**­­CMPS 312 Mobile App Development**

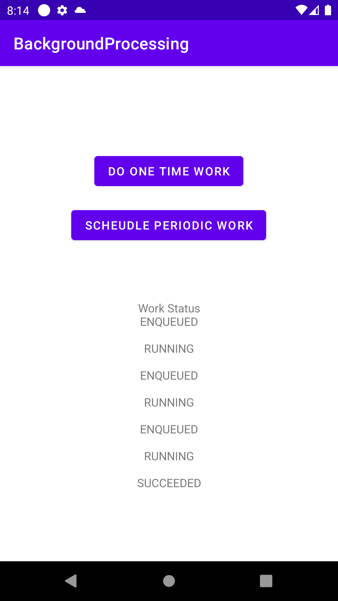
**Lab 12 – Cloud** **Firestore Authentication, Storage &**

**Background Work with WorkManager**

**Objective**

In this Lab you will practice how to:

* Use Firebase Authentication and security rules to secure Firestore data and application
* Create a Cloud Storage bucket to upload and download files
* Create Background Work with WorkManager and support both asynchronous one-off and periodic tasks



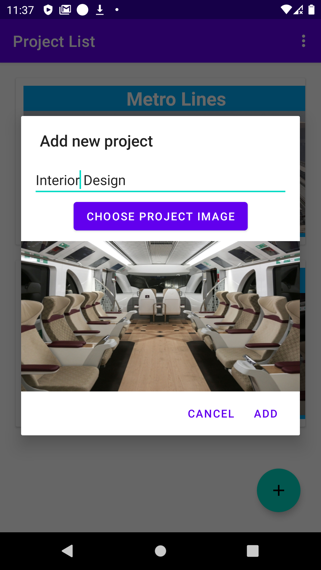
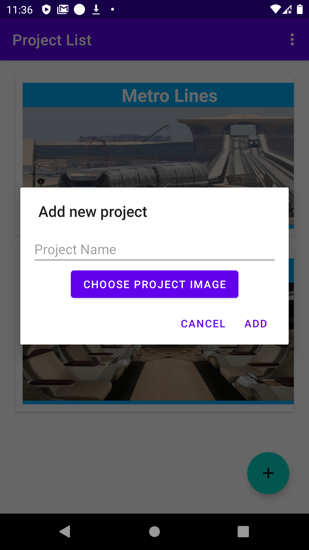
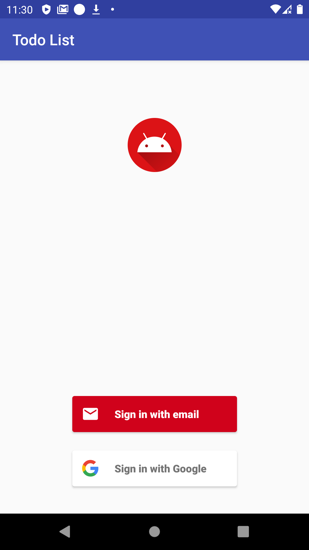
**Preparation**

1. Sync the Lab GitHub repo and copy the **Lab 12-****Cloud Firestore and Background Processing** folder into your repository.
2. Import the **TodoList** project into Android Studio. You will probably see some compilation errors or maybe warning messages. We'll correct this in the next sections.



**PART B: Cloud Firebase Storage**

In this section, we add the images for the project and upload them to firebase storage



1. Add the following dependencies to use Firebase Storage and to download and display images inside your adapter

*// Firebase storage*implementation 'com.google.firebase:firebase-storage-ktx:19.2.0'

*// Glide image download library*implementation 'com.github.bumptech.glide:glide:4.11.0'  
kapt 'com.github.bumptech.glide:compiler:4.11.0'

1. Create a function called **uploadPhoto** inside the TodoListRepo that takes Photo URI and upload it to the firebase storage. Then the function should return the uploaded photo URI

Graphical user interface, text, application

Description automatically generated

1. Modify the **addProject** function inside the ViewModel to call this function and pass the photo URIGraphical user interface, text

   Description automatically generated
2. Create a function inside the **ProjectListFragment** called **openGallery** that allows you to select an image from the gallery.
3. Call this function when the dialog box's **Choose Project Image** button is clicked.
4. Add the **onActivityResult** function and get the image URI. Also, show this image in the dialog box
5. A screenshot of a cell phone

   Description automatically generatedWhen the user presses on add button, then call the **addProject** function inside the project view model and pass the **newProject** object and the **ImageUri**.

projectViewModel.addProject(newProject, photoUri)

1. Test your application

A screenshot of a video game

Description automatically generated

**PART C: Background Processing using Work Manager**

In this section, you will create a simple application that schedules work in the background. You will also learn how to get the status of running work.

1. Create a new project and call it Background Processing
2. Add the following dependency

*// Kotlin + coroutines*implementation "androidx.work:work-runtime-ktx:2.4.0"

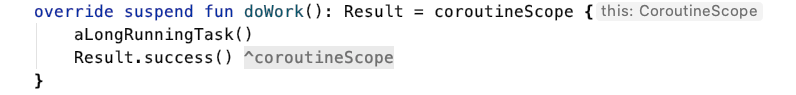
1. Create the following layout

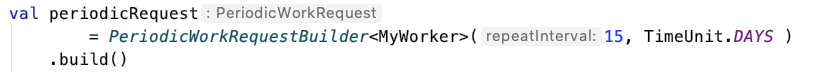
Graphical user interface, text, application, Teams

Description automatically generated

1. Create a worker class and name it **MyWorker** that implements **CoroutineWorker**
2. Override the doWork() function
3. Create another method called aLongRunningTaskA picture containing text

   Description automatically generated
4. Call this method insdie the doWork function

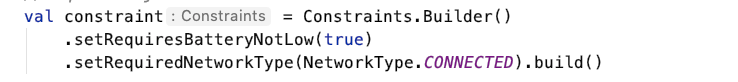
****

1. Create a one time request object inside the MainActivity and enque it when the One Time Button is clicked
2. Create a periodic request and enqueue it when the schedule periodic button is clicked
3. Add the following listener to show the status of the work

Graphical user interface, text, application

Description automatically generated

1. Test your code
2. Add the following constraints and test your code again by modifying the emulator properties



Graphical user interface, text, application

Description automatically generated

A picture containing diagram

Description automatically generated

**A picture containing text

Description automatically generated**