Senior Android Developer Technical Test

Technical Problem: Offline-First Task Management App

Challenge Description:

You are tasked with developing a task management app (to-do list style) that allows users to create, edit, and delete tasks.

The app must support synchronization with a remote API but also work effectively offline, ensuring the user can manage their

tasks even when there is no internet connection.

The focus of this challenge is to demonstrate your abilities in software architecture, state management, and offline data handling.

Additionally, the app should follow Android development best practices, ensuring the code is clean, testable, and maintainable.

Functional Requirements:

1. Task Management:

   - The user should be able to create, edit, delete, and mark tasks as completed.

   - Tasks should be stored locally and synced with an external REST API when a connection is available.

2. Online/Offline Synchronization:

   - When the device is offline, tasks must be stored locally.

   - When the connection is restored, offline changes (inserts, edits, and deletes) should sync with the server.

3. REST API:

- Consume an external API to sync tasks. For testing purposes, you can use a public API like JSONPlaceholder or a mock API.

4. Local Data Persistence:

- Use Room to store tasks locally in a persistent way.
- Manage offline state with a caching strategy.

5. Architecture:

- Implement Clean Architecture using MVVM (Model-View-ViewModel) pattern.
- Use the Repository Pattern to separate business logic from data access.

6. UI:

- Build a friendly and functional interface to manage tasks intuitively.
- Use RecyclerView to display the list of tasks, and ViewBinding or Jetpack Compose to bind the interface.

7. Testing:

- Write unit and instrumented tests to ensure the correctness of the core functionality, such as task storage and synchronization.

Tips:

- Use Kotlin Coroutines for asynchronous tasks, such as API calls and database operations.
- Use WorkManager to handle background data synchronization, ensuring tasks sync even if the app is closed.

- Observe changes in tasks using LiveData or StateFlow.

- Use Room with coroutines for database operations.

- Consider caching recent tasks to avoid unnecessary API calls.

Evaluation Criteria:

- Architecture and Code Quality: Clean Architecture and separation of concerns.

- Use of Best Practices: Efficient use of coroutines, dependency injection, and modern Android architecture.

- Complete Functionality: The app should function online and offline, with proper synchronization.

- Testability: Ensure unit and instrumented tests cover business logic and synchronization.

- Efficiency and Performance: The app should be resource-efficient, handle threading properly, and execute background tasks efficiently.

Estimated Duration:

This task should take approximately 5-6 hours.