**Notes Making Application**

**Flutter Bootcamp Project**

**Submitted by:**

**Anza Tamveel**

Table of Contents

[1. Application Overview: 3](#_Toc189173798)

[2. FireStore: 3](#_Toc189173799)

[2.1 Firestore Setup: 3](#_Toc189173800)

[3. App Structure 3](#_Toc189173801)

[4. Firebase Authentication: 4](#_Toc189173802)

[5. Firebase Firestore: 5](#_Toc189173803)

[6. Firestore Security Rules: 5](#_Toc189173804)

[**Explanation of Firestore Rules:** 6](#_Toc189173805)

[7. Features and Functionality: 6](#_Toc189173806)

[7.1 User Authentication 6](#_Toc189173807)

[7.2 Notes Management 6](#_Toc189173808)

[7.3 Search and Filter 6](#_Toc189173809)

[8. Dependencies: 7](#_Toc189173810)

[9. Wire Frames: 7](#_Toc189173811)

[9. Conclusion: 9](#_Toc189173812)

# 1. Application Overview:

This Note-Taking App allows users to create, edit, delete, and search for notes. Each note can be categorized (e.g., Work, Personal, Study), and users can filter notes by title or category. The app uses Firebase for user authentication and Firestore for storing and managing user data (notes).

Users can:

* **Create** a new note with a title, content, and category.
* **Edit** existing notes.
* **Delete** notes.
* **Search** notes by title.
* **Filter** notes by category (e.g., Work, Personal, Study).

# 2. FireStore:

To make this app functional, Firebase services are utilized for **authentication** and **Firestore** database storage.

## 2.1 Firestore Setup:

Firestore is used to store and retrieve the notes for each user. Each note includes a title, content, and category. Firestore collections and documents are structured as follows:

* **Users collection**: Stores each user's document, identified by their uid (unique user ID).
* **Notes sub-collection**: Each user document contains a sub-collection named notes, which holds individual note documents with fields title, content, and category.

# 3. App Structure

The app follows an MVVM (Model-View-ViewModel) architecture, ensuring a clear separation of concerns between the UI and business logic.

1. **models/note.dart:**

* Defines the Note model, representing a note with properties like noteId, title, content, category, userId, and createdAt.

1. **views:**

* Contains all the screens of the app.
* add\_note\_screen.dart: A screen for creating new notes.
* edit\_note\_screen.dart: A screen for editing existing notes.
* home\_screen.dart: The main screen displaying a list of notes with filtering and search functionalities.
* login\_screen.dart and signup\_screen.dart: Screens for user authentication.
* note\_view\_nodel.dart: View for displaying and managing note data.

1. **view\_models:**

* Holds the business logic for authentication (auth\_view\_model.dart) and note management (note\_view\_model.dart).

1. **widgets:**

* note\_item.dart: A widget for displaying individual notes in the list.

1. **services:**

* auth\_service.dart: Handles Firebase Authentication (sign-in, sign-up, and sign-out).
* firestore\_service.dart: Handles Firebase Firestore CRUD operations (creating, fetching, updating, deleting notes).

1. **main.dart:**

* The entry point of the app, where the app is initialized and routed.

# **4.** **Firebase Authentication:**

The app uses Firebase Authentication for user sign-up and login. Firebase Authentication manages users' sign-in credentials securely.

# 5. **Firebase Firestore:**

Firebase Firestore is used for storing and managing notes. Each note document contains the fields:

* title: The title of the note.
* content: The content of the note.
* category: The category of the note (e.g., 'Work', 'Personal', or 'Study').
* userId: The ID of the user who created the note.
* createdAt: Timestamp of when the note was created.

# 6. Firestore Security Rules:

Here are the Firestore security rules that ensure user access control:

**Firestore Rules for User Access:**

service cloud.firestore {

match /databases/{database}/documents {

// Rule for user-specific notes collection

match /users/{userId}/notes/{noteId} {

// Allow read/write if the user is authenticated and the note belongs to them

allow read, write: if request.auth != null && request.auth.uid == userId && request.resource.data.userId == request.auth.uid;

}

// General rule to allow authenticated users to read/write to their data

match /{document=\*\*} {

allow read, write: if request.auth != null;

}

}

}

### **Explanation of Firestore Rules:**

* **/users/{userId}**: This rule ensures that only the authenticated user with a matching userId can read, write, or update their own document.
* **/users/{userId}/notes/{noteId}**: This rule ensures that only the authenticated user can read, create, update, or delete notes within their notes sub-collection.
* **request.auth != null**: Ensures that the request is coming from an authenticated user.
* **request.auth.uid == userId**: Ensures that users can only access their own notes and not another user's data.

# **7. Features and Functionality:**

## 7.1 User Authentication

* Users sign in using Firebase Authentication.
* After authentication, each user is identified by their unique uid, which is used to store and retrieve their notes.

