Architectural Decisions

Anoop Kaur

SADT, Southern Alberta Institute of Technology

CPSY-303: Mobile Application Development

Dr. Kamini Sehmi

March 15, 2024

1. Native, web, or hybrid app

**Decision**: Native app

**Rationale**: A native approach makes the most sense because it provides access to device features like offline mode and push notifications while maintaining optimal performance. Compared to web or hybrid solutions, this guarantees smooth integration with device capabilities and offers a better user experience.

1. UI Framework

**Decision**: React Native

**Rationale**: React Native allows for Cross-platform development without compromising native performance. Faster development and simpler maintenance are made possible by its robust developer community and wide selection of UI components.

1. Backend language

**Decision**: Node.js

**Rationale**: Node.js provides a scalable and effective backend solution for managing immediate data synchronisation and interfacing with payment gateways. Its event-driven architecture complies well with data synchronization and push notification needs.

1. Permissions

**Decision**: Utilize a custom permission model with OAuth 2.0 authentication

**Rationale**: Ensuring secure access control to app features and data is ensured by implementing a custom permission model with OAuth 2.0 authentication. OAuth 2.0 offers a standardized architecture for authorization and authentication, enabling smooth user authentication integration with a range of third-party services while upholding strong security protocols.

1. Data storage

**Decision**: Realm Database for local storage, PostgreSQL for server-side storage

**Rationale**: With real-time data synchronization features necessary for offline mode support, Realm Database provides a quick and effective local data storage solution for mobile devices. PostgreSQL is a dependable relational database management system that may be used to manage intricate server-side data structures while maintaining data integrity and scalability for handling user and product catalog data.

1. Any additional frameworks or technology stacks

**Decision**: Firebase Cloud Functions for serverless backend, Flutter for UI development

**Rationale**: Push alerts and real-time data synchronization are made possible by the smooth integration of Firebase Cloud Functions with Firebase services, which offer a scalable and affordable serverless backend solution. With Flutter, you can create native interfaces quickly and with consistency across the iOS and Android platforms thanks to a cross-platform UI toolkit.

 Set page margins to 1 inch on all sides.

 Double-space all text, including headings.

 Indent the first line of every paragraph 0.5 inches.

 Use an accessible font (e.g., Times New Roman 12pt., Arial 11pt., or Georgia 11pt.).

 Include a page number on every page.