|  |  |
| --- | --- |
| Team Name: App Masters | Date: 2024-02-11 |
| Team Members: Rohan, Harshit, Jagdeep | Reporting Period: Phase 2 |

### Project Phase 2: App Architecture

We are creating the Uber Eats app for iOS and Android. The app's purpose is to provide customers with a smoother ordering experience by including features like location tracking, real-time order tracking, secure payment gateways, and integration with restaurant inventory systems.

**Decision 1:** Hybrid Mobile Application

**Context:**

To provide a smooth and responsive user experience, we must decide whether to create the Uber Eats app as a hybrid app for iOS and Android, or web development.

**Options Considered**

* Target only iOS devices (iPhone and iPad).
* Target only Android devices (phones and tablets).
* Develop for both iOS and Android devices.

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

**Rationale:**

* **Maximise the User Base:** Developing for both iOS and Android smartphones guarantees that the Uber Eats app reaches the largest possible audience. By targeting two major platforms, we can cater to consumers regardless of their device preferences, hence growing the app's user base.
* **User Preferences:** Users' choices for iOS and Android devices vary depending on factors like as brand loyalty, ecosystem familiarity, and device price. By supporting both platforms, we cater to users' preferences, increasing user happiness and engagement.
* **Market Share:** Both iOS and Android have sizable market shares worldwide. By building for both platforms, we can tap into these vast user bases and capitalise on each platform's popularity, increasing the app's market reach and adoption potential.
* **Consistent User Experience:** Developing for both iOS and Android allows us to provide a uniform user experience across both platforms. By following platform-specific design principles and best practices, we ensure that users have a consistent and familiar experience regardless of the device they are using.

**Consequences:**

* **Increased Development Efforts:** Developing for both iOS and Android necessitates more development and testing resources than targeting a single platform. The requirement to maintain two codebases, address platform-specific bugs, and assure device interoperability adds complexity to the development process.
* **Resource Requirements:** Cross-platform development and maintenance require more resources, such as labour, time, and budget. Managing multiple platforms necessitates competent engineers with experience in both iOS and Android development, as well as resources for testing, debugging, and optimising the programme for each platform.

**Follow-Up Actions:**

* **Platform-Specific factors:** Determine and address platform-specific design and development factors to ensure optimal performance and user experience on iOS and Android devices. This involves optimising UI/UX elements, navigation patterns, and device-specific functionality.
* **Cross-Platform Development Tools:** Use cross-platform development frameworks or libraries like React Native, Flutter, or Xamarin to speed development and enable code exchange across iOS and Android. Determine and choose the best tool based on project needs and team competence.
* **Continuous Testing and Optimisation:** Regularly test and optimise the Uber Eats app for iOS and Android to detect and resolve any performance, compatibility, or bug issues. Implement a comprehensive testing plan that includes device testing, beta testing, and user feedback collection to ensure a high-quality user experience across all supported devices.

**Decision 2: UI Framework**

**Context:**

The user interface (UI) is crucial to the Uber Eats app's success, as it influences user engagement and satisfaction. We must choose a UI framework that will allow us to give a consistent and intuitive experience across both the iOS and Android platforms.

**Options were considered:**

* Stylesheet API for both iOS and android.
* Cross-platform UI frameworks, such as React Native and Flutter.
* Consider using Stylesheet API to create platform-specific user interfaces for both iOS and android.

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

**Rationale:**

* **Native Integration:** Stylesheet API integrate seamlessly with their respective platforms, providing access to platform-specific UI components and optimisations.
* **Developer Productivity:** These current UI frameworks offer a declarative and fast method for creating user interfaces, increasing developer productivity and feature delivery.
* **Consistency Across Platforms:** While Stylesheet API are platform-specific, they enable consistency in UI design and behaviour across iOS and android.

**Consequences:**

* **Learning Curve:** Developers may need time to become acquainted with Stylesheet API, which may influence their early productivity.
* **Platform Consistency:** While Stylesheet API allow for platform-specific UI designs, the app's appearance and feel may differ between iOS versions.
* **Performance:** Modern UI frameworks are performance-optimized, which leads to smoother user interfaces and faster app response.

**Follow-up actions:**

* **Training and Education:** Provide training tools and workshops to assist developers in learning Stylesheet API efficiently.
* **User Interface Consistency Guidelines:** Create UI design guidelines to ensure uniformity across platforms while exploiting each framework's specific features.
* **Performance Monitoring:** Continuously analyse app performance and user comments to find areas for UI improvement.

**Decision 4: Permissions**

**Context:** The Uber Eats app requires access to device permissions for important functions like location tracking and push notifications. To strike a balance between user privacy and programme functionality, we must decide how permissions will be handled.

**Options were considered:**

* Runtime permission requests are made based on programme features.
* During installation, make sure you request all rights.
* Use default permissions without requiring user intervention.

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

**Rationale:**

**User Control:** Requesting permissions at runtime allows consumers to have more control over their data and privacy, leading to increased trust and openness.

**Granular Access:** By requesting rights based on app features and user activities, we can give a more personalised and efficient experience while limiting needless permissions.

**Compliance:** Following platform-specific authorization criteria guarantees that you are in accordance with app store standards and regulations, which reduces the chance of rejection or suspension.

**Consequences:**

**User Trust:** While properly explaining the need for permissions and seeking them at runtime may increase user trust, inappropriate handling may result in user annoyance and distrust.

**Security Considerations:** Requesting permissions during runtime.