Southern Alberta Institute of Technology

CPRG 303B Mobile Application Development (Fall 2023)

Assignment 2: Architecture Decision Record (ADR)

Scenario 3: Food Delivery App

Student Name: Roselyn Chan Student id: 000900134

Background

Our mobile app development team is setting out to produce an application about ordering and delivering food from restuarants. Our mission is to enable customers to easily purchase meals from nearby restaurants for rapid pickup or delivery. We prioritise the user experience by offering account creation and secure payment information storage, all wrapped within a seamless and user-friendly interface.

Choices on the backend language, access management, and data storage options, in addition to whether to create a native, web, or hybrid application and the UI framework, are included in the architecture decision record. In addition, decisions about how to handle payments, implement push notifications, integrate our app with restaurant inventory management systems, and track and geolocate in real-time are documented.

Title: Native, Web, or Hybrid App

Context

The development team must choose an architectural approach for developing the food delivery app. The decision between developing a native, web, or hybrid app will have a substantial impact on the user experience, development effort, and maintenance.

Options Considered

- Native App: Create different versions for iOS and Android using platform-specific languages and technologies.
- Web App: Create a web-based app that can be accessed via browsers on multiple devices.
- Hybrid App: Create a single codebase for cross-platform deployment using a framework such as React Native or Flutter.

Decision

The team has decided to create a native app for both iOS and Android. The following factors influenced our decision: Our target audience primarily uses iOS and Android devices. Native app provides the best performance and user experience since they are tightly integrated with device features like as GPS and push notifications [1].

Status

Accepted

Consequences

This option gives users the best experience and works with all devices, but it means keeping different codebases for iOS and Android, which could make the costs of development and maintenance higher. We think that having a shared codebase like React Native or Flutter would save us money and help development go more quickly.

Title: UI Framework

Context

The development team needs to select a UI framework which supports responsive design and ensures an intuitive and user-friendly experience.

Options Considered

- React Native: Uses JavaScript and React, offering a large community and flexibility for integrating with native code.
- Flutter: known for UI consistency, performance, and native-like experience, using the Dart programming language [2]
- Vue NativeScript: Vue.js-based, flexible, and allows native-like development with NativeScript integration.

Decision

The team has selected React Native as the framework for our food delivery app because of its large and active community, which ensures robust support and a wealth of resources. Additionally, our team's expertise in JavaScript aligns well with React Native, allowing us to leverage our existing skills for efficient development.

Status

Accepted

Consequences

Because of its big and active community, which offers robust support and a multitude of resources, the team has chosen React Native as the framework for our food delivery app. Furthermore, the team has an expertise in JavaScript that fits nicely with React Native.

Title: Backend Language

Context

Selecting the right backend language is critical for developing the server-side components of the food delivery system. Scalability, performance, and integration with other services should all be supported by the chosen language.

Options Considered

- Node.js: Known for its non-blocking I/O and JavaScript ecosystem[3].
- Ruby on Rails: A robust framework for rapid development.
- Python (Django): Suitable for handling complex data operations.

Decision

For the back end language, the team has picked Node.js. This choice was made because it supports easy scaled up or down. Also, there are supportive libraries and frameworks for making RESTful APIs.

Status

Accepted

Consequences

Node.js gives our food delivery app fast performance and the ability to grow, but it might have trouble with heavy computing jobs like processing orders and keeping track of them in real time. These jobs could slow down important parts. Therefore we may need to optimise for computations that use a lot of CPU power.

Title: Permissions

Context

The development team must determine the permission model for the food delivery app. Permissions will govern access to sensitive user data, device features, and application functionality. For user protection and safety, making sure that permissions are handled correctly is essential.

Options Considered

- Role-Based Access Control: Implement a role-based permission system, defining different user roles (e.g., customer, delivery driver, restaurant owner) and their associated permissions.
- OAuth 2.0: Utilize OAuth 2.0 for granting and managing permissions, allowing for third-party integrations and secure access control [4].
- Custom Permission System: Develop a custom permission management solution tailored to the specific needs of the app.

