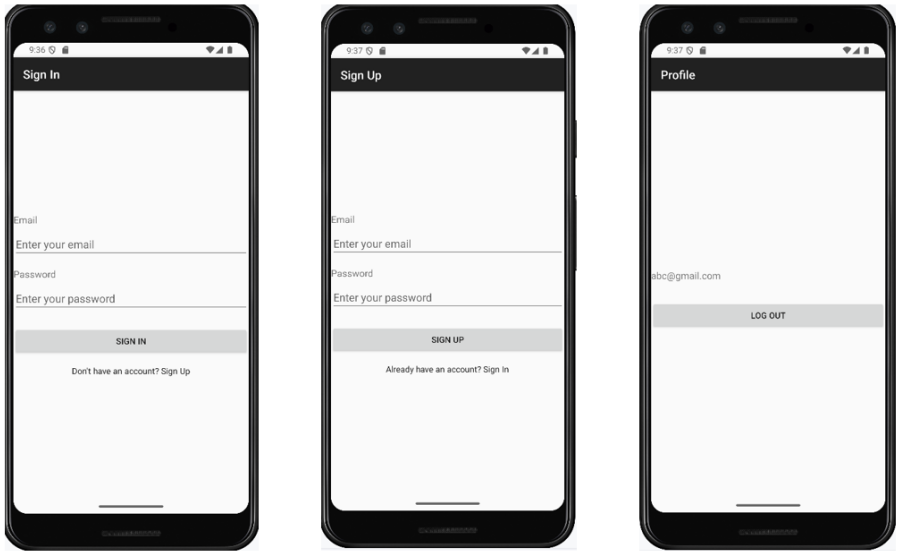
CSE 2200: Advanced Programming
Lab Tutorial: Firebase Authentication

In this lab tutorial, we will implement **login** and **registration** system in Android applications using **Firebase Authentication.**



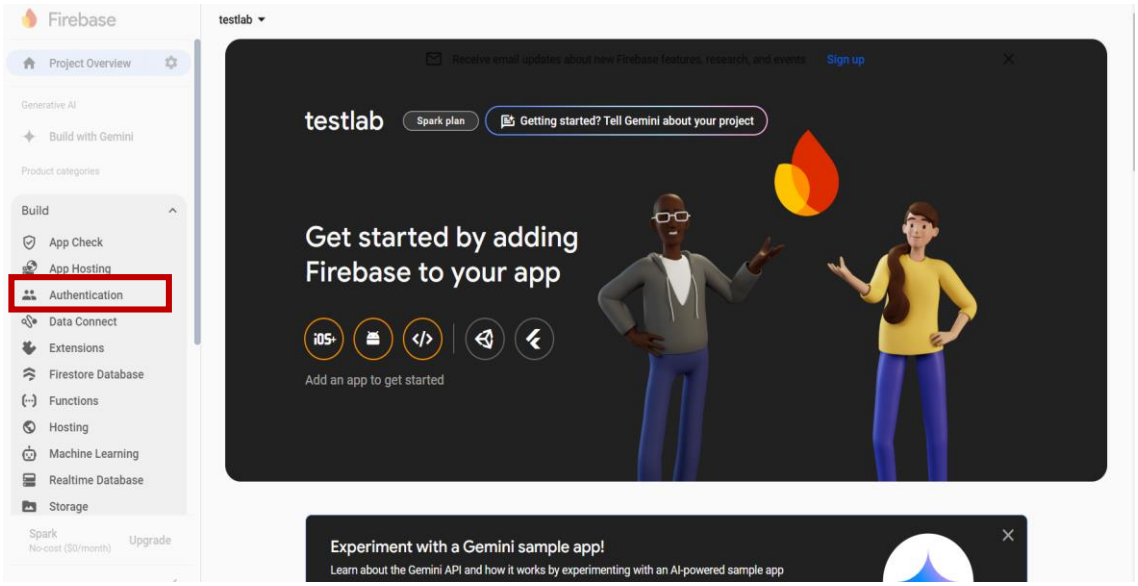
Firebase is a popular choice for Android developers because it offers a suite of cloud-based services that make app development easier, faster, and more scalable. Key features include:

1. **Real-time Database** - Sync data across users instantly.
2. **Authentication** - Simplify user sign-in (email, social, etc.).
3. **Cloud Storage** - Store user-generated content securely.
4. **Cloud Firestore** - Scalable database for complex queries.
5. **Push Notifications** - Engage users with Firebase Cloud Messaging.
6. **Crashlytics** - Real-time crash reporting.
7. **Analytics** - Track user behavior and app usage.
8. **In-App Messaging** - Send messages to active users.
9. **Remote Config** - Update app features without releases.
10. **Performance Monitoring** - Track app performance issues.
11. **ML Kit** - Add machine learning features easily.

