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 from the web and hybrid app.

Rationale

Our primary target users are using both the main platforms iOS and Android. A native mobile app can 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

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. Users can notice the app's high polish, responsiveness, and smooth interactions which enhance user experience.

Decision 2 - UI Framework

Choice: React Native.

A choice of React Native fits the development of this app rather than Xamarin.

Rationale

React Native uses JavaScript and React to build platform-specific UI components while Xamarin uses C# and the .NET framework. Both allow developers to write a single codebase for the interface working across both iOS and Android platforms. According to [1], the market share of using React Native is 40% compared to 11% of Xamarin. In addition, Xamarin will be replaced by .NET MAUI in 2024 which requires developers to obtain new skills and knowledge which results in more time and effort on app development.

Consequences

With consideration of the popularity of usage, the market share of React Native is significantly higher than that of Xamarin. The community support from React Native is higher. There is also no planned transformation of React Native. Developers who are already familiar with these skills are not required to acquire new skills and knowledge. Development time can be shortened.

Because of the upcoming transformation from Xamarin to .NET MAUI, developers need additional time and effort to learn and master the new skills. They may be more likely to make mistakes initially. Using React Native for a UI Framework of this app is a better decision.

Decision 3 - Backend Language

Decision: Node.js

The choice of Node.js as the backend language was made after consideration of alternative options, including Python and Ruby on Rails.

Rationale

Node.js allows JavaScript to be running on both server-side and client-side, enabling full-stack JavaScript development. It can handle asynchronous operations which allow us to process multiple connections at the same time. Its lightweight and fast execution environment results in low-latency responses and high throughput. However, real-time capabilities require thorough design and scaling to handle heavy traffic and loads. Development teams must be extremely proficient in both front-end and back-end development, which could affect team composition.

While Python is renowned for its simplicity and readability, making it easy to maintain and develop, it does not support full-stack JavaScript development. Ruby on Rails is known for its convention over configuration which allows rapid development time due to its clear project structure. This makes it an option for specific projects if time is limited. However, it might not perform as good as Node.js when it comes to handling real-time updates and concurrent connections.

Consequences

Node.js was chosen because of its strong real-time performance. It offers an event-driven, lightweight, and fast execution environment which is essential for managing multiple connections efficiently and delivering 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 like the payment system. By choosing Node.js, we aim to ensure that the app can deliver real-time services and handle the expected loads efficiently. While both Python and Ruby on Rails have their advantages, Node.js is best suited to our project's needs.

Decision 4 - Permissions

Decision: Granular permissions system

The app needs to obtain permission from users before enabling its functionalities to protect user's personal data and privacy.

Rationale

The single Consent approach is a straightforward approach to obtaining permission from users to grant or deny access to all requested permissions at once. Conversely, Granular permissions allow users to configure the privacy and security level by granting or denying each specific function or feature one by one.

Consequences

This app requires permission from users to enable several functions including push notifications and user behavior tracking. If using Single Consent, users may overlook and unintentionally grant access to the app which may lead to unplanned data authorization to the app. Having specific permissions configured one by one requires the user's additional input, it safeguards the user's privacy which enhances user trust and confidence.

Decision 5 - Data Storage

Decision: SQLite for Data Storage

Using SQLite rather than NoSQL for local data processing and storage.

Rationale

SQLite is a Relational Database Management System (RDBMS). All data are organized into tables with predefined schemas. It is a lightweight, efficient, and self-contained database engine for data storage and processing locally, allowing users to access and manipulate data stored locally in offline mode.

NoSQL is a flexible and schema-less data model which encompasses a variety of database management systems. It is usually designed for handling unstructured and/or semi-structured data. Although NoSQL offers offline data access and manipulation, it solely depends on specific databases and corresponding configurations.

Consequences

Although both types of database systems offer data storage, manipulation, and synchronization, all data such as financial transactions and inventory management processed and stored in this app requires fixed data structure, consistency, and integrity. SQLite is more capable of maintaining consistency and reliability and being free from anomalies which results in a lower risk of errors and data corruption.

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.

AWS and Microsoft Azure also offer tools for push notifications, payment gateways, and analytics. However, it requires more extensive development and configuration. Firebase is more like a mobile-focused platform specifically, it is easier to learn and user-friendly which can streamline the integration process.

With the use of JPEG image format, images can be optimized for browsing, loading, and storage. Although WebP can perform the same features as JPEG, its color profile is more limited than that of JPEG.

Consequences

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 [2]. Analysis generated by analytic tools provides detailed insights into user behavior and app usage, helping developers to 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.

It is important to note that 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. Furthermore, JPEG does not support lossless compression. If the image is heavily compressed, the loss of quality is noticeable.

References

1 T. Bahrynovska. “Choosing Your Mobile Development Framework: A Comparison of Xamarin and React Native.” Forbytes. Accessed: October 9, 2023. [Online.] Available: https://forbytes.com/blog/xamarin-vs-react-native/.

2 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.