Enterprise Solution Architecture for Everything Inc. Mobile App

Introduction

This document outlines the proposed architecture for Everything Inc.'s new Android application following the merger with Voda Inc. and Ora Inc. The app will provide a unified experience for customers of all three companies, allowing them to manage their telecom plans, monitor consumption, and personalize their dashboards.

Overall System Architecture

(Diagram: High-level system architecture with components like Mobile App, API Gateway, Authentication Service, MBaaS (future state), Existing systems of Everything Inc., Voda Inc., and Ora Inc.)

- Mobile App: The Android application acts as the primary interface for users to interact with the system.
- API Gateway: This component acts as a single entry point for the mobile app to access backend services. It routes requests to appropriate backend systems based on the user and request type.
- Authentication Service: A central service for user authentication and authorization across all three companies.
- MBaaS (Future State): A middleware layer that will aggregate and mediate API requests to the various backend systems, providing a unified data access layer for the application.
- Existing Systems: The current systems of Everything Inc., Voda Inc., and Ora Inc. will be integrated with the API Gateway to provide necessary data and functionality.

Android Application Architecture

(Diagram: Detailed Android application architecture with layers like Presentation, Business Logic, Data, and External Dependencies)

- Presentation Layer: Responsible for the user interface and user experience. It uses Android Jetpack components like Activities,
  Fragments, and ViewModels to manage the UI and handle user interactions.
- Business Logic Layer: Implements the core business logic of the application, including data processing, validation, and calculations. This layer interacts with the data layer to retrieve and manipulate data.
- Data Layer: Handles data access and persistence. It utilizes the Repository pattern to abstract the data source and provides a clean interface for the business logic layer. It interacts with the API Gateway to fetch data from backend systems.
- External Dependencies: Includes libraries and frameworks used within the application, such as Retrofit for network communication, Glide for image loading, and Room for local data storage.

Runtime/Execution Architecture

(Diagram: Runtime architecture showcasing components like Android OS, Dalvik/ART, Core Libraries, App Components, 3rd party libraries)

- Android OS: Provides the underlying operating system environment for the application.
- Dalvik/ART: The runtime environment responsible for executing the application code.
- Core Libraries: Android provides a set of core libraries for functionalities like UI, network, and data handling.
- App Components: The application itself consists of Activities, Services, Broadcast Receivers, and Content Providers.
- 3rd Party Libraries: External libraries like Retrofit, Glide, Room, etc. enhance the application's functionality and development efficiency.

- **High and Low-Level Design**

  High-Level Design:
  - The application will be modularized with well-defined interfaces between components, allowing for easier maintenance and testing.
  - The MVVM (Model-View-ViewModel) architectural pattern will be implemented to separate UI logic from business logic and data handling.
  - Dependency injection will be used to manage dependencies and improve code testability.

  Low-Level Design:
  - Specific implementation details for each component, including class diagrams, sequence diagrams, and data flow diagrams, will be documented separately.
  - Coding standards and guidelines will be established to ensure code consistency and maintainability.

  Assumptions/Risks
  - Minimum Android API Level: The application will target Android 5.0 (Lollipop) to support a wider range of devices while stilll leveraging modern Android features.
  - Target Devices: The application will be designed to work on a variety of Android smartphones and tablets with different screen sizes and resolutions.
  - Backend Integration: The initial reliance on separate web services might introduce complexity, but the future MBaaS layer will help mitigate this risk.
  - Data Migration: Successfully migrating user data from the three companies existing systems is crucial for a smooth transition.

  Conclusion

  This proposed architecture provides a scalable and flexible foundation for Everything Inc.'s new Android application. By leveraging modern Android technologies and design patterns, the application can deliver a seamless and personalized experience for customers of all three merged companies.

  Additional Notes:
  - Consider incorporating CI/CD pipeline for automated builds and deployments.
  - Implement comprehensive testing strategy including unit, integration, and UI tests.
  - Prioritize security best practices for data protection and user privacy.
  - Continuously monitor and optimize app performance for a smooth user experience.

Visualizing the Architecture: Diagrams

Due to my limitations as a text-based AI, I cannot directly generate images or diagrams. However, I can provide detailed descriptions and suggestions for creating the visuals for your PPT deck.

1. Overall System Architecture Diagram

Tools: Draw.io, Lucidchart, Microsoft Visio

Components:

- Mobile App (Android): Represent this with a smartphone icon.
- API Gateway: A rectangular box labelled "API Gateway".
- Authentication Service: A rectangular box labelled "Authentication Service".
- MBaaS (Future State): A rectangular box with dashed lines indicating its future implementation, labelled "MBaaS".

- Existing Systems: Depict these as separate rectangular boxes for each company (Everything Inc., Voda Inc., Ora Inc.) with their respective logos or names.

Connections:

- Arrows pointing from the Mobile App to the API Gateway, signifying API requests.
- Arrow from API Gateway to Authentication Service, illustrating the authentication flow.
- Arrows from API Gateway to each of the existing systems and the future MBaaS, representing data flow.

2. Detailed Android Application Architecture Diagram

Tools: Same as above

Layers:

- Presentation Layer: Depict this as the top layer with boxes representing UI components like Activities, Fragments, and ViewModels.
- Business Logic Layer: The middle layer containing boxes for Use Cases or Interactors, representing the business logic processing.
- Data Layer: The bottom layer with boxes for Repositories and Data Sources (e.g., Remote data source for API interaction, Local data source for database).

Connections:

• Arrows between layers showing the interaction flow (e.g., Presentation layer interacts with Business Logic layer to perform actions, Business Logic layer interacts with Data layer for data access).

• Within the Data layer, show arrows depicting interaction between Repositories and Data Sources.

3. Runtime/Execution Architecture Diagram

Tools: Same as above

Components:

- Android OS: Base layer represented by the Android logo.
- Dalvik/ART: A layer above Android OS, labelled accordingly.
- Core Libraries: Depict this as a layer with boxes for different core functionalities like UI libraries, Network libraries, etc.
- App Components: A layer containing boxes for Activities, Services, Broadcast Receivers, and Content Providers.
- 3rd Party Libraries: A layer showcasing boxes for external libraries like Retrofit, Glide, Room.

Connections:

• Arrows demonstrating the hierarchy and interaction between layers. For example, App Components layer interacts with Core

Libraries and 3rd Party Libraries.

Additional Tips:

- Use consistent styling and colors for clarity.
- Label all components and connections clearly.
- Maintain a clean and uncluttered layout.
- Consider using different shapes for different types of components.

Remember, these are just basic suggestions, and you can adapt them to your specific preferences and presentation style.