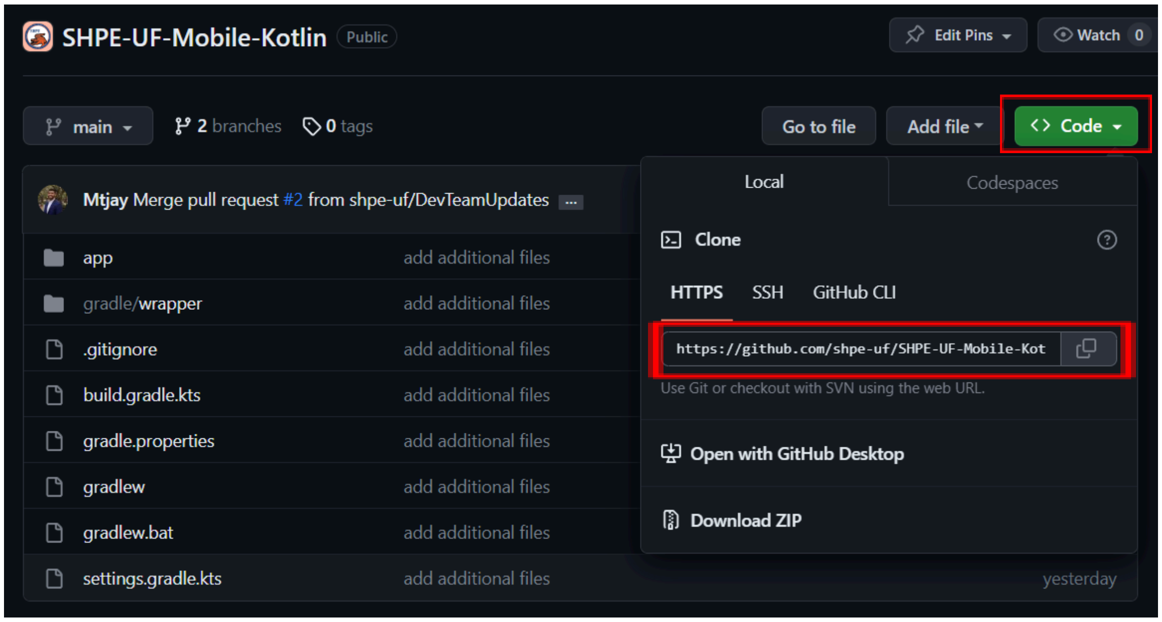
SHPE-UF-Mobile-Kotlin Onboarding

Overview

This document outlines the information required to get up-to-speed with the SHPE UF Android App including proper use of Git/GitHub, Android Studio, and project structure.

Running the app on your local device

1) If you haven't already, set up your Android Studio environment
   a) Follow the Android Studio Setup guide
2) Once you have access to the GitHub repo, copy the https link under "<> Code".



Mtjay Merge pull request #2 from shpe-uf/DevTeamUpdates

app	add additional files
gradle/wrapper	add additional files
.gitignore	add additional files
build.gradle.kts	add additional files
gradle.properties	add additional files
gradlew	add additional files
gradlew.bat	add additional files
settings.gradle.kts	add additional files

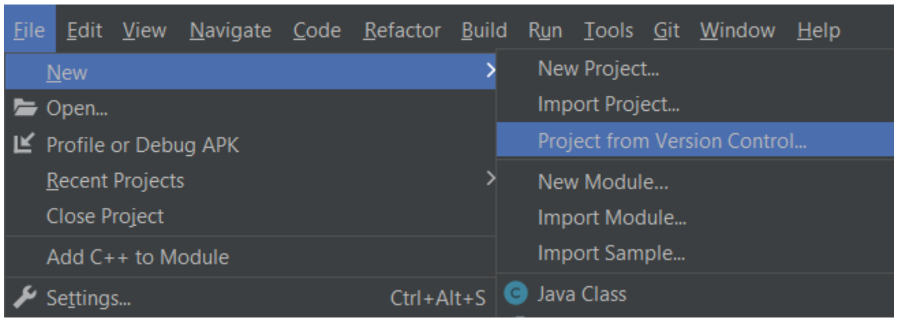
```
https://github.com/shpe-uf/SHPE-UF-Mobile-Kot
```

Use Git or checkout with SVN using the web URL.

