**Title: Architectural Decisions for Retail Mobile App**

**Date:** [October 17, 2023]

**Context:** We are a development team tasked with creating a mobile app for a retail company. The app's purpose is to enable customers to browse and purchase products, view their order history, track deliveries, and participate in a loyalty program. The app has specific requirements, including offline support, push notifications, payment gateways, user behavior tracking, image handling, and internationalization.

**Decision : App Type - Native**

**Status:** Accepted

**Decision:** We have decided to develop a native mobile app for both iOS and Android platforms.

**Justification:**

**Performance and User Experience:** Native apps outshine web and hybrid counterparts by offering top-notch performance and user experience, courtesy of their in-depth integration with platform-specific APIs and capabilities. Their purpose-built nature allows them to fully exploit the device's hardware and software, resulting in optimized performance, snappy responsiveness, and direct access to device features like the camera, GPS, and accelerometer. This integration enables native apps to craft tailored and unique experiences, setting them apart and ensuring user satisfaction.(<https://developer.apple.com/design/human-interface-guidelines/>,<https://developer.android.com/guide>)

**Access to Platform Features**: Native development provides access to platform-specific features, ensuring a seamless integration of functions like in-app payments and hardware-specific interactions.

**Decision 2: UI Framework**

**Status:** Accepted

**Decision:** We have chosen to implement our user interface using platform-specific UI frameworks: UIKit for iOS and Android's native UI framework.

**Justification:**

**Consistency and Familiarity:** Platform-specific UI frameworks offer users a consistent and familiar interface, enhancing their experience. This approach aligns with platform design guidelines, ensuring users feel comfortable with the app's appearance and navigation.(<https://developer.android.com/develop/ui/views/theming/look-and-feel>)

**Optimized UI Performance:** Utilizing native UI components ensures efficient rendering and smooth interactions, minimizing lag. This approach prioritizes a responsive and high-performing user experience, even under heavy usage or resource-intensive tasks.

**Decision 3: Backend Language**

**Status:** Accepted

**Decision:** We have opted to employ Node.js for server-side logic and Python for data processing and analytics.

**Justification:**

**Real-time Features**: Node.js, with its event-driven, non-blocking architecture, is our choice for handling real-time features like push notifications and online-offline synchronization (<https://nodejs.org/en/docs>).This architecture ensures the efficient handling of concurrent connections and is well-suited for applications that require instant updates and communication with minimal latency.

**Data Processing**: Python, renowned for its rich ecosystem of libraries, is the ideal choice for data processing and analytics (<https://docs.python.org/3/tutorial/datastructures.html>).Its extensive selection of tools for data manipulation and statistical analysis equips us to manage and analyze data effectively, essential for meeting the demands of our application. This combination of Node.js and Python ensures a well-rounded backend setup tailored to our specific requirements.

**Decision 4: Permissions**

**Status:** Accepted

**Decision:** Our approach involves implementing a permission model where users will be asked for specific permissions, such as location access for delivery tracking and push notification permissions.

**Justification:**

**Privacy Compliance**: By requesting these permissions, we demonstrate our commitment to safeguarding user privacy and complying with relevant privacy regulations and platform-specific guidelines (<https://developer.android.com/guide/topics/permissions/overview>, <https://developer.apple.com/documentation/corelocation>).This not only builds trust with users but also ensures that our app adheres to the stringent privacy standards set by governing bodies.

**Service Functionality**: The permissions we request are integral to the functioning of our services. They enable critical features such as location-based tracking and push notifications, enriching the core functionality of the app. By obtaining these permissions, we ensure that users can experience the full range of services we provide, ultimately enhancing their overall experience

**Decision 5: Data Storage**

**Status:** Accepted

**Decision:** Our chosen strategy involves utilizing local storage to facilitate offline access and Amazon S3 for data synchronization when an internet connection becomes available.

**Justification:**

**Offline Access**: The use of local storage is instrumental in providing users with uninterrupted access to essential data, even in offline mode. This ensures a seamless and consistent browsing experience, enhancing user satisfaction (<https://developer.android.com/training/data-storage>).

**Data Synchronization**: The selection of Amazon S3 as our cloud storage solution is grounded in its reputation for reliability and scalability. It guarantees secure and efficient data synchronization, safeguarding data integrity and consistency across devices (<https://aws.amazon.com/s3/>).This approach supports the user in keeping their information up to date and readily accessible when they move between online and offline environments, thereby contributing to a robust and reliable user experience.

**Decision 6: Additional Frameworks and Technology Stacks**

**Status: Accepted**

**Decision**: In our tech stack, we have embraced Firebase as a Backend-as-a-Service (BaaS) solution for push notifications, user analytics, and real-time database functionality.

**Justification:**

**Push Notifications**: Leveraging Firebase Cloud Messaging (FCM) streamlines our push notification implementation and guarantees dependable delivery to both iOS and Android devices (<https://firebase.google.com/docs/cloud-messaging>). This decision not only simplifies our development process but also ensures that critical information reaches users effectively and in a timely manner, enhancing their engagement with the application.

**Real-time Features**: The adoption of Firebase Realtime Database enables real-time synchronization of app data, a pivotal requirement for functionalities like order updates (<https://firebase.google.com/docs/database>).This technology empowers us to deliver dynamic, up-to-the-minute information to users, fostering a more interactive and engaging user experience