**Architectural Decision Record (ADR)**

**SCENARIO 3**

**Introduction**

This document talks about choices we made for making a food delivery app. We had to think about things like finding locations, giving real-time updates, keeping transactions safe, and making sure users like the app. We chose to make a mobile app using React Native, with Node.js and Express.js for the computer part. This way, we hope the app is good for users while still being fast and strong.

**1: React Native App**

**Decision**

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

**Rationale**

Better performance, an enhanced user experience, and seamless device feature integration - like GPS position tracking - are all provided by native apps. By utilizing platform-specific features and user interface components, we may give users a more user-friendly experience. Furthermore, native programming makes that the app is compatible with the newest OS features and updates, which increases its scalability and lifetime.

**2: UI Framework**

**Decision**

We will utilize a native UI framework for each platform: Swift UI for iOS and Jetpack Compose for Android.

**Rationale**

Platform-specific design standards and components are accessible through native UI frameworks, ensuring a uniform user experience across many devices. Declarative syntax is provided by Swift UI and Jetpack Compose, which minimizes boilerplate code and streamlines UI development. In addition, these contemporary frameworks allow us to design aesthetically pleasing interfaces without sacrificing performance thanks to features like animations and responsive layouts.

**3: Backend Language**

**Decision**

We will use Node.js for the backend development.

**Rationale**

Non-blocking I/O is a feature of Node.js that makes it ideal for managing concurrent requests in real-time applications such as ours. Scalability and effective management of asynchronous tasks, such retrieving data from external APIs and overseeing payment transactions, are made possible by its event-driven architecture. Furthermore, Node.js boasts a vast library and framework ecosystem that speeds up the creation and implementation of backend services.

**4: Permissions**

**Decision**

We will implement granular permissions management for accessing location, push notifications, and user data.

**Rationale**

Granular permissions limit access to certain functionalities only when expressly permitted, protecting user privacy and security. In accordance with platform policies and industry best practices for handling sensitive data, we will ask users for permission to access their device's GPS data to track their whereabouts. In a similar vein, authorization for push notifications will be asked to provide users with timely updates and alerts. To maintain compliance and confidence, user data permissions will adhere to GDPR and other pertinent rules.

**5: Data Storage**

**Decision**

We will utilize a combination of relational and NoSQL databases for data storage.

**Rationale**

Structured data, such as user accounts, order histories, and restaurant details, will be stored in relational databases like PostgreSQL. In addition to supporting sophisticated queries for effective data management and retrieval, these databases provide ACID compliance. We will leverage NoSQL databases like MongoDB or Firebase Fire store for semi-structured or unstructured data, such user reviews and real-time order tracking. NoSQL databases can manage dynamic and quickly changing data because of their scalability and flexibility.

**6: Additional Frameworks or Technology Stacks**

**Decision**

We will integrate the following additional frameworks and technology stacks:

- Express.js for building RESTful APIs and handling HTTP requests.

- Firebase Authentication for user authentication and authorization.

- Stripe or PayPal for payment gateway integration.

- Firebase Cloud Messaging (FCM) for push notification delivery.

**Rationale**

By offering a lightweight and adaptable framework for building RESTful APIs and facilitating smooth client-server communication, Express.js streamlines backend development. Strong user authentication features including social login, email/password authentication, and JWT token-based permission are provided by Firebase Authentication, which also improves the app's security and user management. Payment processing can be made safe and dependable by integrating Stripe or PayPal, which supports a range of currencies and payment methods to satisfy different user preferences. To ensure prompt communication and engagement, Firebase Cloud Messaging (FCM) offers a dependable and scalable infrastructure for push notification delivery to users across many platforms.

**Conclusion**

Our choices should make a good and strong food app. We need to keep checking if people like it, if it works well, and if more people can use it. We need to keep watching and changing things when needed.