Projet Flutter — Plateforme e-commerce simplifiée

Nom du projet : SUP4 DEV E-Commerce Flutter App

Objectif :

Créer une application Flutter de vente en ligne basique, permettant :

- la gestion des produits,
- la gestion du panier et des commandes,
- un espace client et un espace administrateur.

Fonctionnalités attendues

Espace client

- Parcourir les produits.
- Ajouter au panier / Supprimer du panier.
- Passer une commande.
- Suivre l'historique des commandes.
- Appliquer des coupons de réduction.

Espace admin

- CRUD sur les produits.
- Suivi des stocks.
- Mise en avant des produits vedettes.
- Gestion des promotions.
- (Optionnel) Suivi logistique des colis (Track & Trace fictif).

Aspects techniques

Flutter & Dart

- Utilisation de **Flutter** pour le frontend.
- Stockage local via **SQLite**, ou persistance cloud avec Firebase ou Supabase.
- Architecture MVC ou **Clean Architecture (recommandé)**.
- Utilisation de **Provider, Riverpod ou Bloc** pour la gestion d'état.

Backend (optionnel)

- API simulée via **mock data** ou back-end Node/Spring/PHP si déjà existant.
- Paiement : simulation via un écran ou intégration d'une API de paiement sandbox (Stripe par exemple).

UI/UX

- Responsive.
- Theming clair/sombre.
- UI agréable avec **Flutter Material** ou **Flutter + Tailwind + Shadcn-like UI**.

Sécurité & Rôles

- Authentification (Firebase Auth ou solution personnalisée).
- Gestion des rôles : Client / Admin.

- Séparation des interfaces et des permissions.

Maquette visuelle de l'application (obligatoire)

Vous devez proposer une maquette visuelle de votre application (réalisée avec Figma, Adobe XD, Balsamiq).

- Fournissez un lien vers la maquette ou intégrez une capture d'écran claire de l'interface prévue.
- La maquette doit refléter la structure de votre application (navigation, organisation des pages, composants essentiels).
- Elle servira de base de réflexion et de développement tout au long du projet.

Cette étape est obligatoire et sera évaluée. Un bonus pourra être accordé si l'implémentation finale est fidèle à la maquette.

Aspects techniques

- Gestion sécurisée des utilisateurs via Firebase Auth.
- Affichage dynamique des créneaux avec interaction.
- Notifications des rendez-vous confirmés / annulés.
- Paiement possible via Stripe (ou simulation).
- Tests unitaires des services de réservation.
- Dashboards en temps réel pour les professionnels.

Structure du projet (exemple Clean Architecture)

```
lib/
├── models/          # Modèles de données : Produit, Utilisateur, Commande, Coupon
├── providers/       # Fournisseurs d'état (ex : panierProvider, authProvider)
├── services/        # Services : AuthService, ApiService, PaymentService
├── views/           # UI : écrans client et admin
│   ├── client/      # Accueil, panier, commande, historique
│   └── admin/       # CRUD produits, gestion promo, stock
├── widgets/         # Composants réutilisables (cards, boutons, champs)
└── main.dart
```

Critères d'évaluation pour accepter et recommander :

Critère	**Détail**
Durée du travail	1 mois minimum
Commits Git	20 commits minimum, bien nommés
Fonctionnel	Panier, commandes, espace admin et client
Architecture	Respect de MVC ou Clean Architecture
Qualité du code	Bien structuré, modulaire, lisible
Interface	Responsive et agréable

Déploiement	Démonstration via APK ou démo GitHub Pages/Firebase
Tests (bonus)	Présence de quelques tests unitaires ou widget tests

---

Suggestions bonus (si le développeur veut aller plus loin)

- Notifications (commande validée, livraison, etc.).
- Intégration Firebase Cloud Messaging.
- Upload d'image pour les produits (via image_picker).
- Dashboard admin simple (ventes, produits populaires).
- Multilingue (fr/en).