

**Lab # 10**

**Mobile Application Development  
Fall 2022**

****

|  |  |
| --- | --- |
| Instructor | Bakht Muhammad |
| Student Name | Jawad Ahmed Tareen |
| CMSID | 411342 |
| Department | Computer Science |
| Semester | 5th |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Lesson Set 10** | **Adding Support for Remote Storage and Databases**  **Using Firebase** | | | |
| **Purpose** | 1. Integrate Firebase into a Flutter app. 2. Perform CRUD operations using Cloud Firestore. 3. Use Firebase Storage to upload and retrieve files. | | | |
| **Procedure** | 1. Students should read the Pre-lab Reading assignment before coming to the lab. 2. Students should complete the Pre-lab Writing assignment before entering the lab. 3. In the lab, students should complete Labs 10.1 through 10.4 in sequence. Your instructor will give further instructions on grading and completing the lab. 4. Students should complete the set of lab tasks before the next lab and get them checked by their lab instructor. | | | |
|  | **Contents** | **Pre-requisites** | **Completion Time** | **Page Number** |
|  | Pre-lab Reading Assignment | - | 20 min | 3 |
|  | Pre-lab Writing Assignment | Pre-lab Reading | 10 min | 7 |
|  | **Lab 10** | | | |
|  | **Lab 10.1**  Connecting to firebase | Pre-lab reading | 30 min | 5 |
|  | **Lab 10.2**  Lab Tasks | Awareness of Programming | - | 8 |

|  |  |
| --- | --- |
| **PRE-LAB READING ASSIGNMENT** | |
| **Introduction to Firebase** | Firebase is a platform by Google that provides a suite of cloud-based services to support app development. It includes features like real-time databases, cloud storage, authentication, and analytics. Firebase's cloud services eliminate the need to set up your backend infrastructure, enabling rapid development and deployment.  Firebase and Its Components  **Firebase Components Used in This Lab**   1. **Firebase Core:** The foundation required to connect your app to Firebase. 2. **Cloud Firestore:** A NoSQL document database that allows you to store, sync, and query data for mobile and web apps. 3. **Firebase Authentication:** A service that provides easy sign-in methods to ensure secure user-specific data handling. 4. **Firebase Storage:** A cloud-based storage service for managing user-uploaded files like images or documents. |
| **Advantages of Firebase for App** | * **Scalability:** Automatic scaling with your user base. * **Cross-Platform Support:** Works with Android, iOS, and web apps. * **Real-Time Data Sync:** Synchronize data seamlessly across devices. * **Offline Capabilities:** Firestore stores data locally when offline and syncs it when back online. * **Ease of Use:** Simple APIs and integration with tools like Flutter. |
| **Core Concepts in Firestore** | **Documents and Collections:** Data in Firestore is stored as **documents** organized into **collections**. A document is a set of key-value pairs.    **Example:**  Collection: posts  Document: {  id: "1",  title: "My First Post",  body: "This is the content of my first post."  }  **CRUD Operations:**   * **Create:** Adding new documents. * **Read:** Retrieving documents or collections. * **Update:** Modifying existing documents. * **Delete:** Removing documents.   **Offline Data Sync:** Firestore supports offline access by caching data locally. |
| **Integrating Firebase** | **Steps to Integrate Firebase with Flutter**  **Step 1: Register Your App**   * Go to [Firebase Console](https://console.firebase.google.com/). * Create a new project or select an existing one. * Add your app by clicking on "Add App" (Android or iOS). * Follow the instructions to download the google-services.json (for Android) or GoogleService-Info.plist (for iOS) file.   **Step 2: Configure Your Flutter Project**   * Add the google-services.json file to your Android project under android/app/. * Add the GoogleService-Info.plist file to your iOS project under ios/Runner/.   **Step 3: Update** pubspec.yaml **with Firebase Packages**   * Add the necessary dependencies:   dependencies:  firebase\_core: ^2.17.0  cloud\_firestore: ^4.10.0  firebase\_auth: ^4.8.0   * Run flutter pub get to fetch the packages.   **Step 4: Initialize Firebase**   * Update the main.dart file:   import 'package:firebase\_core/firebase\_core.dart';  import 'package:flutter/material.dart';  void main() async {  WidgetsFlutterBinding.ensureInitialized();  await Firebase.initializeApp();  runApp(MyApp());  }  class MyApp extends StatelessWidget {  @override  Widget build(BuildContext context) {  return MaterialApp(  title: 'Firebase Demo',  home: HomeScreen(),  );  }  } |
| **CRUD Operations in Firestore** | **Create (Add Data):** Add a new document to a collection:  import 'package:cloud\_firestore/cloud\_firestore.dart';  Future<void> addData(String collectionName, Map<String, dynamic> data) async {  await FirebaseFirestore.instance.collection(collectionName).add(data);  }  **// Example usage:**  await addData('posts', {'title': 'New Post', 'bo1dy': 'This is a post.'});  **Read (Retrieve Data):** Fetch all documents in a collection:  Future<List<QueryDocumentSnapshot>> fetchData(String collectionName) async {  final QuerySnapshot snapshot =  await FirebaseFirestore.instance.collection(collectionName).get();  return snapshot.docs;  }  **// Example usage:**  final posts = await fetchData('posts');  for (var post in posts) {  print(post.data());  }  **Update (Modify Data):** Update a specific document  Future<void> updateData(String collectionName, String docId, Map<String, dynamic> updatedData) async {  await FirebaseFirestore.instance  .collection(collectionName)  .doc(docId)  .update(updatedData);}  **// Example usage:**  await updateData('posts', 'postId', {'title': 'Updated Title'});  **Delete (Remove Data)**: Delete a document from a collection  Future<void> deleteData(String collectionName, String docId) async {  await FirebaseFirestore.instance.collection(collectionName).doc(docId).delete();  }  **// Example usage:**  await deleteData('posts', 'postId'); |

