CMPS 312 Mobile App Development
Lab 11 – Firebase

Objective

In this Lab, you will continue with the **Todo app** that you created in Lab 10 and replace the local SQLite database with **Cloud Firestore;** which is a NoSQL document-oriented database that lets you easily store and query data for your mobile apps at global scale.

In this Lab you will practice:

- Read and write data to Firestore from a Flutter app.
- Query Firestore collections.
- Listen real-time changes of Firestore data.
- Integrate Firebase APIs into an MVVM architecture.
- User Authentication using Firebase email/password Authentication.

The image below shows MVVM architecture with Cloud **Firestore** that could be used as an alternative of complement a local database and Web API.

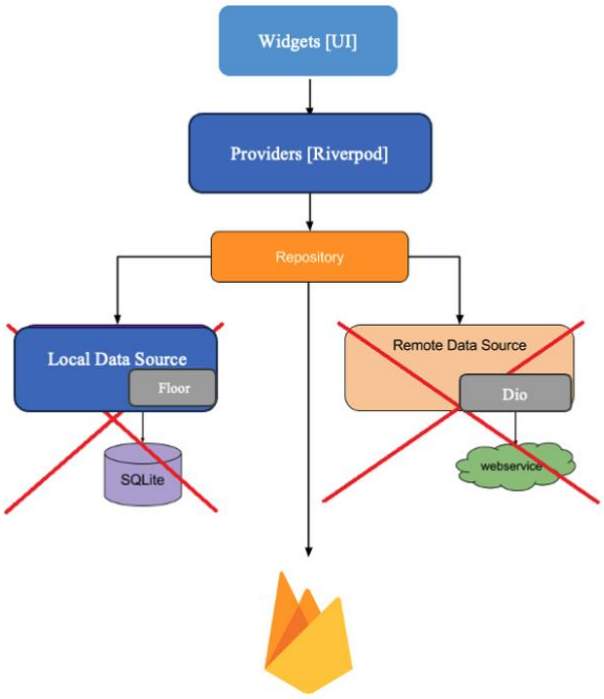


Figure 1 MVVM architecture with Cloud Firestore

Preparation

1. Sync the Lab GitHub repo and copy the **Lab 11-Cloud Firestore** folder into your repository.
2. Open the TodoList project in VS Code. You will see some compilation errors or warning messages. You will correct this in the next sections.

PART A: Refactor the Todo App to use Firestore

In this lab you will refactor the Todo app you created in Lab 10 and make the necessary changes to replace the local database with Firestore. The functionality of the app will remain the same including get, add, update delete projects and to-dos.

