Application de gestion de colocation

Nom du projet 14: DEVMOB-Coloc'App

Objectif

Créer une application mobile pour les colocataires afin de faciliter la gestion quotidienne d'une colocation : répartition des tâches, suivi des dépenses, communication, et partage de documents.

C'est quoi l'idée ?

C'est une application mobile pour les colocataires (des gens qui vivent ensemble dans une maison/appartement) pour :

- mieux s'organiser,
- répartir les tâches ménagères (faire la vaisselle, sortir les poubelles, etc.),
- suivre qui a payé quoi (électricité, courses...),
- discuter entre eux via un petit chat,
- partager des documents comme les factures ou le contrat de location.

Qui utilise l'application ?

- Tous les colocataires : ils peuvent voir les tâches, les dépenses, les messages, les documents...
- Un colocataire "référent" (admin) : il peut ajouter ou supprimer des membres, valider certaines dépenses, modifier les règles de la colocation, etc.

Fonctionnalités attendues

Pour tous les colocataires :

- Voir et ajouter les tâches à faire (ménage, courses...).
- Ajouter une dépense (ex: j'ai payé 40DT de courses → le reste de la coloc me doit une part).
- Voir qui doit de l'argent à qui.
- Partager des documents (comme des factures, le contrat...).
- Voir un calendrier des événements de la maison (réunions, rappels...).
- Discuter avec les autres via une messagerie.

Espace colocataire (utilisateur)

- Créer ou rejoindre une colocation avec un code d'invitation.
- Répartir les **tâches ménagères** entre les membres.
- Ajouter des **dépenses** (électricité, courses, Internet...) et voir qui doit quoi à qui.
- **Calendrier partagé** : réunions, échéances de paiement, poubelles...
- **Liste de courses** collaborative.
- **Messagerie interne** (mini chat pour les colocataires).
- Stocker et consulter les **documents partagés** (PDF : contrat, factures...).
- Notifications (tâches à faire, dépenses à rembourser, événements à venir).

Espace admin (coloc référent)

- Gérer les membres (ajouter / supprimer).
- Définir les règles de répartition des tâches.
- Valider les dépenses importantes.
- Suivi global du fonctionnement de la colocation.

Aspects techniques

Flutter & Dart

- Flutter pour le développement multi-plateforme.
- Architecture Clean Architecture ou MVC.
- Gestion d'état avec Provider, Riverpod ou Bloc.
- Stockage avec Firebase (Auth + Firestore + Storage) ou Supabase.

Backend (optionnel)

- Utilisation de Firebase ou WebSocket pour la messagerie en temps réel.
- API REST personnalisée (Node.js, Laravel, Spring Boot).

Persistances

- Données en temps réel (Firestore).
- Stockage des documents (Firebase Storage).
- Synchronisation hors ligne possible (bonus).

Comment ça marche techniquement ?

Tu vas créer l'application avec Flutter, et tu peux utiliser Firebase pour :

- Auth : pour que chaque utilisateur se connecte.
- Firestore : pour stocker les données (tâches, dépenses...).
- Storage : pour les fichiers (factures, PDF...).
- Cloud Messaging (optionnel) : pour envoyer des notifications.

UI/UX

- Design convivial et jeune, ambiance coloc'.
- Mode sombre / clair.
- Thèmes personnalisables selon la colocation.
- Navigation intuitive avec BottomNavigationBar.

Sécurité & rôles

- Authentification via Firebase Auth (e-mail / mot de passe ou Google).
- Rôles : colocataire / référent.
- Accès aux fonctionnalités selon les permissions.

Maquette visuelle (obligatoire)

- Réaliser une maquette claire et interactive via Figma / Adobe XD.
- Présenter les écrans suivants : accueil, tâches, dépenses, documents, chat.

- Un bonus est accordé si le design est respecté dans l'app finale.

Structure du projet recommandée

```
lib/
├── models/          # Utilisateur, Tâche, Dépense, Document, Message
├── providers/       # authProvider, tasksProvider, expensesProvider
├── services/        # AuthService, TaskService, ChatService, FileService
├── views/
│   ├── home/        # Vue d'ensemble, navigation principale
│   ├── tasks/       # Gestion des tâches
│   ├── expenses/    # Dépenses et remboursements
│   ├── chat/        # Messagerie
│   └── settings/    # Paramètres de la colocation
├── widgets/         # Composants réutilisables : avatars, cards, dialogs
└── main.dart
```

Critères d'évaluation

Critère	**Détail**
Durée	1 mois minimum
Commits Git	Minimum 20, avec messages clairs
Fonctionnel	Tâches, dépenses, documents, chat
Architecture	Clean Architecture ou MVC
Qualité du code	Lisible, organisé, modulaire
Interface	Ergonomique, intuitive, responsive
Démo / APK	Application testable ou déployée sur Firebase
Tests (bonus)	Unitaire / widget tests

Suggestions bonus

- Génération automatique de planning de tâches.
- Export PDF des comptes ou du contrat de colocation.
- Notifications push via Firebase Cloud Messaging.
- Intégration du scan de tickets de caisse (OCR).
- Traduction multi-langue (français / anglais / espagnol).