Take Home Assignment

Build a simple Android application using Jetpack Compose that fetches and displays a list of products from a public API (https://dummyjson.com/docs/products) . The app should follow best practices in **architecture, UI state management, and offline caching**.

Features

1. Fetch a list of products from the API and display them in a list.
2. Show a product details screen when a user taps on an item.
3. Implement **offline caching** so the app can work without an internet connection.
4. Implement **error handling** (e.g., no network, API errors).
5. Add a **search bar** to filter products by name.

Technical Requirements

Use modern Android development tools and frameworks:

- Jetpack Compose for UI
- Hilt for Dependency Injection
- Retrofit for API calls
- Room for local caching
- Coroutines / Flow for async operations
- ViewModel for state management

UI Design

1. Product List Screen

- Display products.
- Each item should show **image, title, and price**.
- Clicking an item navigates to the Product Details Screen.
- A search bar should filter the list based on the product name.

2. Product Details Screen

- Display **image, title, description, and price**.
- Show a **“Buy Now”** button.

Evaluated based on:

1. **Code structure & best practices**
2. **Architecture (MVVM, Clean Architecture)**
3. **Use of Jetpack Compose & state management**
4. **Error handling & offline caching**
5. **Code readability & maintainability**
6. **Unit tests**

Secure Data Handling in Android

Scenario: You are developing an Android application that handles sensitive user data, including personally identifiable information (PII). Security is paramount, and you must ensure this data is encrypted both at rest on the device and during transmission to a backend server.

Task: Design and implement a solution for securely storing and transmitting user data. Specifically, demonstrate how you would:

1. **Encrypt data at rest:** Choose an appropriate encryption method for storing sensitive data locally on the Android device's storage. Justify your choice.
2. **Encrypt data in transit:** Show how you would encrypt this data when making a network request to a specified URL. Assume the backend server expects data encrypted using the same method as at rest.
3. **Handle decryption:** Briefly outline how the data would be decrypted on the backend (you don't need to implement the server-side decryption).

Requirements:

- Use a robust and secure encryption algorithm. Avoid insecure methods like MD5 or simple XOR.
- Prioritize security best practices, including proper key management. Explain how you would store and protect the encryption key.
- Provide a code sample in Kotlin or Java demonstrating the encryption and decryption processes. Focus on the core encryption/decryption logic. Assume you have helper functions for making network requests.
- Consider performance implications. While security is paramount, the encryption/decryption process shouldn't significantly impact the user experience.
- Assume the app targets API level 26 (Android 8.0 Oreo) or higher.