Decision

The team has chosen to implement a role-based access control. The app needs different user roles, and each part needs different access rights and this choice fits with those needs. It offers flexibility and ease of management for defining and controlling permissions.

Status

Accepted

Consequences

Role-based access control provide granular control over access but will require careful configuration set up to ensure that permissions match with user roles and their responsibilities.

Title: Data Storage

Context

The food delivery app needs to be able to manage user profiles, order data, location information, customer ratings and reviews of restaurants, and other things. Users need to be able to see their order history and reorder their favourite items. Since the data to be stored is enomous and the retrieval needs to be efficient, it's important to choose the right data storage option that can keep the data safe.

Options Considered

- Relational Database (e.g., PostgreSQL): Use a traditional relational database for structured data storage.
- NoSQL Database (e.g., MongoDB): Opt for a NoSQL database for flexibility and scalability, especially for semi-structured data [5].
- Cloud-Based Storage (e.g., Amazon S3): Utilize cloud-based storage services for handling images, media, and user-generated content[6].

Decision

The team has chosen to use a relational database (PostgreSQL) for structured data storage for the need for transactional data, consistency, and support for complex queries. PostgreSQL offers robust features and has a strong track record for reliability. Status: Accepted

Consequences

Using a relational database provides data integrity and structured storage. However, it may require careful schema design in response to data requirements.

Title: Payment Handling

Context: It is critical to provide safe and easy transactions within the food delivery app. For the app to successfully fulfil this criteria, it must be integrated with multiple payment gateways.

Options Considered:

- Stripe: A developer-focused payment gateway known for its simplicity and flexibility. It enables businesses to accept online payments with ease, offering a wide range of customization options and support for various payment methods[7].

- Braintree: A subsidiary of PayPal, provides a scalable and customizable payment processing solution. It offers support for mobile and web payments, making it suitable for in-app transactions and providing a seamless user experience[8].

Decision

The team has chosen to integrate Stripe as the payment gateway solution based on Stripe's well-known reputation for safe and developer-friendly payment processing. Stripe provides an extensive feature set along with simple integration options.

Status

Accepted

Consequences

Integrating Stripe simplifies payment processing, enhancing the user experience. To ensure safe transactions within the app, it is necessary to closely follow the payment card industry compliance guidelines and keep a close eye on Stripe's API updates.

Title: Real-time Tracking and Geolocation

Context

In order to give accurate restaurant recommendations, estimate delivery times, and display nearby options, real-time tracking and geolocation technologies must be implemented.

Options Considered

- Google Map API: A comprehensive geospatial service known for its accuracy and developer-friendly tools
- Mapbox API: An API known for its mapping and location-based services, providing custom maps and navigation solutions

Decision

The team has decided to utilize the Google Map API for real-time tracking and geolocation. Google Maps provides extensive geospatial services and is well-known for its precision and developer-friendly tools.

Status

Accepted

Consequences

Integrating Google Map API enables exact location monitoring but necessitates compliance with Google's usage regulations and potential charges.

Title: Integration with Restaurant Inventory Management Systems

Context

To provide accurate and up-to-date restaurant menus, the food delivery app must be integrated with the inventory management systems of the restaurants. The team must discover the most effective and dependable technique of synchronising menu data, taking advantage of current technologies that support this function.

Options Considered

- RESTful API Integration: Establish direct RESTful API connections with restaurant inventory management systems using technologies like Node.js and Express.js to retrieve and synchronize menu data.
- GraphQL Integration: Implement GraphQL-based data queries[9] to seamlessly fetch and synchronize menu data from restaurant systems
- Web Scraping with Puppeteer[10]: Utilize Puppeteer, a headless browser automation tool, to perform web scraping techniques on restaurant websites and synchronize the extracted menu data with the app's database.

