Cross Platform Mobile App Developer

Of course. Based on the technologies you've provided, here is a tailored project for a Cross-Platform Mobile App Developer using the Flutter and Firebase ecosystem.

Project: Collaborative Shopping List App

- **Objective**: To build a full-stack, real-time collaborative shopping list application for both iOS and Android using Flutter and the Firebase suite.
- **Description**: You will create a mobile app where users can register an account, create shared shopping lists, and invite others to collaborate. Any changes to a list, such as adding an item or checking it off, will appear in real-time on all collaborators' devices. This project will emphasize a clean UI, a secure backend, and a tested codebase.
- **Technologies to use**: **Flutter**, **Dart**, **Firebase**, **Git & GitHub**, **Unit testing**, **UI-UX**, **Security Essentials**, **Mobile Deployment**.

Week 1: Setup, UI/UX, and Flutter Basics

Tasks:

- **Project Setup**: Initialize a new Flutter project and set up a repository with **Git & GitHub** for version control.
- **UI - UX Design**: Create wireframes for the key screens: user login/registration, the main screen showing all shopping lists, and the detailed view of a single list with its items.
- **Build Static UI**: Applying **Dart Fundamentals** and **Flutter Basics**, code the static user interface for the screens you designed. Focus on layout and visual appearance without implementing any logic yet.

Deliverables:

- A Flutter project initialized and pushed to a GitHub repository.
- Wireframes for the application's user flow.
- Static, non-functional UI screens for the login, main list, and detail views.

Week 2: Firebase Integration and Authentication

Tasks:

- **Integrate Firebase**: Add Firebase to your Flutter project for both Android and iOS platforms.
- **Implement Authentication**: Use **Firebase** Authentication to build the user registration and login functionality. Apply **Security Essentials** by setting up rules to ensure users must be authenticated to access the app's data.
- **Database Structure**: Design and configure a Cloud Firestore database structure to store users and their shopping lists.

Deliverables:

- A Flutter app that is successfully connected to a Firebase project.
- A fully functional user registration and login system.
- A Firestore database with security rules configured.

Week 3: Real-Time CRUD and Testing

Tasks:

- **Real-Time Functionality**: Implement the core Create, Read, Update, and Delete (CRUD) operations for shopping lists and items. Use Firebase's real-time capabilities to ensure that UI updates instantly when the database changes.
- **State Management**: Manage the application's state effectively as data flows from Firebase to the UI.
- **Functional Documentation & Unit testing**: Write **Unit tests** for your core business logic (e.g., functions that add or format data). Begin writing simple **Functional Documentation** for your code.

Deliverables:

- Users can create, view, and delete their shopping lists.
- Users can add, edit, and check off items in a list, with changes reflected in real-time.
- A suite of passing unit tests for your core logic.

Week 4: Collaboration, Finalizing, and Deployment

Tasks:

- **Implement Sharing**: Add the functionality for a user to invite another registered user to collaborate on a shopping list.
- **Final Testing**: Perform thorough end-to-end testing of all features, including the real-time collaboration.

- **Mobile Deployment for android and ios**: Follow the official guides to prepare the application for release. This includes creating app icons, generating signed release builds (APK/App Bundle for Android, IPA for iOS), and taking screenshots for the app stores.

Deliverables:

- A working list-sharing feature.
- A final, tested, and high-quality cross-platform application.
- Release-ready builds for both the Google Play Store and Apple App Store.
- Completed project documentation.