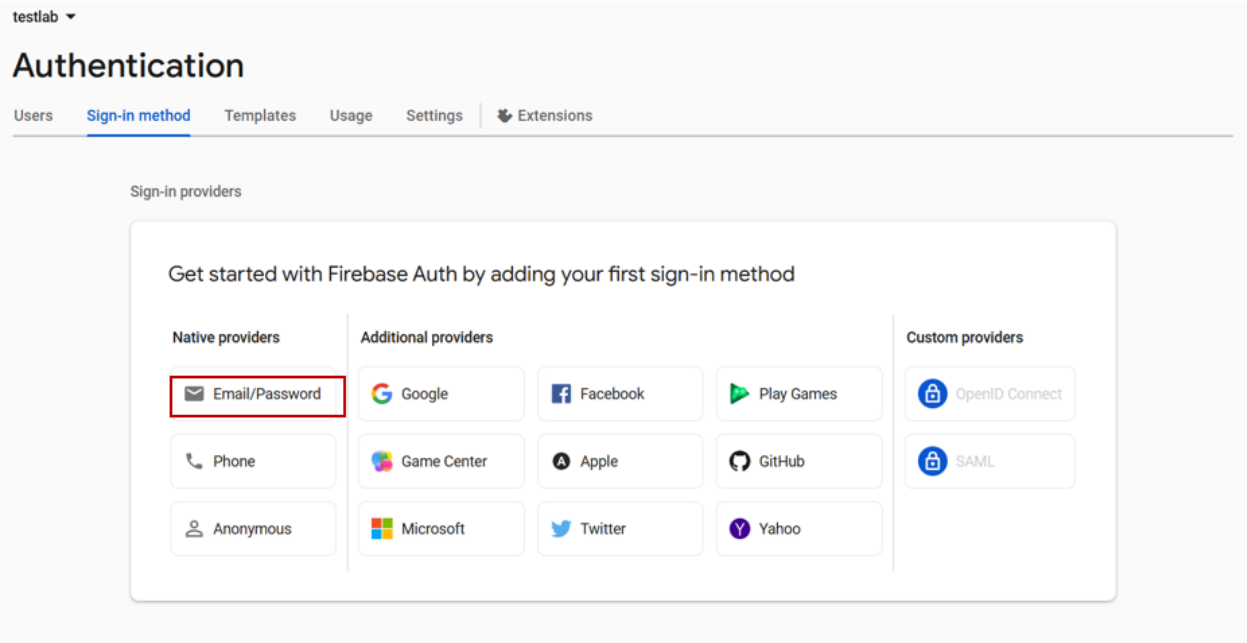
Firebase helps build secure, scalable apps quickly with less backend work. We will use Firebase Authentication in this task.

Step-1: go to https://console.firebase.google.com/ > Sign in through your google ID > Click "Getting Started" > "Create a Project with name and id" > You can turn off Analytics option for now. Then you will have the below interface in your console.

