Mumzworld Project : Tech & Product Decisions along with API Improvement Suggestions

By - Nishant Kaushik

Tech Decisions Made

1. **Chose Expo Managed Workflow for React Native Project**
    - **Why?**
        1. Less configuration required.
        2. Easy to start and quick to set up.
    - **Trade-offs:**
        1. Limited ability to modify native modules due to restricted configuration.
        2. Limited availability of third-party libraries, e.g., challenges finding a robust ORM tool compatible with Expo (tried Watermelon DB and TypeORM).
2. **EAS Deployment System**
    - Set up EAS (Expo Application Services) for deployment as per project requirements.
3. **Architecture Specific**
    - **Chose MVVM Pattern:**
        - Keeps the project clean and maintainable.
        - Utilizes Expo Secure Store, File Systems, and Axios for handling local and remote data sources.
4. **Performance Enhancements**

- Data is fetched at app launch.
- Implemented data caching to improve performance via ETAG comparison with remote resources.
- Cached data is stored in the file system.
- Once fetched, data is accessed from the in-memory domain model.
- **Decided to Use "File System" for Caching Instead of a "DB Approach"**
    - **Why?**
        - **Performance:**
            - **Speed**: Fast read/write access, especially for local file systems with minimal latency. This also results on a faster Search operation.
            - **Direct Access**: Avoids the overhead of SQL query parsing and execution.
        - **Efficiency:**
            - Requires only one I/O operation for changes, updating the local file in the background.
            - Changes to the dataset involve a single swap of the entire in-memory model.
    - **Trade-offs:**
        - **Scalability:**
            - **Issue**: Limited scalability for large datasets due to higher memory consumption.
        - **Search and Query Capabilities:**
            - **Issue**: Less efficient for complex queries as searching here is liner, DB approach would have resulted in efficient search as the Brand, Category and Product listings are indexed.
        - **Mitigation for Trade-offs**: For now the 2MB data size poses no issues; future scalability & search capabilities can be managed by efficient usage of DB

Product Decisions Made

1. **Brand & Category Listing**
    - Decided to display the first product’s image on the Brand & Category listing page since the listings do not contain images.
2. **Localization & Account Feature**
    - Added localization options directly to the Action Bar/Header, as the "Account" feature is not part of the MVP.
3. **Product Detail Page**
    - Displaying the same product details for all products due to the unavailability of the "get product by ID" feature.
4. **Search Functionality**
    - Limited search results to three Brands and three Categories to manage performance and user experience.

Suggested API Changes

1. Product label’s activeness is dependent on From Date and To Date right now, it would be good to have that from backend
2. Category & Brand’s image URL is missing
3. There is no spec for Error Codes
4. Base URL is missing from Product Details
5. In Listing API, Categories & Sub Categories are combined - ideally should be separate
6. Price values can be simplified
7. Some of the Brands have invalid ID