ADR FOR GROUP 6

Title - Mobile App Architecture for Food Ordering Service

Date - 03/13/24

Authors - Azim Mohammed, Ananya Panthagan, Daljit Kaur **&** Jashanpreet Singh

Status - Accepted

Context - Our team has been tasked with developing a mobile app for a client in the food industry. The app aims to allow users to order food from local restaurants for delivery or pickup, create accounts, and save payment information. It should have a user-friendly interface, support real-time order tracking, integrate with various payment gateways, and provide access to user order history.

Decision:

1. **Type of App:** Native App
   Rationale: A native app is chosen to provide optimal performance, access to device features, and seamless user experience. Given the nature of the food ordering service, a native approach ensures reliability and responsiveness.Native development allows us to leverage platform-specific optimizations, resulting in faster load times and smoother interactions. Additionally, access to device features like GPS for location tracking and camera for scanning enhances the app's functionality and user experience.

2. **UI Framework:** Flutter
   Rationale: Flutter offers cross-platform development with native performance, enabling us to build a consistent UI across iOS and Android platforms. Its hot reload feature enhances development speed and iteration cycles. Furthermore, Flutter's rich set of customizable widgets allows us to create visually appealing interfaces that adhere to the client's branding guidelines. Additionally, Flutter's growing community and

extensive documentation provide valuable resources for troubleshooting and support during the development process.

3. **Backend Language:** Python with Django Framework
   Rationale: Python with Django provides a robust and scalable backend solution. Django's built-in authentication system, ORM, and REST framework facilitate rapid development of APIs for user authentication, order management, and integration with external services.Django's security features, such as CSRF protection and SQL injection prevention, enhance the overall security posture of the application, ensuring data integrity and protection against common web vulnerabilities. Additionally, Django's extensive ecosystem of third-party packages and libraries further accelerates development by providing pre-built solutions for common tasks and functionalities.

4. **Permissions**: Minimal Permissions Approach
   Rationale: To prioritize user privacy and security, we will adopt a minimal permissions approach, only requesting necessary permissions such as location for restaurant recommendations and payment information for transaction processing.This approach not only enhances user trust but also reduces potential risks associated with excessive data collection. Additionally, clear communication with users regarding the purpose of requested permissions fosters transparency and reinforces our commitment to protecting user privacy throughout their interaction with the app.

5. **Data Storage:** SQL Database (PostgreSQL)
   Rationale: PostgreSQL offers ACID compliance, scalability, and support for complex queries. Its relational model suits the structured nature of user accounts, order data, and restaurant menus. Furthermore, PostgreSQL's extensibility allows us to implement custom data types and functions tailored to the specific needs of our application, enhancing flexibility and performance. Additionally, PostgreSQL's robust replication and failover capabilities ensure high availability and data durability, crucial for a mission-critical application like ours.

6. **Additional Frameworks/Technology Stacks:**

   Google Maps API for Location Tracking

   a. **Rationale:** Google Maps API provides accurate location tracking and geocoding services, essential for recommending nearby restaurants and estimating delivery times. Additionally, its comprehensive set of mapping features enables us to visualize restaurant locations, display interactive maps to users, and calculate optimal routes for delivery. Moreover, integration with Google Maps API allows seamless user experience, as users can easily navigate through the app and view restaurant details with familiar map functionalities.

   WebSocket for Real-time Order Tracking

   b. **Rationale:** WebSocket enables bi-directional communication between the client and server, allowing real-time updates on order status and delivery progress. This facilitates a more dynamic and engaging user experience, as users can receive immediate notifications and updates without needing to manually refresh the app. Moreover, WebSocket's low latency and efficient data transfer make it ideal for applications requiring real-time interaction, ensuring timely delivery of critical information such as order confirmations and driver updates.

Consequences:

Pros:

- Native apps ensure optimal performance and user experience.

- Flutter facilitates cross-platform development with native-like UI.

- Python with Django offers a robust backend solution with built-in security features.

- Minimal permissions approach prioritizes user privacy.

- PostgreSQL provides reliability and scalability for data storage.

- Integration of Google Maps API and WebSocket enhances app functionality and user experience.

Cons:

- Adoption of multiple technologies (Flutter, Django, Google Maps API, WebSocket) may require an additional learning curve for team members.
- Maintenance and updates may be needed to ensure compatibility with evolving versions of integrated frameworks and APIs.
- Dependency on external services like Google Maps API and WebSocket may introduce potential points of failure and reliability concerns, requiring robust error handling and fallback mechanisms.

References:

- Flutter: https://flutter.dev/
- Django: https://www.djangoproject.com/
- PostgreSQL: https://www.postgresql.org/
- Google Maps API: https://developers.google.com/maps/documentation
- WebSocket: https://developer.mozilla.org/en-US/docs/Web/API/WebSocket

Conclusion:

The chosen architectural decisions align with the requirements of the food ordering service mobile app. By opting for a native approach with Flutter for frontend development, Python with Django for backend services, and integrating essential tools such as Google Maps API and WebSocket, we ensure a scalable, secure, and user-centric solution. This architecture lays a strong foundation for delivering a high-quality mobile app that meets the client's expectations and provides a seamless food ordering experience for users.