CSE 2200: Advanced Programming
Lab Tutorial: Firebase Authentication

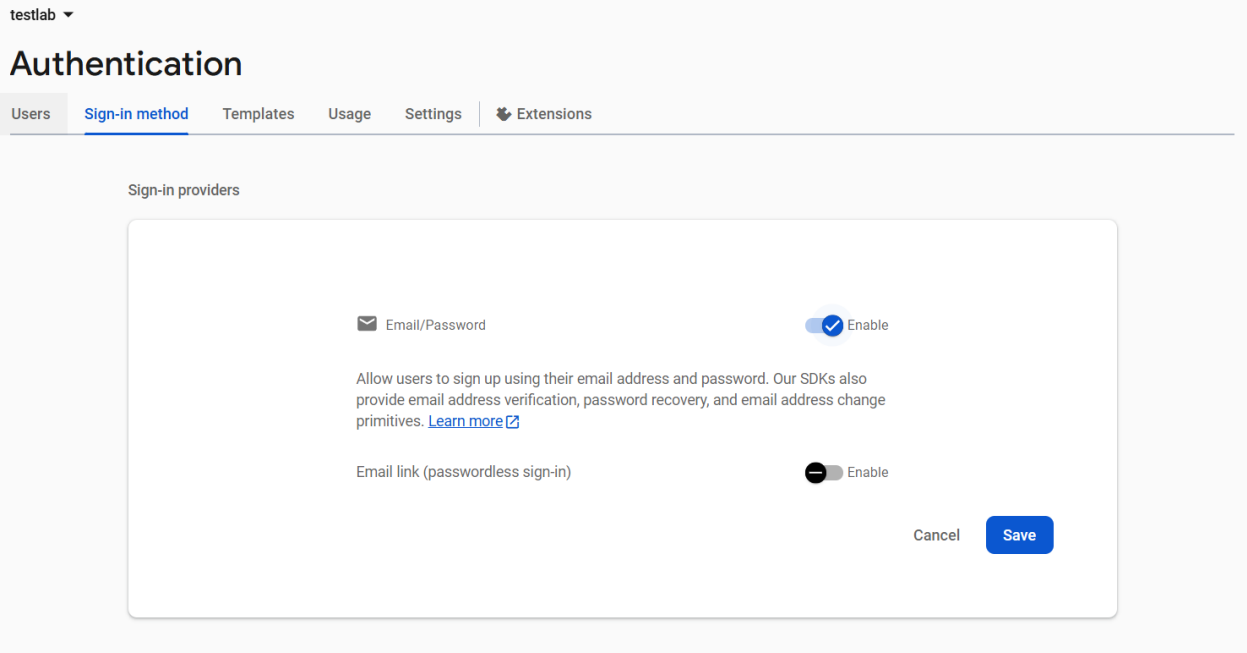


Then go to **"Build"> Choose "Authentication" > "Get Started"**



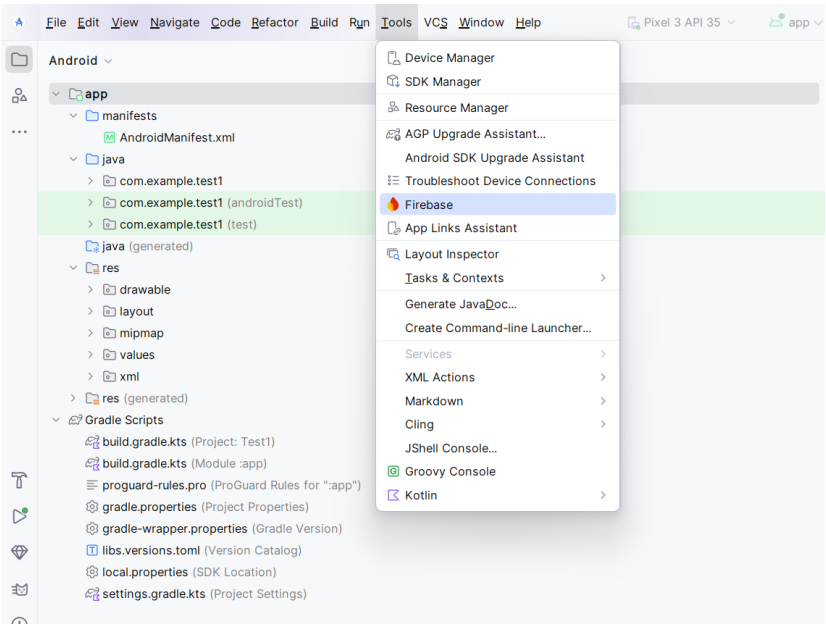
Now we will implement Authentication using **"Email/Password"**.

CSE 2200: Advanced Programming
Lab Tutorial: Firebase Authentication



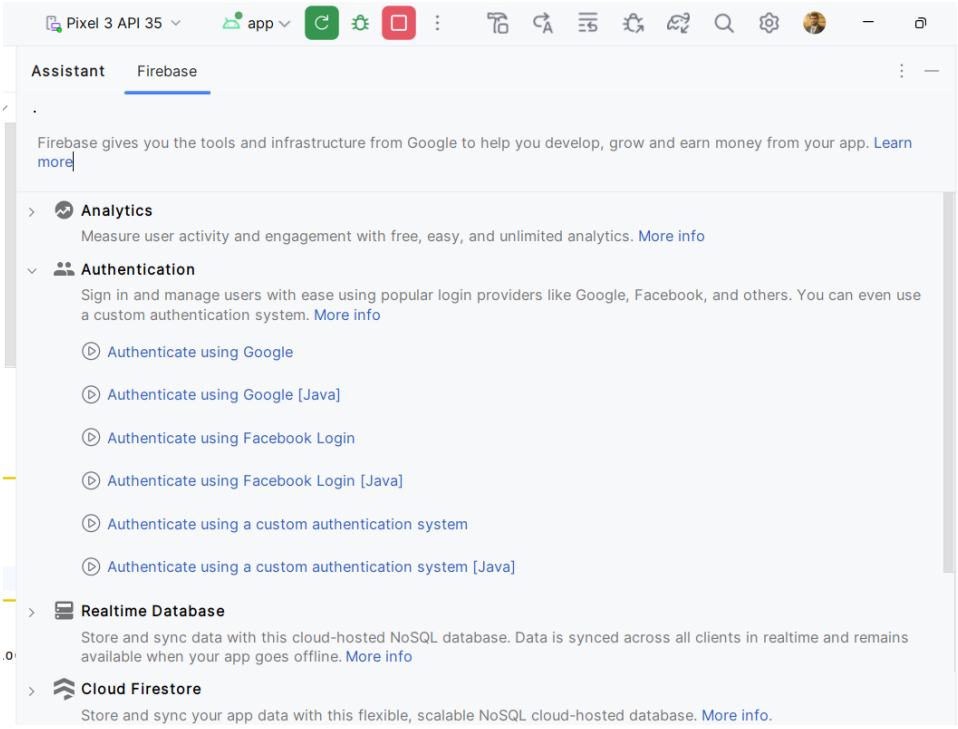
Once you save this, Step-1 is complete.

