, and optimising the programme for each device, as well as skilled engineers with knowledge in both iOS and Android development.

**Greater Development Efforts:** Compared to focusing on a single platform, developing for both iOS and Android requires a greater investment in development and testing resources. The development process is complicated by the need to manage two codebases, fix defects unique to each platform, and ensure device interoperability.

**Problems with Version Syncing:** It might be difficult to plan parallel upgrades and feature launches for the iOS and Android platforms. Ensuring consistency in app versions and similar features demands thorough planning and sync efforts.

**User Fragmentation Considerations:** Variations in hardware and screen sizes provide difficulties because of the wide variety of devices in the Android ecosystem. More testing and optimization work is needed to address this fragmentation in the app experience.

**Follow-Up Actions:**

**Cross-Platform Development Tools:** To expedite development and facilitate code interchange between iOS and Android, use cross-platform development frameworks or libraries like React Native, Flutter, or Xamarin. Assess the team's skill level and the needs of the project to determine which tool is ideal.

**Platform-Specific aspects:** To guarantee the best possible performance and user experience on iOS and Android devices, identify and take care of platform-specific design and development elements. This entails optimising device-specific functionality, navigation patterns, and UI/UX features.

**Continuous Performance Monitoring:** Use tools for tracking the reactivity, loading speed, and general user experience of the app on iOS and Android. These regular evaluations provide the best possible app performance and assist in identifying areas for development.

**Input from users Integration**: Create a simplified procedure specific to each platform for collecting and assessing user input. This user-centric method can reveal platform-specific problems or preferences, guiding the development of future updates and improvements to the needs of users on both iOS and Android.