THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

COMP4521
Mobile Application Development

Lecture 6

Cloud Data Management | Edward Szeto

Lesson Outline

- Firebase Introduction
- Realtime Database
- Cloud Firestore
- Firebase Authentication
- Firebase Cloud Storage

Firebase Introduction

Firebase Introduction

❑What is Firebase?

- A comprehensive platform developed by Google
- Provides a variety of tools and services
- Helps developers build and manage mobile and web applications
- Offers backend services (i.e. Backend as a Service – BaaS)

Firebase Introduction

❑What is Firebase?

https://www.youtube.com/watch?v=O17OWyx08Cg

Firebase Introduction

❑What main services are provided by Firebase?

- Realtime Database
- **Cloud Firestore** recommended to replace the realtime database. a NoSQL DB
- **Firebase Authentication**
- **Cloud Storage** not db, just file storage
- Firebase Cloud Messaging (FCM)
- Firebase Analytics
- Firebase Hosting

Realtime Database

Realtime Database

❑Realtime Database

❑Features

hard to scale a very LARGE json file

- A NoSQL cloud database that allows data to be **stored as JSON** and **synchronized in real-time** across all clients
- Support CRUD Operations

❑Use cases

- Chat Applications
- Collaborative Tools
- Live Dashboards

Cloud Firestore

Cloud Firestore

- **Cloud Firestore**
  - **Features**
    - A **scalable** NoSQL database
    - With **advanced querying capabilities**
    - Organizes data into **collections and documents**

      tables records
  - **Use cases**
    - Social Media Applications
    - E-Commerce Platforms
    - Content Management Systems (CMS)

Cloud Firestore

❑Cloud Firestore

❑Data Structure - Collections and Documents

- Data is organized into collections, each containing multiple documents
- Each document can store various data types

- Example

```
/users (collection)
  /user1 (document)
    - name: "Alice"
    - age: 30
  /user2 (document)
    - name: "Bob"
    - age: 25
```

Cloud Firestore

❑Cloud Firestore

❑CRUD Operations - Create Data

- To add a document to a collection

- Example

```
val db = FirebaseFirestore.getInstance()
val user = hashMapOf(
    "name" to "Alice",
    "age" to 30
)

// Add a new document with a generated ID
db.collection("users")
    .add(user)
    .addOnSuccessListener { documentReference ->
        Log.d("Firestore", "Document added with ID: ${documentReference.id}")
    }
    .addOnFailureListener { e ->
        Log.e("Firestore", "Error adding document", e)
    }
```

Cloud Firestore

❑Cloud Firestore

❑CRUD Operations - Read Data

- To retrieve documents from a collection

- Example

```
db.collection("users")
    .get()
    .addOnSuccessListener { documents ->
        for (document in documents) {
            Log.d("Firestore", "${document.id} => ${document.data}")
        }
    }
    .addOnFailureListener { exception ->
        Log.e("Firestore", "Error getting documents: ", exception)
    }
```

Cloud Firestore

❑Cloud Firestore

❑CRUD Operations - Update Data

- To update an existing document

- Example

```
val userId = "user1" // Replace with the actual document ID
val updates = hashMapOf<String, Any>(
    "age" to 31
)

db.collection("users").document(userId)
    .update(updates)
    .addOnSuccessListener {
        Log.d("Firestore", "Document updated successfully")
    }
    .addOnFailureListener { e ->
        Log.e("Firestore", "Error updating document", e)
    }
```

Cloud Firestore

❑Cloud Firestore

❑CRUD Operations - Delete Data

- To remove a document from a collection

- Example

```
db.collection("users").document(userId)
    .delete()
    .addOnSuccessListener {
        Log.d("Firestore", "Document deleted successfully")
    }
    .addOnFailureListener { e ->
        Log.e("Firestore", "Error deleting document", e)
    }
```

Cloud Firestore

❑Cloud Firestore

❑Advanced Operations – Realtime Update

- To listen for real-time updates, you can set up a listener

- Example

```
db.collection("users")
    .addSnapshotListener { snapshots, e ->
        if (e != null) {
            Log.e("Firestore", "Listen failed.", e)
            return@addSnapshotListener
        }

        for (doc in snapshots!!.documentChanges) {
            when (doc.type) {
                DocumentChange.Type.ADDED -> Log.d("Firestore", "New user: ${doc.document.data}")
                DocumentChange.Type.MODIFIED -> Log.d("Firestore", "Modified user:
${doc.document.data}")
                DocumentChange.Type.REMOVED -> Log.d("Firestore", "Removed user:
${doc.document.data}")
            }
        }
    }
```

Cloud Firestore

❑Cloud Firestore

❑Security Rules

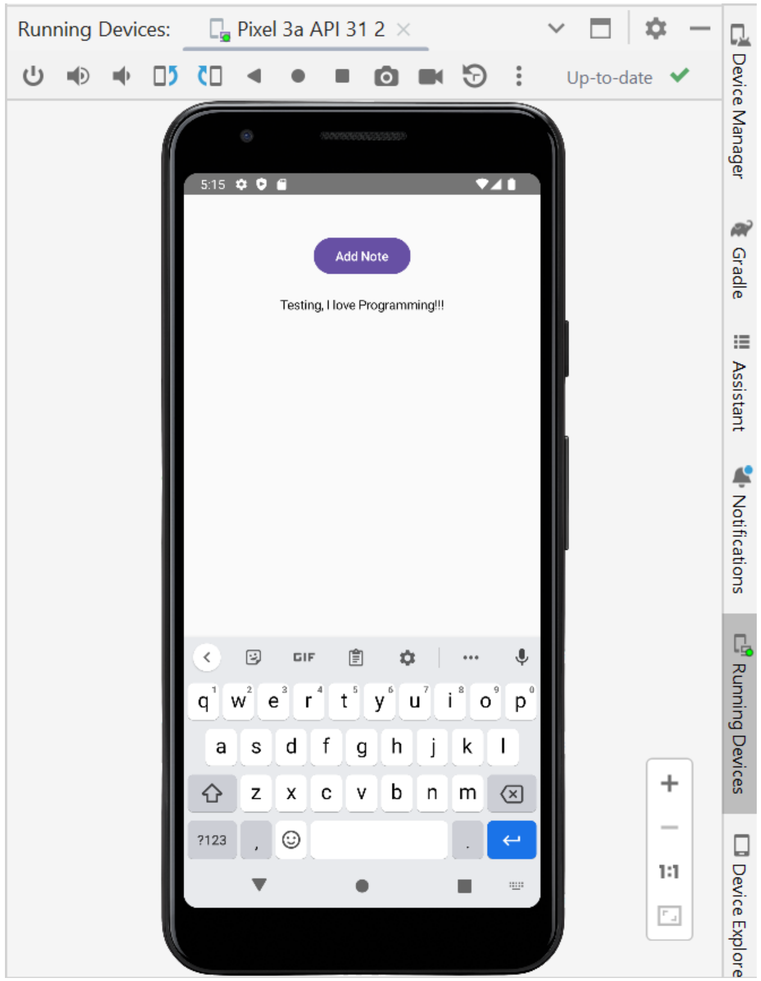
- Define security rules in the Firebase Console to control access

- Example

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{userId} {
      allow read, write: if request.auth != null; // Only
authenticated users can access
    }
  }
}
```

Cloud Firestore

- Demonstration



Firebase Authentication

Firebase Authentication

❑Firebase Authentication

❑Features

- Enables developers to **authenticate** users in **various methods**
- Simplifies the implementation of user authentication
- Integration with Other Firebase Services

❑Use cases

- User Registration and Login
- Social Media Integration
- Access Control