Step-2: Now you need to go to Android Studio, Create an empty project. Go to **"Tools".** Choose Firebase.



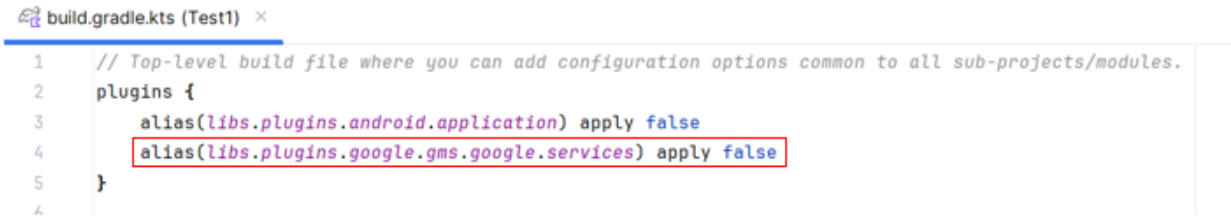
CSE 2200: Advanced Programming
Lab Tutorial: Firebase Authentication

Now with the assistant window, go to Authentication and Enable **"Authenticate using a custom Authentication System".**



Step-3:

- Click "Connect your app to Firebase", a pop up window will be opened in your browser. Choose the app in your firebase control that you have created in Step-1.
- Click "Add the Firebase Authentication SDK to your app".
- Once both of the process are marked done, you can check in the Gradle files to see if the dependencies are added properly. If not, you have to do it manually.



CSE 2200: Advanced Programming
Lab Tutorial: Firebase Authentication

Step-4: Now you have **'MainActivity'** and **'layout_main.xml'** in your android project. Create two more activities for **Sign Up** and **Profile Page**. You can **design** your login, signup and profile page as per your requirements. For now, you can **use** the **xml files provided** and make necessary changes in your app. (***activity_main.xml, activity_sign_up.xml, activity_logout.xml***)

Step-5: Add the action bar to your app. Go to **'AndroidManifest.xml'** and replace the theme portion to this line:

```
android:theme="@style/Theme.AppCompat.Light.DarkActionBar"
```

go to each activity classes and inside **'onCreate'** function add the following line:

```
this.setTitle("Sign In"); //inside main activity class
```

Add similarly to other activity classes too.

Step-6: (Inside SignUpActivity.java)

Now initialize objects for your **EditText, TextView** and **Button** widgets that you have created in your layout and connect their IDs. In addition to these objects you also need to initialize the **ProgressBar** and **FirebaseAuth** objects.

```
ProgressBar signUpProgressBar;
FirebaseAuth mAuth;
```

And inside **'onCreate'** function:

```
mAuth = FirebaseAuth.getInstance();
signUpProgressBar = findViewById(R.id.sprogressBar);
```

The line `mAuth = FirebaseAuth.getInstance();` is used in Android development with Firebase to initialize an instance of the FirebaseAuth class, which handles authentication for the app. You will now be able to use various functions with your **mAuth** object.