3) Go to your desired directory in your local device and run the **git clone** command with the copied link in cmd:

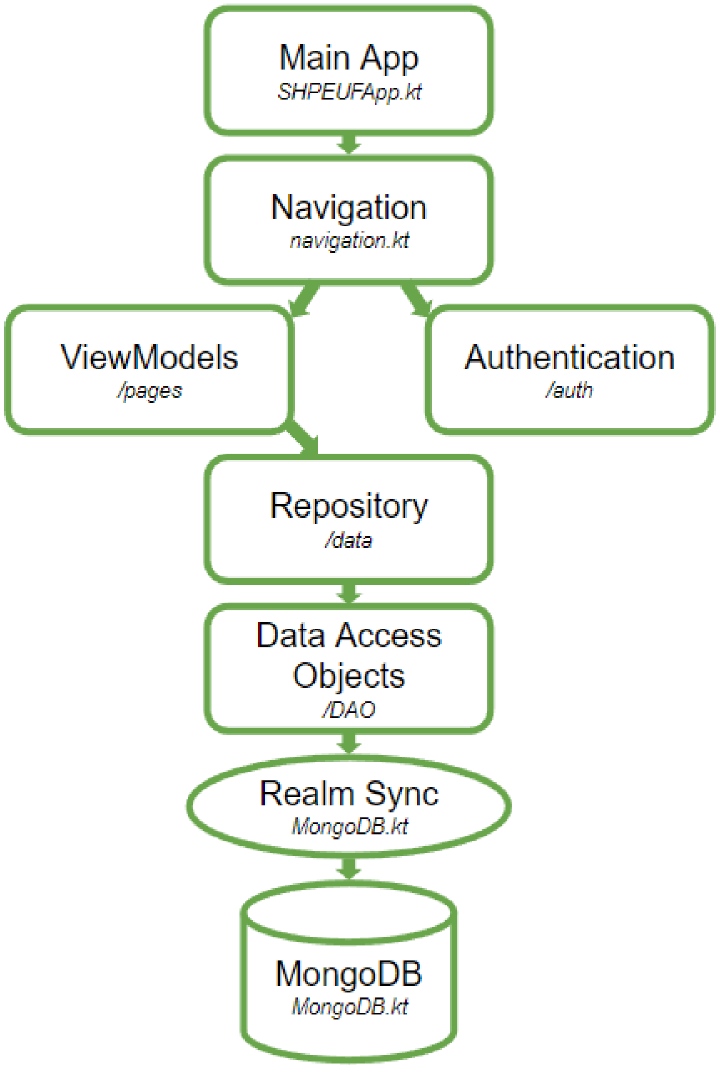
```
git clone https://github.com/shpe-uf/SHPE-UF-Mobile-Kotlin.git
```

4) Open SHPE-UF-Mobile-Kotlin in Android Studio.
   a) Alternatively, you can open the project directly from Android Studio by going to File->New->Project from Version Control and adding the link (you must have your GitHub account linked to Android Studio)



5) Run the application in your emulator by clicking on the ▶ button located on the top bar of your Android Studio window. You should be able to see the app on your emulator.

How it works



Data Layer

The entry point of the data layer is *repositories*.The data layer is made of *repositories* that each can contain zero to many *data sources*. A single repository class should represent each different type of data handled in an app. For example, a `MoviesRepository` class for data related to movies, or a `PaymentsRepository` class for data related to payments.

`Repository classes` are mainly responsible for exposing data to the rest of the app, among other things. State Holder classes such as ViewModels (found in the UI Layer) should use repository classes as the only entry point to the data layer.

