CS394 Mobile Application Development

Lecture 20
Firebase Authentication Service

BSCS 6th Semester Session 2022 RCET

Teacher: Shehzad Aslam

Email: shehzadaslam@uet.edu.pk

Contents

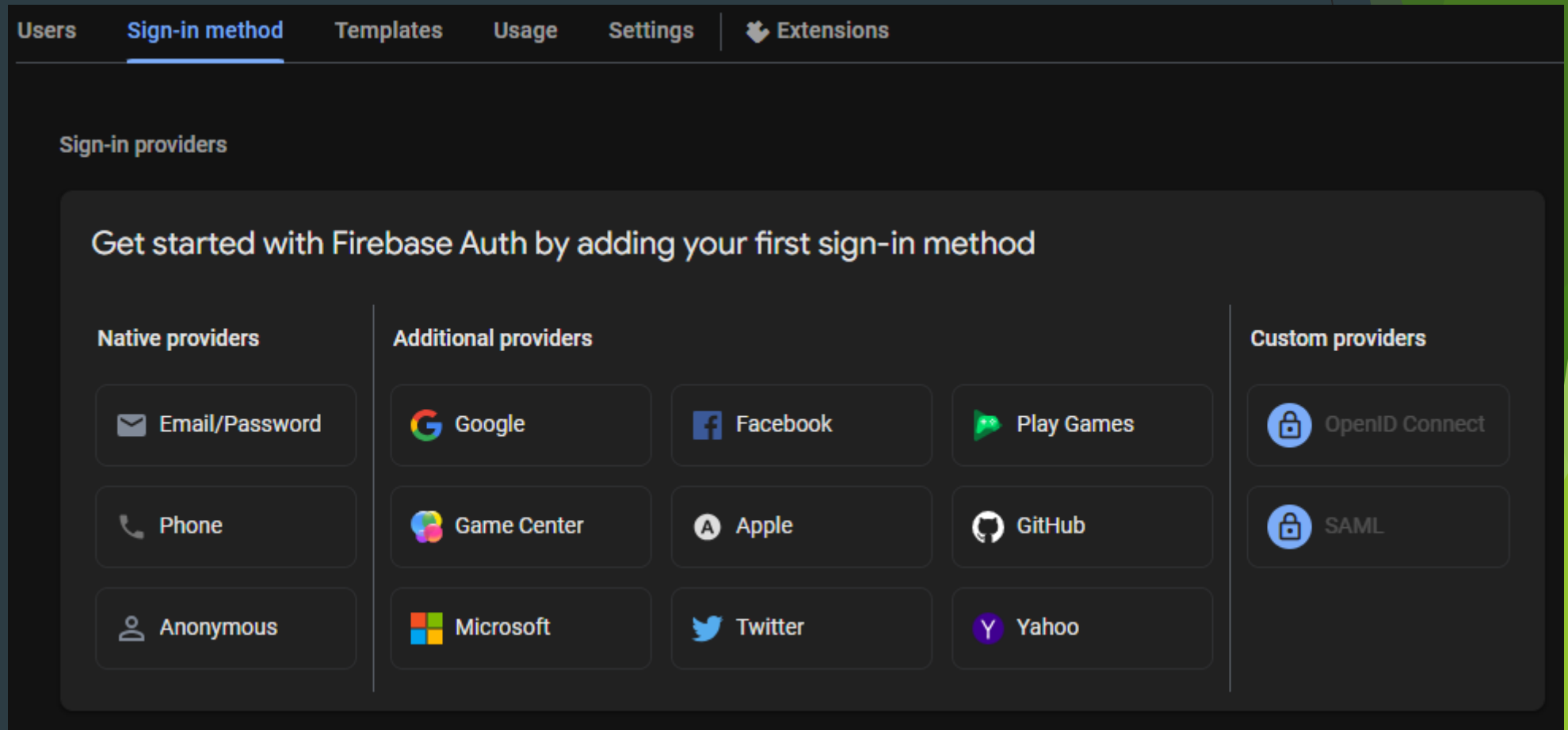
- Signup Backend
- Saving Profile
- Login

