Note-Taking App Documentation

App Overview

This is a note-taking app built with Flutter that allows users to sign up, log in, and manage their notes. The app uses Firebase for authentication and Firestore for storing user-specific notes.

Features:

- **User Authentication**: Users can sign up with their email and password, log in, and log out.
- **Add, Edit, and Delete Notes**: Once logged in, users can create new notes, update existing ones, and delete notes.
- **Firestore Database**: Notes are stored in Firestore, associated with the authenticated user via their unique `userId`.

Firebase Setup

1. **Create a Firebase Project**:
    - Go to the Firebase Console.
    - Click on **Add Project** and follow the setup steps (name your project, set up Google Analytics, etc.).
    - After creating the project, you will be redirected to your Firebase project dashboard.
2. **Add Firebase to Your Flutter App**: Follow these steps to integrate Firebase into your Flutter project:
    - **Set up Firebase for iOS**:
        - In the Firebase console, select **Project Settings** (gear icon) > **Project Settings**.
        - Under **Your apps**, select **iOS** to add Firebase to your iOS app.
        - Register your iOS app with your **bundle ID**.
        - Download the `GoogleService-Info.plist` file.
        - Place this file in your Flutter project in `ios/Runner/`.
    - **Set up Firebase for Android**:
        - In the Firebase console, select **Project Settings** (gear icon) > **Project Settings**.

        - Under **Your apps**, select **Android** to add Firebase to your Android app.
        - Register your Android app with your **package name**.
        - Download the `google-services.json` file.
        - Place this file in your Flutter project in `android/app/`.
3. **Update Firebase SDK**: Add the Firebase SDK to your Flutter app by modifying your project's dependencies.

   In your `pubspec.yaml` file, add the following dependencies:

```
dependencies:
  flutter:
    sdk: flutter
  firebase_core: ^latest_version
  firebase_auth: ^latest_version
  cloud_firestore: ^latest_version
  provider: ^latest_version
```

   - Run `flutter pub get` to install the packages.

4. **Initialize Firebase in Your App**: In the `main.dart` file, initialize Firebase before running the app:

```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'my_app.dart'; // Replace with your actual app entry point

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(MyApp());
}
```

5. This ensures that Firebase is initialized before the app starts.

6. **Enable Firebase Authentication**:

   - In the Firebase Console, go to **Authentication** > **Sign-in method**.
   - Enable **Email/Password** sign-in provider.

7. **Set Up Firestore Database**:

   - In the Firebase Console, go to **Firestore Database** and create a Firestore database.
   - Choose **Start in test mode** for initial development (Note: Change the rules before production).

Configure Firestore Rules: To ensure that only authenticated users can read/write their notes, configure the following Firestore rules:

```
 rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

    // Rule for the "notes" collection
    match /notes/{noteId} {

      // Allow read and write only if the user is authenticated and owns the note
      allow read, write: if request.auth != null && request.auth.uid == resource.data.userId;

      // Alternatively, only allow writing a new note if the user is authenticated
      allow create: if request.auth != null;
    }
  }
}
```

Test the Firebase Setup:

- After configuring Firebase Authentication and Firestore, run the app on an emulator or real device.
- Ensure that you can sign up, log in, add, edit, and delete notes, and that each user's notes are stored securely in Firestore.

Firestore Security Rules

To ensure that only authenticated users can access and modify their own notes, we use the following Firestore security rules:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

    // Rule for the "notes" collection
    match /notes/{noteId} {

      // Allow read and write only if the user is authenticated and owns the note
      allow read, write: if request.auth != null && request.auth.uid == resource.data.userId;

      // Alternatively, only allow writing a new note if the user is authenticated
      allow create: if request.auth != null;
```

```
  }
 }
}
```

Explanation of Firestore Rules:

- **Authentication Check**: `request.auth != null` ensures that only authenticated users can read or write data.
- **User-specific Access**: `request.auth.uid == resource.data.userId` ensures that users can only access and modify their own notes. A note's `userId` field must match the current authenticated user's UID.
- **Create Rule**: The `create` rule allows authenticated users to create new notes, but only if they are signed in.

App Flow and Screens

1. **Splash Screen**: On app launch, the splash screen checks if the user is logged in. If the user is logged in, they are directed to the Home screen; otherwise, they are taken to the Login or Signup screen.

2. **Login Screen**:
    - Users can log in using their email and password. Upon successful login, they are redirected to the Home screen.
3. **Signup Screen**:
    - Users can create a new account using their email and password. Upon successful sign-up, the user is logged in automatically and redirected to the Home screen.
4. **Home Screen**:
    - Displays a list of the user's notes retrieved from Firestore.
    - Users can add a new note, edit existing notes, or delete notes.
    - The notes are fetched from Firestore using the current user's UID.
5. **Add/Edit Note Screen**:
    - Users can add a new note or edit an existing one. The note's title and description are saved to Firestore, and changes are reflected in the Home screen.
6. **Logout**:

- Users can log out, which clears their session and returns them to the Login screen.