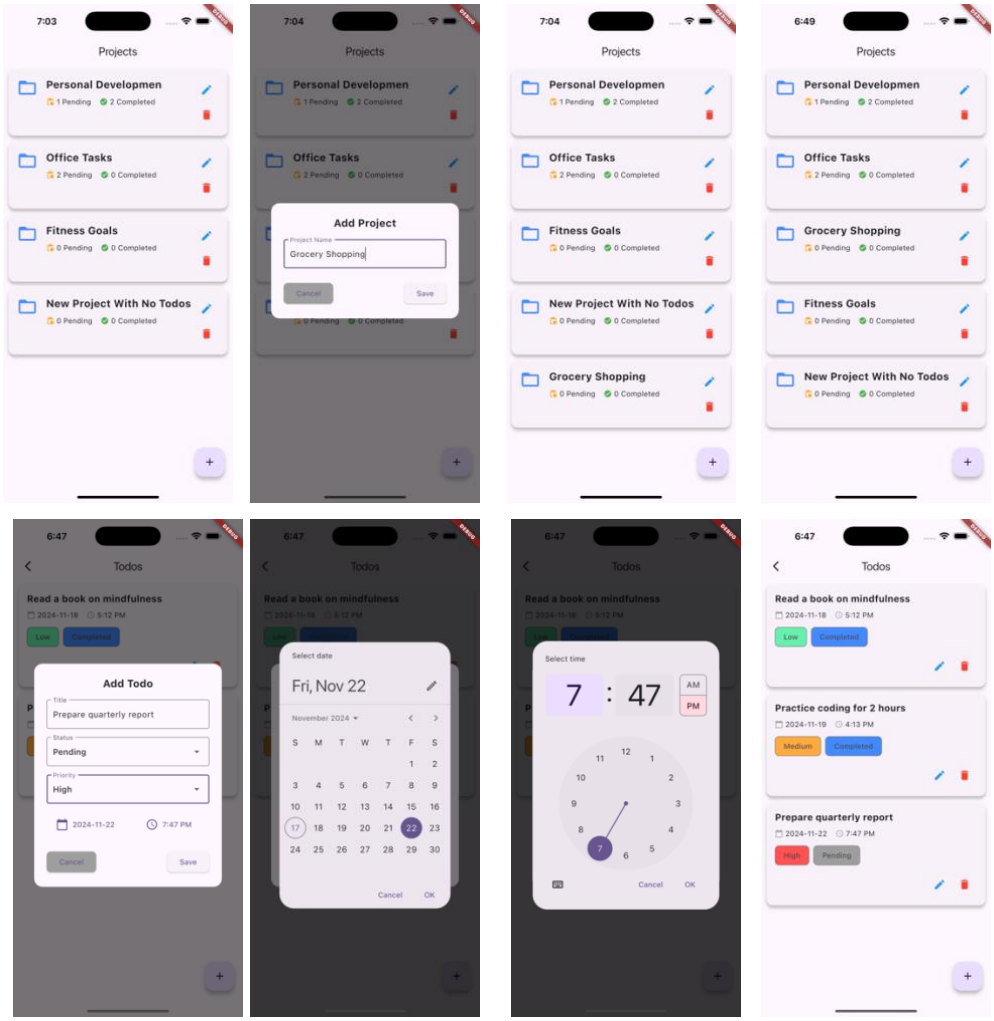


Figure 2 TodoList App

Task 1 : Setting up Firebase in your Flutter project

The following steps help you set up Firebase in your Flutter app, preparing it to use Firebase services such as Firestore, Authentication, etc.

A. Step 1: Create Firebase Project Online

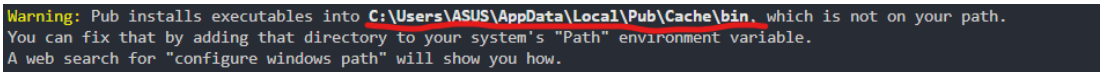
a. Sign in to your google account
b. Go to the Firebase Console: https://console.firebase.google.com
c. Click on "Create Project" enter a project name (e.g., todolist) and click "Continue." To keep it simple disable Google Analytics.

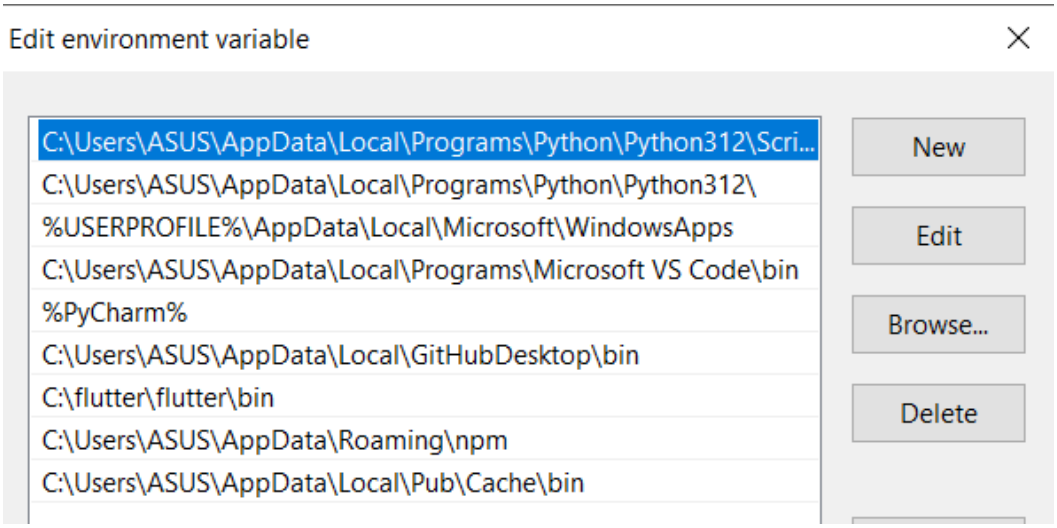
Step 2 : Install Firebase CLI Firebase CLI (Command Line Interface) to manage Firebase projects from your terminal.

