**Steps for creating Micro-App Template**

1. **Windows**

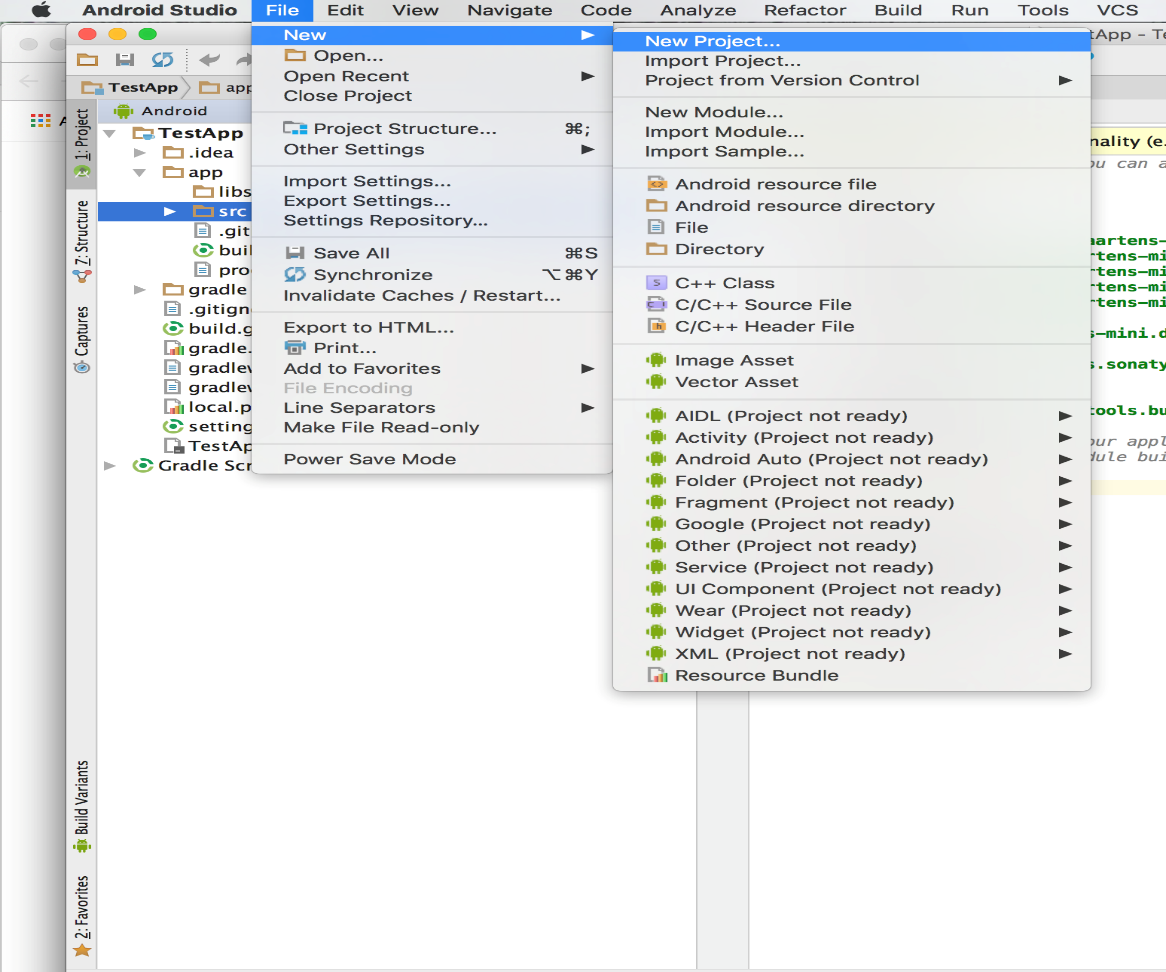
* Navigate to Android studio installed path
* Navigate to templates folder through plugins 🡪 android 🡪 lib 🡪 templates
* Tap on activities folder and copy PhilipsMicroApp folder
* Tap on gradle-projects 🡪 NewAndroidProject 🡪 root 🡪 build.gradle
* Open build.gradle and copy artifactory repository urls, find the build.gradle.ftl shared for your reference.
* Restart android studio and select PhilipsMicroApp either as new project or new Module
* As App-infra and u-App are direct dependencies required to compile, to update versions of it, need to update build.gradle explicitly.