Step-7: (Inside SignUpActivity.java)

Implement Button click listener method for your **"Sign Up" button** and TextView click listener method for your **'Sign In' redirect text**. Inside the Sign in Redirect click method, you can change the intent to your sign in activity.

```
signInRedirect.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(getApplicationContext(),
MainActivity.class);
        startActivity(intent);
        finish();
    }
});
```

Now inside the **onClick** method of the **Sign Up** button:

Set the **visibility** of the progress bar to **visible**.

```
signUpProgressBar.setVisibility(View.VISIBLE);
```

Add various **constraints** to check the user input is according to your **requirements**.

```
String email, password;
email = String.valueOf(signUpEmailEditText.getText());
password = String.valueOf(signUpPasswordEditText.getText());
if(TextUtils.isEmpty(email)){
    Toast.makeText(SignUpActivity.this, "Enter Email",
Toast.LENGTH_SHORT).show();
    return;
}
if(TextUtils.isEmpty(password)){
    Toast.makeText(SignUpActivity.this, "Enter Password",
Toast.LENGTH_SHORT).show();
    return;
}
```

Then you need to do the **Firebase authentication** task for Sign up. go to https://firebase.google.com/docs/auth/android/password-auth and copy the following code to paste inside the **onClick** method.

4. Create a new account by passing the new user's email address and password to `createUserWithEmailAndPassword`:

Kotlin+KTX Android | Java Android

```java
mAuth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    // Sign in success, update UI with the signed-in user's information


                } else {
                    // If sign in fails, display a message to the user.


                }
            }
        });
```

EmailPasswordActivity.java

Now if the "sign up" is **successful**, you can add any **Toast Message** and also **change the intent** to your **sign in activity class**. Also, you can show **Toast Message** when the "sign up" attempt **fails**.

Important: Add the following function before **'onCreate'** function:

```java
@Override
public void onStart() {
    super.onStart();
    FirebaseUser currentUser = mAuth.getCurrentUser();
    if(currentUser != null){
        Intent intent = new Intent(getApplicationContext(), logout.class);
        startActivity(intent);
        finish();
    }
}
```

This code is part of an Android `Activity` that checks if a user is already logged in when the activity starts.

Explanation

- **`onStart()`**: This method runs every time the activity becomes visible to the user.
- **`mAuth.getCurrentUser()`**: Checks if a user is currently logged in (returns `null` if no user is logged in).

- **If `currentUser` is not null**: It means the user is logged in. The code then:
  - Creates an `Intent` to navigate to the `logout` activity.
  - Starts the `logout` activity.
  - Calls `finish()` to close the current activity.

This code effectively redirects logged-in users to the `logout` activity.

Step-8: (Inside MainActivity.java)

Same as Step-6. Make necessary changes for sign in instead of sign up.

Step-9: (Inside MainActivity.java)

Same as Step-7. Make necessary changes for sign in instead of sign up.

Then you need to do the Firebase authentication task for log in. go to https://firebase.google.com/docs/auth/android/password-auth and copy the following code to paste inside the **onClick** method.

3. When a user signs in to your app, pass the user's email address and password to `signInWithEmailAndPassword`:

Kotlin+KTX Android | Java Android

```
mAuth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    // Sign in success, update UI with the signed-in user's information


                } else {
                    // If sign in fails, display a message to the user.


                }
            }
        });
```

EmailPasswordActivity.java

CSE 2200: Advanced Programming
Lab Tutorial: Firebase Authentication

Important: Add the following function before **'onCreate'** function:

```
@Override
public void onStart() {
    super.onStart();
    FirebaseUser currentUser = mAuth.getCurrentUser();
    if(currentUser != null){
        Intent intent = new Intent(getApplicationContext(), logout.class);
        startActivity(intent);
        finish();
    }
}
```

Step-10: (Inside logout.java)

Initialize objects and connect IDs for your Button (logout), TextView (to show email). Also initialize FirebaseAuth and FirebaseUser object too.

```
FirebaseAuth mAuth;
FirebaseUser user;
```

And inside **'onCreate'** function:

```
mAuth = FirebaseAuth.getInstance();
user = mAuth.getCurrentUser();
// user stores the information associated with the user currently logged in.
```

So you can check if the user is **NULL** or not. If null, you can switch the activity to sign in page. Otherwise you can **display** the **email** of the user.

```
if (user == null){
    Intent intent = new Intent(getApplicationContext(), MainActivity.class);
    startActivity(intent);
    finish();
}
else {
    showEmail.setText(user.getEmail());
}
```

Implement 'Log Out' button click listener and inside onClick method: sign out the mAuth instance and change the **intent** to sign in page.

```
mAuth.signOut();
Intent intent = new Intent(getApplicationContext(), MainActivity.class);
startActivity(intent);
finish();
```

The **login** and **registration** system in Android applications using **Firebase Authentication** is now complete. Run the app and check its functionality. Thank you. 😊