TIERS LIMITED SUMMER INTERNSHIP 2024

MOBILE APP DEVELOPMENT

Firebase Database

CLASS # 19

Overview

In this class, we will learn how to use Firebase Realtime Database in a Flutter application to manage student data. The Firebase Realtime Database allows you to store and sync data between users in real-time.

Objectives

- Understand the basics of Firebase Realtime Database.
- Learn how to perform CRUD (Create, Read, Update, Delete) operations on student data.
- Integrate Firebase Realtime Database into a Flutter app.

Steps to Integrate Firebase Realtime Database

1. **Set Up Firebase Project**
   - Follow the steps from the previous class to set up Firebase in your Flutter project.
2. **Add Firebase Realtime Database Dependency**

   In your **`pubspec.yaml`** file, add the Firebase Realtime Database dependency:

   ```
   dependencies:
     firebase_core: latest_version
     firebase_database: latest_version
   ```

3. **Student Class**

   Student class is same that we used in our previous lectures.

4. **Database Service**

   Create a service class to handle all database operations:

```
final DatabaseReference _database =
                FirebaseDatabase.instance.ref().child('students');

        Future<void> addStudent(Student student) async {
          await _database.child(student.id.toString()).set(student.toJson());
        }

        Future<void> updateStudent(String id, Student student) async {
          await _database.child(id).update(student.toJson());
        }

        Future<void> deleteStudent(String id) async {
          await _database.child(id).remove();
        }

        DatabaseReference getStudents() {
          return _database; }
```

Reading and Writing Data

Update your screen to interact with the Firebase Realtime Database:

```
final DatabaseService _databaseService = DatabaseService();
late List<Student> _students;
bool _isLoading = true;

@override
void initState() {
  super.initState();
  _fetchStudents();
}

Future<void> _fetchStudents() async {
  _databaseService.getStudents().onValue.listen((event) {
    List<Student> students = [];
    final data = event.snapshot.value;
    if(data != null && data is List)
      students = data .where((element) => element != null).map((e) =>
       Student.fromJson(Map<String, dynamic>.from(e))).toList();
    setState(() {
      _students = students;
      _isLoading = false;
    });
  });
}

void _addStudent() {
   final student = Student(
     id: _students.length+1,name: 'Ahmad', age: 20, semester:'7th'
   );
   _databaseService.addStudent(student);
   _fetchStudents();
}

void _editStudent(Student student) {
   final updatedStudent = Student(
        id: student.id,name: 'Umar', age: 21, semester: '7th'
   );
   _databaseService.updateStudent(.id.toString(), updatedStudent);
   _fetchStudents();
}

 _isLoading
  ? Center(child: CircularProgressIndicator())
  : ListView.builder(
      itemCount: _students.length,
      itemBuilder: (context, index) {
        return ListTile(
          title: Text(_students[index].name),
          subtitle: Text(
               'Age: ${_students[index].age},
                Semester: ${_students[index].semester}'
          ),
        );
```

```
            },
        ),
```

Key Points:

1. **Initialization and Fetching Students**:
    - In `initState`, `_fetchStudents` is called to listen for changes in the student data and update the UI accordingly.
2. **Adding Students**:
    - The `_addStudent` method adds a new student with hardcoded values. When the add button in the app bar is pressed, this method is called.
3. **Editing Students**:
    - The `_editStudent` method updates the selected student's data with hardcoded values. When a list item is tapped, this method is called.
4. **Deleting Students**:
    - The `deleteStudent` method in the `DatabaseService` is called to delete the student when the delete icon is pressed.

Explanation

- **Initialization**: Initialize Firebase in the `main.dart` file.
- **Student Class**: Define the `Student` class with methods to convert to/from JSON.
- **DatabaseService**: Create a service class to handle all database operations (add, update, delete, fetch).
- **CRUD Operations**: Implement methods to perform CRUD operations on the student data.

Exercises:

- Implement Add, Edit, Remove and Load Operations using Firebase Database.
- Use Dialogs for add and edit operations
- Use Dialog “Are you sure to delete?” Yes or No when user press on delete button.