What's a DAO (Data Access Object)?

> A Database Access Object (DAO) is a design pattern used to abstract and encapsulate the interaction with a database.
> **Translation**: it's an object that will handle communication with the database.

The repository class will expose the data to the UI layer via the ViewModel. The repository class will call the DAO, and the DAO will then perform the call to the database.

To learn more about the Data layer, visit this codelab [(Building a Data Layer)]

UI Layer

The UI layer is responsible for reflecting and handling the UI state. UI state is the data that the user should see at a given moment. For example, if a user clicks a `Read Movie Description` button on a Movie App, the user should see the description for that movie. This is UI State.

The UI layer is broken down into UI elements and State holders. UI elements are activities that display the data (buttons, lists, cards, text boxes). State holders are responsible for the production of UI state and contain the necessary logic for that task.

To learn more about the UI layer, visit this codelab [(ViewModel and State in Compose)]

Here is a quick walkthrough of the main folders in our UI:

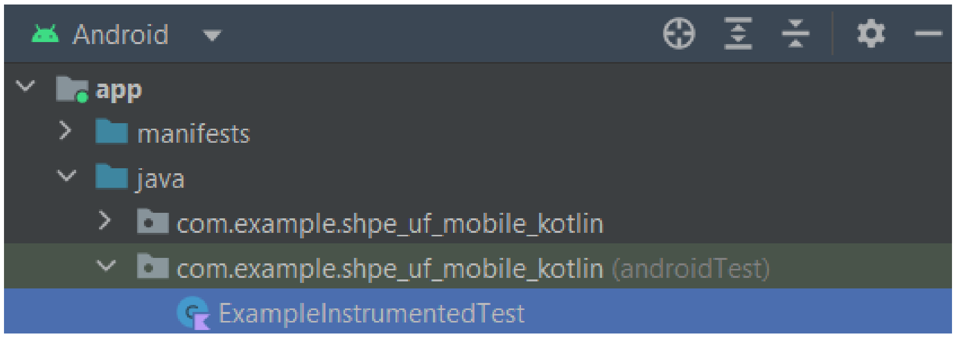
- /navigation
    - Contains the navigation controller, used to switch between screens in the app

- /pages
    - Contains all the screens in the app as well as their respective ViewModels (Sign In, Register, Home, Points)
- /sharedComponents
    - Contains common UI elements such as buttons, cards, forms, etc.
- /theme
    - Contains color, shape and type files for application theming
- /java/res
    - This folder contains all the images, strings, and other static assets

Testing

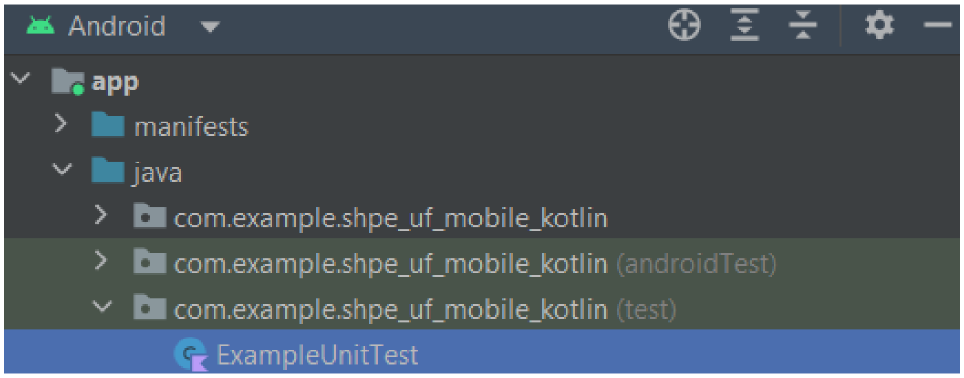
Testing files are found in the following folders:

Instrumented tests (UI) - app -> java -> com.example.shpe_uf_mobile_kotlin (androidTest)



```
Android
app
  manifests
  java
    com.example.shpe_uf_mobile_kotlin
    com.example.shpe_uf_mobile_kotlin (androidTest)
      ExampleInstrumentedTest
```

Unit test - app -> java -> com.example.shpe_uf_mobile_kotlin (test)



```
Android
app
  manifests
  java
    com.example.shpe_uf_mobile_kotlin
    com.example.shpe_uf_mobile_kotlin (androidTest)
    com.example.shpe_uf_mobile_kotlin (test)
      ExampleUnitTest
```

Making changes to the frontend

If you add any new files, make sure everything stays organized and follows the proper file structure. Try to create **reusable** components and only create new components when needed. Additionally, if you are styling a component, do not hardcode colors or shapes. Use the Color.kt, Theme.kt, and Type.kt. This will allow us to maintain a consistent theme throughout the app.

Working with GitHub

High level walkthrough:

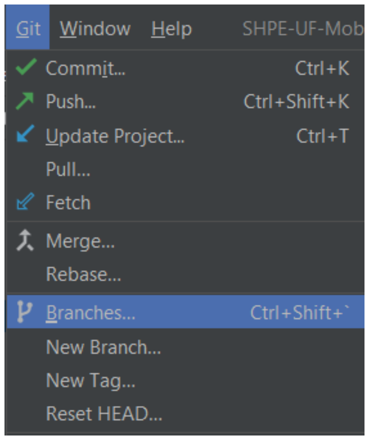
1) Once you are satisfied with your code, you will push your code and open a pull request. In your pull request, make sure to add a brief description of your code/implementation and link it to your task (more about this below).
2) Once your pull request passes all the tests and is approved by a technical officer, it is ready to be merged with the master branch.

Making branches, commits with Android Studio

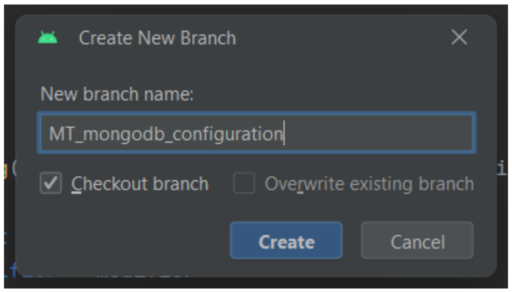
Branches

When you are working on adding a new feature or fixing a bug in your source code, group all related changes in a separate branch. This way, you can manage everything related to that particular task in one place. For example, if you decide not to add a new feature after building it without using source control, it can be a significant effort to undo the work. If you organize your work in a branch, you can decide not to merge the branch and avoid having to undo work.

In Android Studio, you can look at all the local branches(those on your device) and remote branches (those on the online repository) on the Git tab.