Decision

The team has decided to use GraphQL Integration to synchronise menu data from restaurant inventory management systems. This decision is influenced by a number of important aspects. GraphQL facilitates data retrieval by eliminating needless data transport and increasing synchronisation speed. It offers clients flexibility by allowing them to request only the data they require for menu data synchronisation. The scalability of GraphQL is advantageous as the app grows, meeting varying data requirements from numerous restaurant partners.

Status

Accepted

Consequences

To implement GraphQL integration, work with restaurant partners to create GraphQL schemas and resolvers for menu data retrieval. Continuous monitoring is required to guarantee that data synchronisation stays reliable. To improve efficiency, the team could also investigate handling GraphQL error replies and creating caching techniques.

Title: Push Notification Implementation

Context

It is absolutely necessary to put in place a push notification system in order to keep users up to current on order confirmations, shipping status updates, and promotional offer information.

Options Considered

- Firebase Cloud Messaging: Google-based messaging service [11]
- OneSignal: A third-party push notification service known for its ease of use and integrations with other platforms

Decision

The team has decided to use OneSignal to implement push notifications. This selection is based on OneSignal's reputation as a user-friendly and feature-rich push notification service that is simple to integrate with a variety of platforms [12].

Status

Accepted

Consequences

Choosing OneSignal simplifies the development of push notifications and provides a uniform platform for communication across numerous media. However, it is critical to be aware of potential delays in notice transmission, particularly during peak traffic periods.

Summary

To summarize, the ADR denotes the important choices that were made to make an innovative food delivery app that is efficient, scalable, and incoporated with technologies to enhance functionality and user engagement.

#References [1] M. Reality, “Cross-platform vs Native App Development: Ultimate comparison,” Cross-platform vs native app development: Ultimate Comparison, https://themobilereality.com/blog/cross-platform-vs-native-app-development (accessed Oct. 9, 2023). [2] Admin, “Flutter vs. react native: Which is the right cross-platform,” Scadea, https://scadea.com/flutter-vs-react-native-which-is-the-right-cross-platform/ (accessed Oct. 9, 2023). [3] “What is node.js and why you should use it,” Kinsta®, https://kinsta.com/knowledgebase/what-is-node-js/ (accessed Oct. 9, 2023). [4] Auth0, “OAuth 2.0 Authorization Framework,” Auth0 Docs, https://auth0.com/docs/authenticate/protocols/oauth (accessed Oct. 9, 2023). [5] K. Supe, “When to Use a NoSQL Database,” When to use a nosql database, https://memgraph.com/blog/when-to-use-a-nosql-database (accessed Oct. 9, 2023). [6] S. Engdahl, “Processing user-generated content using AWS Lambda and FFmpeg,” Amazon, https://aws.amazon.com/blogs/media/processing-user-generated-content-using-aws-lambda-and-ffmpeg/ (accessed Oct. 9, 2023). [7] Stripe, “Payments stack 101: How to create the right one,” Stripe, https://stripe.com/en-at/resources/more/payments-stack-101 (accessed Oct. 9, 2023). [8] Braintree, “Boost revenue with a global payments partner,” Braintree, https://www.braintreepayments.com/ca (accessed Oct. 9, 2023). [9] Apollo Docs, “Queries - Fetch data with the useQuery hook,” Apollo Docs, https://www.apollographql.com/docs/react/data/queries/ (accessed Oct. 9, 2023). [10] G. Thomas, “Web scraping in JavaScript – how to use puppeteer to scrape web pages,” freeCodeCamp.org, https://www.freecodecamp.org/news/web-scraping-in-javascript-with-puppeteer/ (accessed Oct. 9, 2023). [11] Google, “Firebase Cloud Messaging,” Google, https://firebase.google.com/docs/cloud-messaging (accessed Oct. 9, 2023). [12] J. M.

Adler, “Firebase Cloud Messaging (FCM) vs. OneSignal,” Customer Engagement Blog, https://onesignal.com/blog/firebase-vs-onesignal/ (accessed Oct. 9, 2023).