a. Download Node.js from nodejs.org.
b. Follow the installation steps for your operating system. Ensure you include npm (Node Package Manager) during installation.
c. **Install Firebase CLI:** Open a terminal and run:

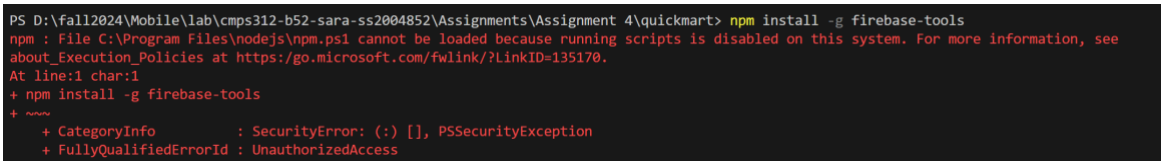
```
npm install -g firebase-tools
```

Error that might occur for some of you





Add to environmental variable



Solution

- Open PowerShell as administrator
- run **Set-ExecutionPolicy RemoteSigned OR Set-ExecutionPolicy Unrestricted**
- Run this command again " npm install -g firebase-tools "

d. Verify the installation by checking the version:

```
firebase --version
```

e. Login to firebase by running the command below. This command will open a browser window to authenticate with your Google account. After successful login, return to the terminal.

```
firebase login
```

B. Step 3: Activate FlutterFire CLI

a. Install the FlutterFire CLI by running:

```
dart pub global activate flutterfire_cli
```

C. Step 4 : Connect Your Flutter Project with Firebase

a. Make sure you are the root of your project ***cd /path-to-your-flutter-project***

b. Run the FlutterFire CLI to link your Firebase project:

```
flutterfire configure
```

c. Select your Firebase project and the platforms (e.g., Android, iOS, Web) you want to configure.

d. This will generate a **firebase_options.dart** file in your Flutter project under the lib/ directory. It contains all the Firebase configuration required for your app.

e. Add the cloud firestore dependency to your project

```
flutter pub add cloud_firestore firebase_core
```

f. In your lib/main.dart file, initialize Firebase using the DefaultFirebaseOptions object exported by the configuration file:

```
WidgetsFlutterBinding.ensureInitialized();
await Firebase.initializeApp(
  options: DefaultFirebaseOptions.currentPlatform,);
runApp(const MyApp());
```

Task 2: Write , Read and Query Data from Cloud Firestore Database

In this section we will implement add/update/delete/query todo list data using **Firestore**.
Note that it is possible to add document data manually using the Firebase console.

1. Implement `TodoRepository` class that call the methods on `projectRef` and `todoRef` to read/write data from `projects` and `todos` Firestore collections.

```
class TodoListRepo {
  final CollectionReference projectRef;
  final CollectionReference todoRef;

  TodoListRepo({required this.projectRef, required this.todoRef});
```

2. Implement the repository methods by calling the corresponding `projectRef` and `todoRef` functions. For the repository have the following methods:

```
// Projects
  Stream<List<Project>> observeProjects(); //get a real time data
  Future<Project?> getProjectById(String id);
  Future<void> addProject(Project project);
  Future<void> updateProject(Project project);
  Future<void> deleteProject(Project project);

  // Todos
  Stream<List<Todo>> observeTodos(String projectId);
  Future<Todo> getTodoById(int id);
  Future<void> addTodo(Todo todo);
  Future<void> updateTodo(Todo todo);
  Future<void> deleteTodo(Todo todo);
```

```
  // Project Todos Status Counts
  Stream<ProjectTodoStatusCounts?> getProjectTodosStatusCounts(String
projectId);
```

3. Create a TodoList Repository Provider. This provider should instantiate the Firestore database and then return an instance of the `TodoListRepo`.

```
final todoListRepoProvider = FutureProvider<TodoListRepo>((ref)
async {
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  var db = FirebaseFirestore.instance;
  var projectRef = db.collection('projects');
  var todoRef = db.collection('todos');
  return TodoListRepo(projectRef: projectRef, todoRef: todoRef);

});
```