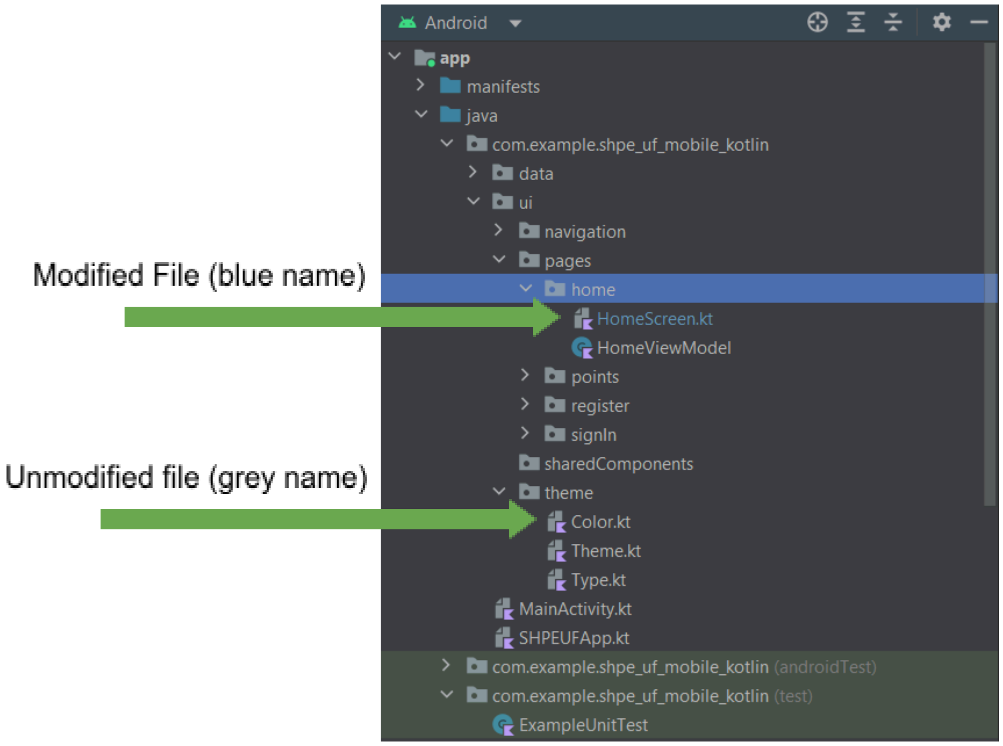


Additionally, you can create a branch by clicking “New Branch…” below “Branches…”. Once you are ready to create your new branch, make sure it follows the naming standard of *INITIALS_brief_description,* and make sure “Checkout branch” is checked if you want to work on that branch right after creating it.

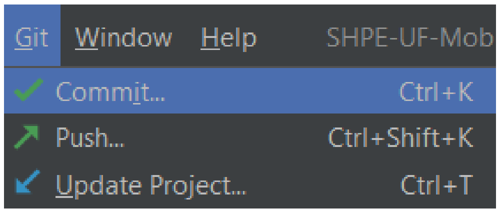


Commits
Commiting is like saving a snapshot of your project's current state, allowing you to document changes, save a version of the code, and revert back to it when needed.

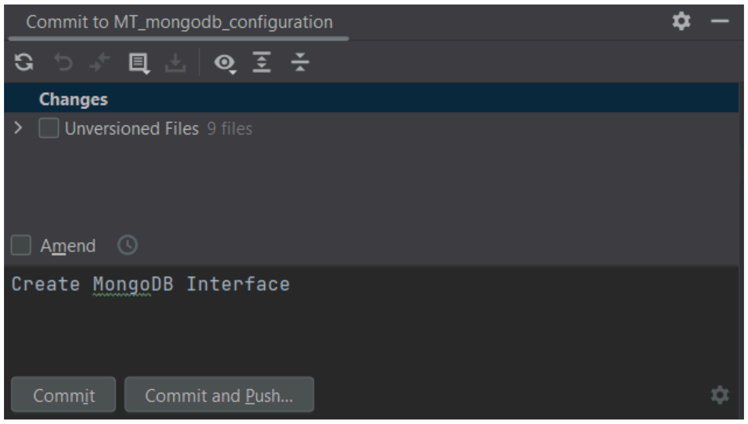
As you make changes to your source code in a source control repository, Android Studio tracks those changes and highlights them in the Project navigator on the left side panel. While you're working on a change, look at the Project navigator to see your changes in the current branch.



When you're done making changes, commit them to save them permanently in your source control repository. Choose Git > Commit, and review your changes.