## 7.2 Notes Management

* **Create a Note**: Users can create new notes by providing a title, content, and category (e.g., Work, Personal, Study). Notes are saved in the user's Firestore sub-collection notes.
* **Edit a Note**: Users can edit the title, content, and category of their existing notes. The app navigates to an EditNoteScreen where they can update the note.
* **Delete a Note**: Users can delete any note they have created. Once deleted, the note is removed from Firestore permanently.

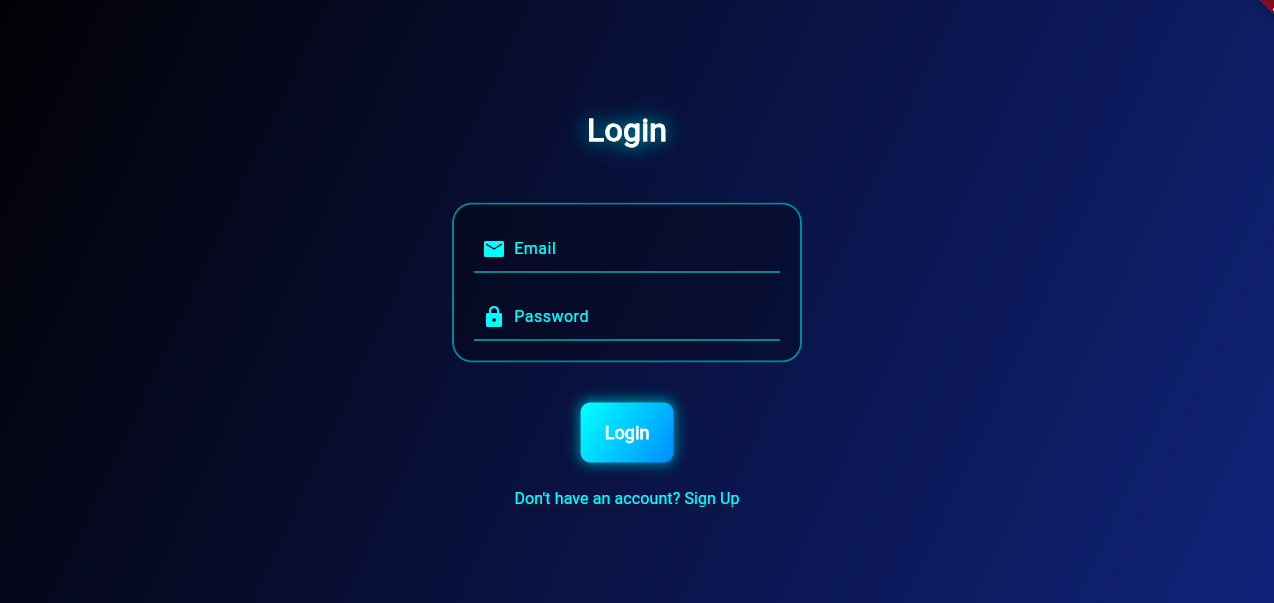
## 7.3 Search and Filter

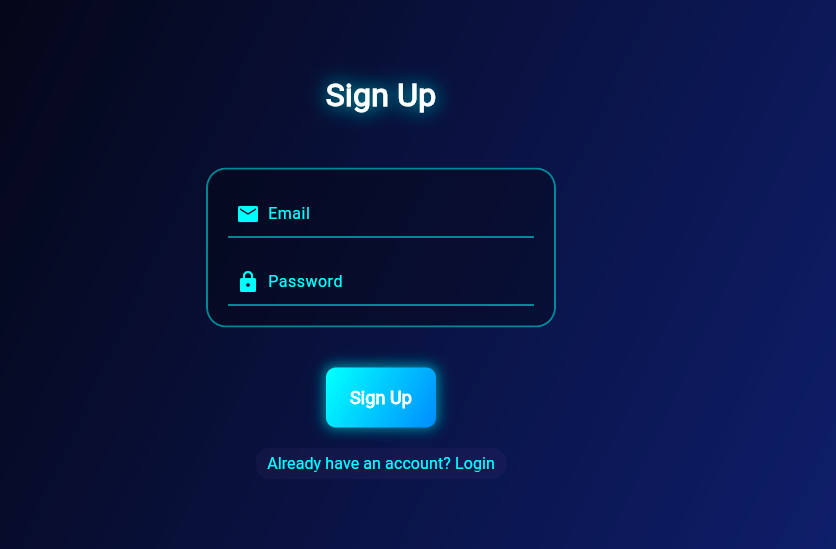
* **Search by Title**: Users can search for notes by typing keywords into a search field. The app filters notes that match the search criteria based on the title.
* **Filter by Category**: Users can filter notes based on predefined categories (Work, Personal, Study). The app will only display notes that match the selected category.