Firebase Introduction

- Firebase is backend as service (BaaS)
  - Provide authentication
  - Realtime database
  - File storage
  - Analytics
  - OTPs etc
- Firestore database

Signup – Setting firebase Service

- Goto firebase console, open your project
  - Choose authentication and click get started
  - Enable email password in signin method



Install Package

- Install package in flutter
  - Firebase auth
  - Cloud Firestore //for firestore database
  - Firebase database //if wants firebase database
- After Running command “flutter pub get” packaged will be downloaded

Signup – Flutter backend

- Create a file user_auth.dart in lib>util to manage code
- Create a static method signup that return bool in future and accept details
- Following method create user and return UserCredential
- `FirebaseAuth.instance.createUserWithEmailAndPassword (email: ‘email’, password: ‘pass’);`
- Call this method and wait for to create user
- Wrap in try catch and return appropriate flag
- Can send verification email after signup
- `await userCredential.user!.sendEmailVerification();`

Signup – Flutter backend

```
4   class UserAuth {
5
6     static Future<bool> signupWithEmailPass ({
7       required String email,
8       required String pass,
9     }) async {
10      try {
11        UserCredential cred = await FirebaseAuth.instance
12        .createUserWithEmailAndPassword(email: email, password: pass);
13
14        return true;
15      } catch(e) {
16        print("Some  error ");
17        return false;
18      }
19    }
20  }
```

Modify OnPress Event in UI

- Goto signup ui and call this method on button click
  - Get data from controller and pass it to signup method
  - Check for error and show message or take to login screen
  - Future<bool> f = UserAuth.signupWithEmailPass(email: _cemail.text, password: _cpass.text);

Signup Profile Detail – Flutter backend

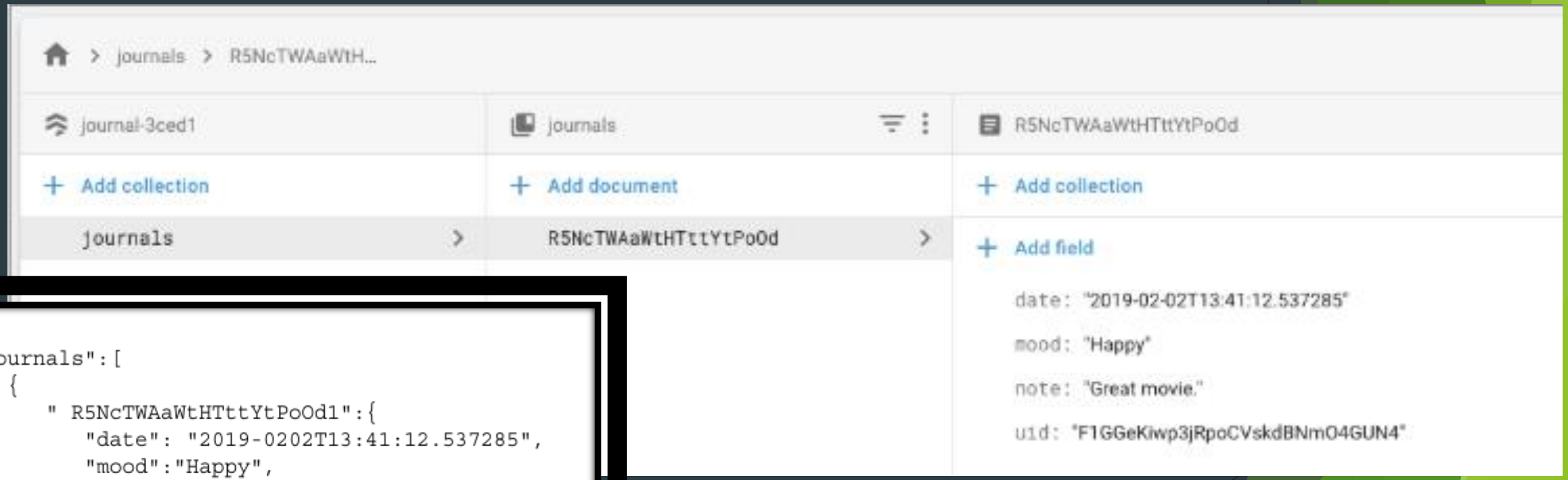
- Remaining user data like city, address etc will be saved in separate collection
- You can use FireStore
  - Document based database
- OR Firebase
  - Json based database

