Assignment: Architectural Decisions

Student Name:

Ho Pong Chan (Nathan)

Man Ho Cheung (Felix)

Chosen Scenario:

1 - Retail Company

Title

Develop a new mobile app for a retail company.

Status

Approved.

Context

Our team has been invited to develop a new mobile application (app) for a retail company. The goal is to enhance profitability and expand the company's global presence by improving user experiences through the app. The app is designed to offer the following features to customers:

- browse products and view order history without an internet connection.
- purchase products.
- track order delivery status.
- access and participate in the loyalty program to earn and redeem points.

In addition to the functionalities for customers mentioned above, the retail company requires the following functionalities that allow it to collect user data for business analysis and processing payments. The data collected will be used and analyzed to design business strategies for business expansion on an international scale. Data in the app must be synchronized with the server once internet connection is available. The following are the functionalities that the retail company requires.

- notify users about order updates, new product arrivals, exclusive offers, and order delivery status via push notification.
- integrate secured payment gateways.
- track user behavior for business analysis and performance enhancement purposes.
- implement image optimization techniques for product image display by choosing a suitable image storage solution.
- equip database to support multiple languages and cultural preferences.

According to the above requirements given by the retail company, we will develop a mobile app with 6 key components as follows.

i. **Support offline mode**

The retail company wants customers to be able to browse its products and view their order history when the internet is not available. Having the app supported with offline mode is a way to achieve this feature. To ensure the downloaded database is updated, the app will be set to synchronize its database with the server once an internet connection is available.

ii. **Push notification**

The retail company wants customers to be updated about its new product arrival, exclusion offers, order status, and order delivery status. Push notifications will be migrated to this app thus its customers will be notified accordingly.

iii. **Payment gateways**

Different payment gateways will be integrated into this app to allow customers to pay for their orders to facilitate secure and convenient payments. Credit card and third-party payment platform, named PayPal, will be used due to its security, ease of use, and compatibility with this app.

iv. **User behavior tracking**

The retail company wants to collect user behaviors in terms of product views, purchases, loyalty program interaction, etc. to improve app performance, enhance user experience, analyze user data, and design business strategies. To achieve this, the tracking function and analytics tool will be migrated to this app.

v. **Product image display**

Photos of all products need to be displayed to users. Since a higher level of user experience in terms of aesthetics is required, the size and resolution of photos need to be unified for easy browsing and app optimal performance.

vi. Multi-language

Because of the goal of global business expansion, to avoid surfing difficulties associated with language barriers and cultural preferences for a high level of user experience, choices of language will be available in the app.

To build this app, we made 6 architectural decisions that define the technology stack and approach for this app development. These architectural decisions are developed based on the factors of performance, maintainability, and scalability.

Decision 1 - Mobile App Type

Choice: Native Mobile App

Based on the needs of optimal performance, device-specific features, and offline functionality, the native mobile app has been chosen.

Rationale

A native mobile app was chosen due to its ability to deliver optimal performance, leverage device-specific features, and support offline functionality. Native apps are platform-specific, designed explicitly for iOS and Android, enabling users to benefit from platform-specific capabilities, and ensuring smooth offline operation by efficiently managing device resources and data storage.

Consequences

Pros

Because of its optimal performance and device-specific features such as offline data and resource access, users can access the locally stored content of the app in offline mode.

Cons

Since iOS and Android are two different Operating Systems, extra effort and time are required for app development when compared to cross-platform solutions. Maintenance costs may rise when, for instance, fixing bugs and updating the app system on two different platforms.

Decision 2 - UI Framework

Choice: React Native.

React Native allows developers to write code in JavaScript for the interface which is working across both iOS and Android platforms. Multi-language support can be built in within the user interface.

Rationale

React Native can significantly reduce development time and effort due to the ability of code reuse. The result of code updates can be seen in a real-time environment. The built-in component library brings ease and efficiency for developers to build and maintain the user interface with choices of different languages which can be rendered on both iOS and Android platforms. React Native also provides internationalized and localized libraries that enable the app to handle multiple languages, including emojis.

Consequences

Pros

Both code and component library can be used in both iOS and Android Operating Systems which reduces development time and effort. Multi-language support enables the company to attract users from various regions with different language and cultural backgrounds.

Cons

Although React Native offers a wide range of native modules, it may not cover specific platform-specific features or APIs.

Decision 3 - Backend Language

Decision: Node.js.

The choice of Node.js as the backend language can handle real-time updates and integration of various services due to its lightweight and efficient performance.

Rationale

Node.js allows an event-driven, lightweight, and fast execution environment that can efficiently handle concurrent connections and real-time updates such as order status, push notification, data synchronization, analytic tools, etc. Integration of various services such as linking APIs to our app fits the functionalities provided in this app such as the payment system.

Consequences

Pros

Node.js allows JavaScript to be running on both server-side and client-side, enabling full-stack JavaScript development. It can handle asynchronous operations thus several connections can be processed simultaneously. Its lightweight and fast execution environment results in low-latency responses and high throughput.