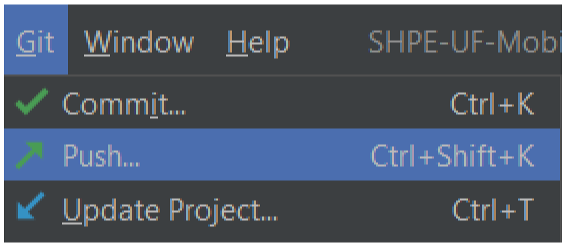


Document your changes with a commit message that describes what your commit accomplishes. Click the “Commit” button to commit your changes, or click the “-” to continue working without committing to the repository.



```
Create MongoDB Interface
```

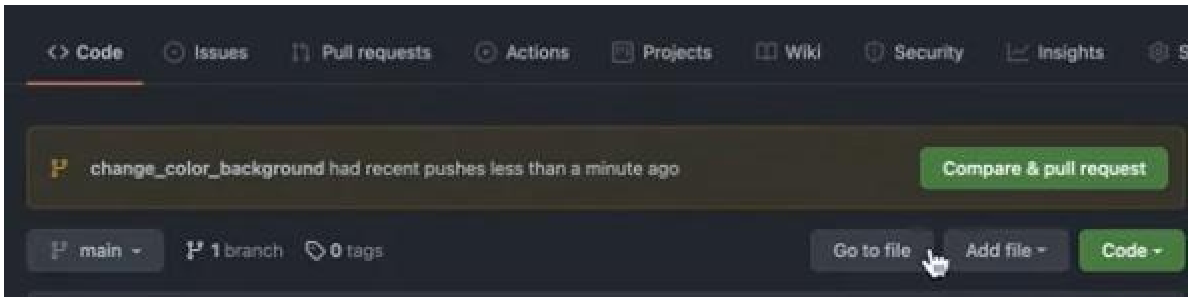
If you want to push your code to GitHub as well, click the “Commit and Push” button. Alternatively, you can click “Push” on the Git tab (Git > Push >Push)



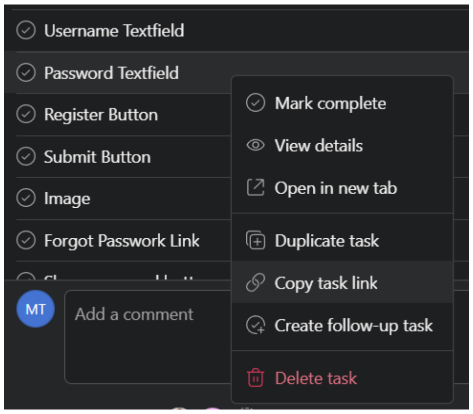
Sending a pull request

- Make sure your branch has the naming standard of *INITIALS_brief_description.* That is, a brief description of what you are working on.
- Make sure your code passes all the tests before sending the pull request. As our application grows, we will be creating new tests to ensure the reliability of the app.
- Make sure to follow the template the file *pull_request_template.md* here

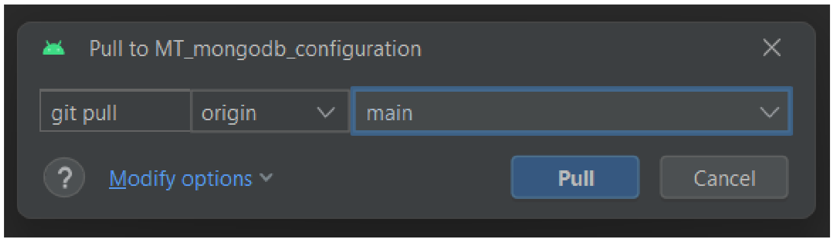
After committing and pushing your changes, go to Github to open a pull request, it will be labeled as “Compare & pull request”.



Once you create your pull request, add a brief description and link it to your Asana task. In Asana, right-click that task you were assigned and click “copy task link”. Paste in the pull request description.



To get new changes from the remote repository you can click source Git > Pull. In the dialog that appears, select the branch with the changes you want to apply to your local repository, and click Pull.



For using Git/GitHub without Android Studio, watch this video (Git and GitHub for Beginners - Crash Course)

What’s Next?

You are finally ready to contribute to the SHPE UF Android App. Lastly, make sure you are in the SHPE UF Tech Cabinet Discord as well as the SHPE-UF Mobile (Kotlin) Asana project.

If you haven’t, go over the Development Process doc to learn how to assign and complete tasks using Asana, and what to expect from meetings.