|  |  |
| --- | --- |
| **PRELAB WRITING ASSIGNMENT** | |
| **Fill in the blanks** | 1. Firebase provides a suite of cloud-based services for **mobile and web** app development. 2. The Firebase component used for storing and syncing data in real-time is **Firebase Realtime Database**. 3. A document in Firestore is a set of **key-value** pairs stored in a collection. 4. To connect a Flutter app with Firebase, the package **firebase\_core** is required. 5. Firestore supports offline access by **caching** data locally when offline and syncing it when back online. |

|  |  |
| --- | --- |
| **Lab 10.2** | **Lab Tasks** |

**Task 1: Add Firebase to a Flutter App**

**Setup Firebase**

* Open the Firebase console and create a new project.
* Register your app (for both Android and iOS if applicable).
* Download and place the necessary configuration files (google-services.json or GoogleService-Info.plist) in your project.

**Update Dependencies**: Add the following dependencies in your pubspec.yaml:

**dependencies:**

firebase\_core: ^2.17.0

cloud\_firestore: ^4.10.0

firebase\_auth: ^4.8.0

firebase\_storage: ^11.0.0

* Run flutter pub get to install the packages.

**Initialize Firebase**: Update your main.dart:

import 'package:firebase\_core/firebase\_core.dart';

import 'package:flutter/material.dart';

void main() async {

WidgetsFlutterBinding.ensureInitialized();

await Firebase.initializeApp();

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Firebase App',

home: PostScreen(),

);}}

|  |
| --- |
| * Screenshot should be pasted here.        * GitHub Repository link. |

**Task 2: Design a Social Media App UI with Image Upload and Download Features**

The objective of this task is to design a User Interface (UI) for a basic social media app. The app should allow users to upload, update, read, and delete posts, with an additional feature to include an image in the post and download it by long-tapping.

**Upload Screen**:

* Create a screen to allow users to add a new post.
* Include the following elements:
  + A **text field** for the post title.
  + A **text field** for the post description.
  + A **button to select an image** from the device gallery.
  + A preview of the selected image.
  + An **"Upload" button** to save the post.

**Read (Feed) Screen**:

* Display a list of posts, including:
  + **Post title**.
  + **Post description**.
  + **Post image** (displayed as a thumbnail).
* Add a **delete icon (**trash **icon)** beside each post to allow deletion of a specific post.
* Implement a **long-tap gesture** on the post image to download it to the device. Display a snackbar or toast notification to indicate that the image has been downloaded.

**Update Screen**:

* Include an **"Edit" button** next to each post in the feed.
* When the "Edit" button is tapped, open a screen with:
  + Pre-filled fields for the post title and description.
  + An option to change the image.
  + An **"Update" button** to save changes.

**Delete Functionality**:

* On tapping the delete icon, show a confirmation dialog:
  + "Are you sure you want to delete this post?"
  + Include **"Yes"** and **"No"** buttons.

**Download Image**:

* Implement a **long-press gesture** on the image in the feed to save it to the device's storage.
* Provide feedback, such as a **snackbar notification**:
  + "Image downloaded successfully!"

**Guidelines**

1. Use **Flutter widgets** to design the app, including:
   * **ListView** for the feed screen.
   * **ImagePicker package** to select images from the gallery.
   * **GestureDetector** for handling long-press gestures.
   * **SnackBar** or **Toast** for feedback.
   * **Dialog** for delete confirmation.
2. Keep the design user-friendly and responsive.
3. Ensure that:
   * Images are displayed properly in the feed (e.g., as thumbnails).
   * Full-size images are downloaded when long-tapped.

|  |
| --- |
| * Screenshot should be pasted here. * GitHub Repository link. |

**Note:** Make sure to upload the code for each task to a GitHub repository. Do not update or change the repository after the due date. Provide the link to your GitHub repository with each task submission.