Cons

Real-time capabilities require thorough design and scaling to handle heavy traffic and loads. Development teams must be extremely proficient in front-end and back-end development knowledge, which could affect team composition.

Decision 4 - Permissions

Decision: Granular permissions system

A granular permissions system will be used in this app ensuring users have control of what data and device features they can access. It also restricts the system from accessing sensitive data of users.

Rationale

Granular permissions allow users to configure the privacy and security level such as enabling the function of a push notification, user of microphone, and location, etc. when required for corresponding tasks. A top priority of privacy and security and permission handling can be guaranteed and practiced, respectively. This system upholds user control.

Consequences

Pros

Protecting user privacy enhances user trust and confidence by implementing a granular permissions system.

Cons

This system adds complexity to the development process. The development of a granular permissions system requires careful system building otherwise incorrect configuration may lead to privacy and security risks and even leakage. If the restrictions are overly strict, it limits the app's functionality which may discourage users from using the app.

Decision 5 - Data Storage

Decision: SQLite for local offline storage and API for server-side data.

Using SQLite for local offline storage and API for service-side data balances offline functionality with real-time data updates.

Rationale

A lightweight and efficient database engine can be developed by SQLite for product data and order storage locally. Data consistency can be ensured by using APIs for server-side data storage. Hybrid data storage solutions will be implemented to support both offline and online mode operations. Synchronizing data with the server will be completed instantly when the internet is available for the app to connect to.

Consequences

Pros

SQLite is a self-contained technology that does not rely on a separate service process that supports offline mode operation. It does not require complex setup or configuration while being capable of working across different platforms. Using APIs allows the native app to access external services, data sources, and functionalities without a complex setup. It also enables real-time data updates.

Cons

Data synchronization between local SQLite database and remote APIs increases system complexity which requires thorough design to avoid incorrect configuration. Using APIs limited user controls (both the company and customers) as those are owned and operated by other third parties. Unexpected service discontinuation and/or limit adjustments and updates can significantly impact the app's behavior and performance which lowers user experience and satisfaction. Storage capacity is solely dependent on users' devices.

Decision 6 - Additional Frameworks or Technology Stacks

Decision: Firebase for push notification, payment gateway, analytics integration.

Image Optimization by using the format of JPEG.

Firebase provides a robust set of tools and services such as push notifications, payment gateway integration, and analytics. The image format of JPEG is chosen for image optimization.

Rationale

Firebase is a comprehensive tool to easily migrate different functions (including analytics, message real-time delivery, and payment) into the app that streamlines common mobile app features resulting in a reduction of development effort and time. Analytic tools monitor user behavior and app performance and help make data-driven decisions by having insights from user data. The format of JPEG maintains images in high resolution with a lower file size. With the use of JPEG image format, images can be optimized for browsing, loading, and storage.

Consequences

Pros

Push notification, analytics, and payment gateway provided by Firebase can work across different platforms resulting in implementation simplicity. Firebase's security features help protect sensitive payment information by using data encryption and compliance with Payment Card Industry Data Security Standard requirements [1]. Analysis generated by analytic tools provides detailed insights into user behavior and app usage etc. help fine-tune and further develop the app. Image optimization leads to higher loading and storage efficiency that reduces the load on both user devices and servers while maintaining relatively high image resolution. A better browsing experience can be delivered to users, especially where bandwidth and storage constraints occur.

Cons

The cost of using Firebase can be very high if the usage volume such as frequent payment processing is large. Since event tracking is enabled in the app (when the user enables this function), improper handling of user data can result in legal and reputation issues. Clear, logical, and strict data governance is solely dependent on how the company designs it. If improper government is set, user's privacy and data cannot be safeguarded. JPEG does not support lossless compression. If the image is heavily compressed, the loss of quality is noticeable.

Conclusion

In this Architectural Decision Report, the essential choices guiding the development of a new mobile app for a retail company are laid out. These decisions include the app's functionality and technology stack which align with the project's goals.

We have chosen a native mobile app that prioritizes optimal performance and offline functionality. React Native will be used for the UI framework which runs across multiple platforms. Node.js is selected for back-end technology because of the advantages of its real-time capabilities and service integration. A granular permissions system can protect sensitive data security while leaving control to the user. A hybrid approach of using both SQLite for local storage and APIs for server-side data ensures seamless data synchronization.

Firebase handles and processes push notifications, payment gateway integration, and analytics. Image Optimization associated with using the JPEG image format helps streamline photo processing and loading which enhances user experience. With all the decisions made in this report, the app is correctly positioned for the retail company to meet its goal of global business expansion and profitability. The main objective achieved by this app is to enhance the user experience. With this positive impact, revenue can be pushed up to a higher level which increases financial assets to expand its business internationally.

References

1 N. Barney. “PCI DSS (Payment Card Industry Data Security Standard.” TechTarget. Accessed: October 7, 2023. [Online.] Available: https://www.techtarget.com/searchsecurity/definition/PCI-DSS-Payment-Card-Industry-Data-Security-Standard#:~:text=The%20primary%20goal%20of%20PCI,breaches%2C%20fraud%20and%20identity%20theft.