WEEK 1

Project Title

CampusConnect: A Social Platform for University Students

Alternative Titles: UniHub, StudentSphere, CampusCircle

The name "CampusConnect" emphasizes the app's mission to link university students and enrich their academic and social experiences through a dedicated platform.

Problem Statement

University students encounter several issues that this app seeks to resolve:

Academic Collaboration: It's challenging to find or create study groups for specific courses or subjects.

Information Overload: There's no unified platform for campus events, deadlines, or club updates, leading to missed opportunities.

Resource Sharing: Students lack a trusted space to discuss coursework or exchange resources like notes or study guides.

Marketplace Needs: There's no secure, student-only platform for buying and selling items such as textbooks or dorm supplies.

Current social media platforms are too broad, missing features tailored to university life, which fragments communication and limits student engagement.

Objectives

The app's goals follow the SMART framework (Specific, Measurable, Achievable, Relevant, Time-bound):

Enhance Academic Support: Allow students to easily form study groups, share resources, and discuss coursework.

Streamline Campus Updates: Offer a centralized hub for university events, deadlines, and club activities.

Build Community: Provide a space for students to connect with peers, join interest-based groups, and engage in campus life.

Prioritize Safety: Limit access to verified university students for a secure, relevant environment.

Drive Adoption: Aim for 500 active users within six months of launch and expand to multiple universities within two years.

Target Users

Primary Audience: Undergraduate and graduate university students.

Secondary Audience: University clubs, organizations, and faculty sharing events or resources.

User Needs:

Quick access to study groups and academic materials.

A dependable source for campus event details.

A safe space to connect with peers and trade items.

User Traits: Tech-savvy, mobile-focused, privacy-conscious, and convenience-driven.

Technologies and Frameworks

The tech stack is optimized for efficiency and scalability, with Flutter as the frontend cornerstone:

Frontend:

Flutter: Enables cross-platform development for iOS and Android with one codebase.

Backend:

Firebase: Provides a scalable, real-time backend ideal for Flutter integration.

Node.js with Express: Optional for a custom API with greater control.

Python with Django: Alternative for a robust, secure backend.

Database:

Firestore (Firebase): Supports real-time data storage and syncing.

PostgreSQL or MongoDB: Alternatives for custom backend setups.

Authentication:

Firebase Authentication: Ensures secure login and email verification.

Real-Time Features:

Firebase Realtime Database or Firestore: Powers live chat and notifications.

Cloud Services:

Firebase Hosting and Cloud Functions: Offers scalable hosting and serverless logic.

Expected Design Patterns

These patterns ensure a clean, maintainable app structure:

MVC (Model-View-Controller): Separates data, UI, and logic using Flutter's widget system.

Singleton Pattern: Maintains single instances for services like authentication or database access.

Observer Pattern: Uses Flutter's StreamBuilder for real-time Firestore updates.

Repository Pattern: Centralizes data access for simplified management.

Provider Pattern: Employs the provider package for state management, syncing UI with app state.

Initial High-Level System Overview

The app's architecture is modular, scalable, and user-friendly, leveraging Flutter for the frontend.

Frontend (Flutter):

Cross-platform mobile app for iOS and Android.

Core screens: Home Feed, Event Calendar, Group Chats, Marketplace, Profile.

Navigation via BottomNavigationBar for intuitive access.

Backend:

Firebase Cloud Functions or a custom backend (e.g., Node.js with Express) for API and logic handling.

Database:

Firestore stores user profiles, posts, events, groups, and marketplace listings.

Authentication:

Firebase Authentication verifies users via university emails (e.g., .edu).

Real-Time Engine:

Firestore or Firebase Realtime Database drives live chats and notifications, integrated with StreamBuilder.

Cloud Infrastructure:

Firebase Hosting for a potential web app; Cloud Storage for media files.

Third-Party Tools:

Firebase Cloud Messaging (FCM) for push notifications.

Firebase Analytics for usage insights.

System Flow:

Students sign up with a university email, verified by Firebase Authentication.

The Flutter app fetches Firestore data to display the Home Feed, Events, and Groups.

Users post updates, join groups, share events, or list marketplace items, saved to Firestore.

Real-time features (chats, notifications) update instantly via Firestore listeners.

The backend processes tasks like reminders, while Flutter handles the frontend.

Firebase's infrastructure ensures seamless scaling.