DevSolutions
GROUP

Flutter Developer Technical Task

Objective:

Build a simple eCommerce product viewer app using Flutter that showcases a list of products fetched from an API. The app should allow browsing products, viewing details, and managing a favorites list.

—

UI Design (Figma)

Use this public Figma design as inspiration:

[Figma - eCommerce App UI](*https://www.figma.com/design/cq5WGeh3thc09aYQYeuDLI/Ecommerce-App--Community-?node-id=60-27&p=f&t=h8ZAlTd4qRR4m7PC-0*)

You're not required to copy the design exactly, but aim for a clean and responsive UI similar to it.

Task Requirements

 # 1. Screens

- **Home Screen**: List products with image, name, and price.

- **Product Detail Screen**: Large image, title, price, description, and “Add to Favorites” button.

- **Favorites Screen**: List of favorited items only.

 # 2. Data

- Fetch products from:

https://fakestoreapi.com/products

 # 3. Features

- Add/remove items from favorites
- Persist favorites using local storage (e.g., Hive or SharedPreferences)
- Pull-to-refresh on the home screen
- Loading and error states

State Management Requirement

Let the candidate choose between Provider, Riverpod, or Bloc, and ask them to:

- Justify their choice in the README
- Show how state is managed across screens
- Structure the app into layers: UI, State, Service/API

> This will help you evaluate their understanding of architecture, separation of concerns, and reasoning skills.

Submission Guidelines

- Share code via GitHub.
- Include:
    - `**README.md**` with:
        - Setup instructions
        - Explanation of state management
        - Decisions made or trade-offs
        - Optional: test coverage or suggestions for improvement

What to Evaluate

Skill Area	What to Look For
UI/UX	How closely the app follows the Figma mockups, responsiveness, polish
Code Quality	Folder structure, naming conventions, clean architecture
State Management	Reasoning for chosen approach, clarity in state handling
API Integration	Proper error handling, loading states, clean data parsing
Local Storage	Effective persistence of favorites
Best Practices	Null safety, async handling, modular code, reusability

---

Estimated Duration:

3–5 hours