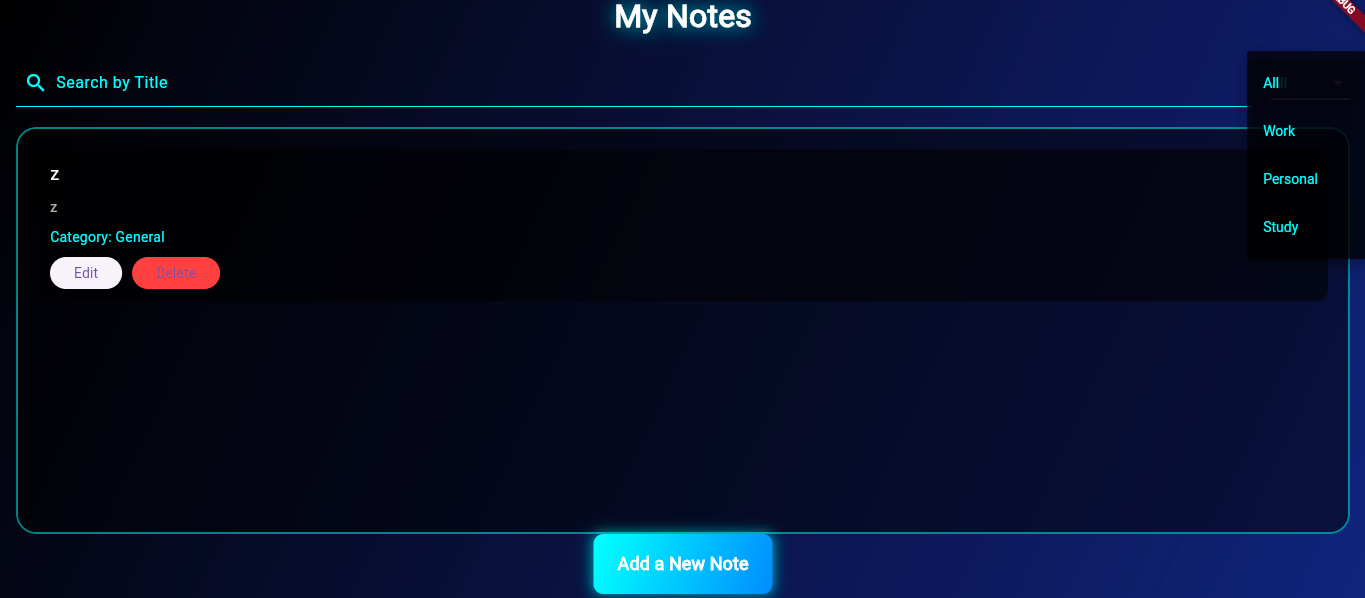
# 8. Dependencies:

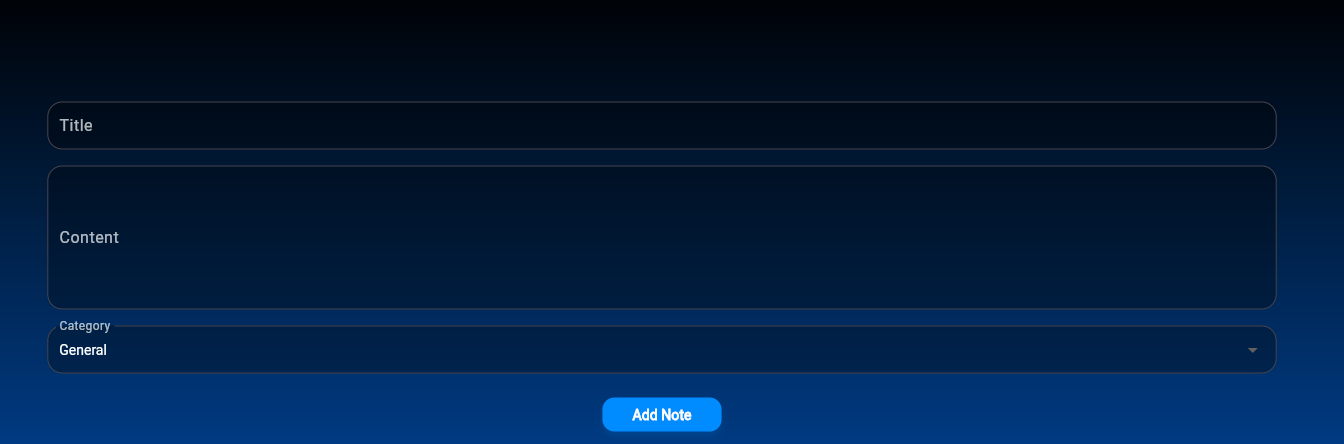
* **Firebase Core**: Initializes Firebase in the app.
* **Firebase Auth**: Handles user authentication.
* **Cloud Firestore**: Stores and manages user data.
* **Flutter**: The main framework for building the app's user interface.

# 9. Wire Frames:









# 9. **Conclusion:**

This Note-Taking App leverages Firebase Authentication and Firestore to provide a secure and efficient way for users to manage their notes. The app allows users to create, edit, delete, and search notes, with robust security rules ensuring that each user can only access their own data.