Firebase Database

- Path (branches) to save data e.g. /app/posts/someid
- Always send Map while updating or saving
- Push() create new node in given path and return key
- Set method save values in current path
- Update method update the values in the current path
- remove method is used to delete data on given path
- Get() used to read data once from given path
- Listeners listen to change, add, delete event on given path

Firestore Database

Relational DB	Cloud Firestore
Table	Collection
Row	Document
Fields	Data




```
{
   "journals":[
      {
         " R5NcTWAaWtHTttYtPoOd1":{
            "date": "2019-0202T13:41:12.537285",
            "mood":"Happy",
            "note":"Great movie."
            "uid": " F1GGeKiwp3jRpoCVskdBNmO4GUN4",
         }
      }
   ]
}
```

Creating Firebase DB Object

- Switch to realtime database in firebase console
  - Create database, Start in test mode
- Import Firebase_database package
- We need to name our data, you can think about table
- Suppose we want our tree root “lifehero_db”
- We need Firebase database Object
  - ```
    DatabaseReference db =
    FirebaseDatabase.instance.ref("lifehero_db");
    ```
    ```

Saving Data -

- Push method create new child and return a key
- Call set method by providing a map
- `db.push().set(someMapObject);`
- Goto firebase console in the browser and check that data is added

Catching Errors

- Async way
- Db.push().set(object)

```
.then((_){                    })
.catchError((error){                    });
```

- Then is called at successful save
- Error callback is called upon an error
- Or use traditional try catch way
- try {
- await db.push().set(object);
- } catch(error) { … }

Signup Profile Detail – Flutter backend

- Modify signup function and pass remains data
- Create firebase instance with users/
- Push data with user_id
- User id can be get from credentials
  - `String userId = cred.user!.uid`

Signup Profile Detail - Flutter backend

```
DatabaseReference db = FirebaseDatabase.instance.ref("lifehero_db/users");

// Save additional user data to firebase
await db.push().set({
  'uid': userId,
  'email': email,
  'name': name,
  'cellNo': cell,
  'city': city,
  'address': address,
  'createdAt': FieldValue.serverTimestamp(),
});
```

Login - Flutter backend

- Open user_auth.dart
- Create a static method login that return bool in future and accept details
- Following method authenticate user and return UserCredential
- `FirebaseAuth.instance.signInWithEmailAndPassword (email: 'email', password: 'pass');`
- Call this method and wait for to create user
- Wrap in try catch and return appropriate flag
- Get Auth Token
- `await userCredential.user?.getIdToken();`

Modify OnPress Event in UI

- Goto login ui and call this method on button click
  - Get data from controller and pass it to signup method
  - Check for error and show message or take to login screen
  - Future<bool> f = UserAuth.login(email: email, _cemail.text: _cpass.text);

Multidex Error – File Size Exceed

- If you see “D8: Cannot fit requested classes in a single dex file” error then enable multi dex
  - Open app/build.gradle
  - under defaultConfig add
    - multiDexEnabled true
  - Under dependancies add
  - implementation ‘com.android.support:multidex:1.0.3’

Next Class

- Working with Firebase
  - Saving Auth Tokens
  - Autologins

Reading

- Lecture via White board / Slides
- https://firebase.google.com/docs/flutter/setup