1. **Mac**

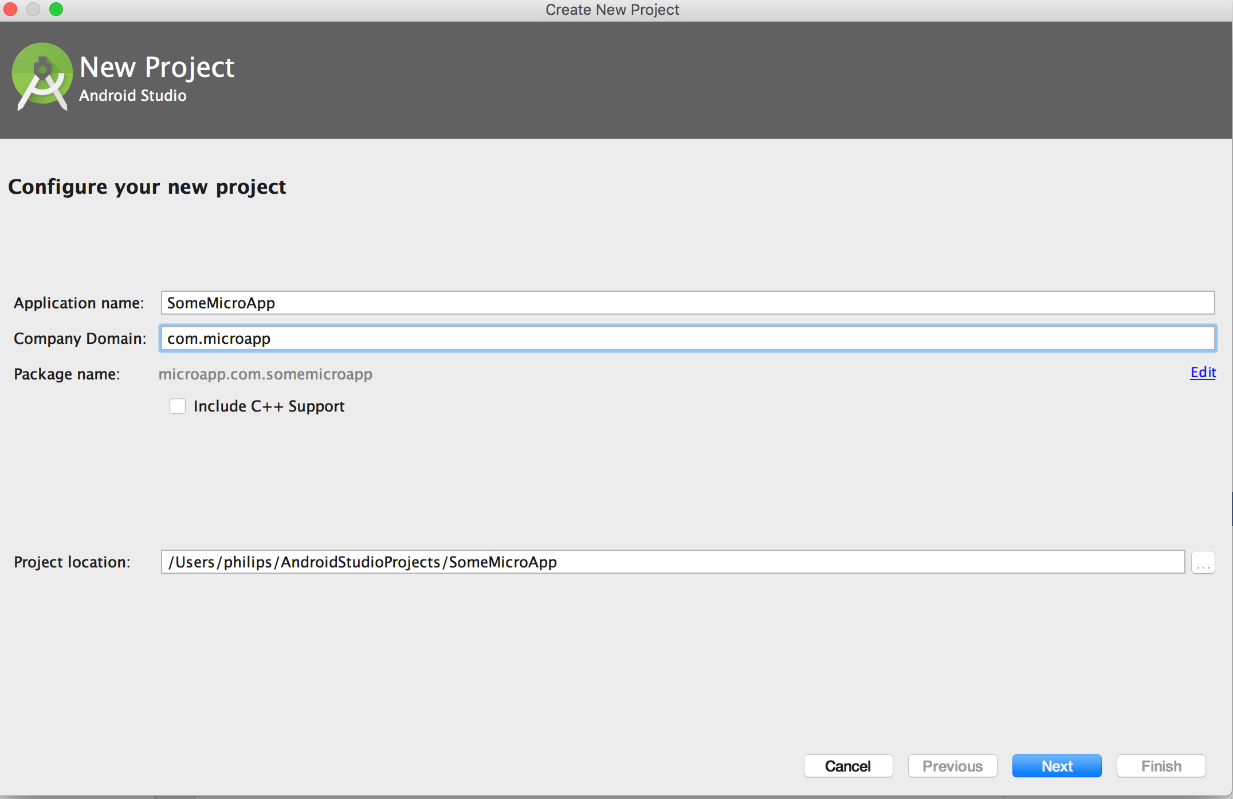
* Navigate to Applications from Finder
* Right click on Android Studio and tap show package contents
* Navigate to templates folder through Contents 🡪 plugins 🡪 android 🡪 lib 🡪 templates
* Tap on activities folder and copy PhilipsMicroApp folder
* Tap on gradle-projects 🡪 NewAndroidProject 🡪 root 🡪 build.gradle.ftl
* Open build.gradle.ftl and copy artifactory repository urls, find the build.gradle.ftl shared for your reference.
* Restart android studio and select PhilipsMicroApp either as new project or new Module
* As App-infra and u-App are direct dependencies required to compile, to update versions of it, need to update build.gradle explicitly.

**Steps in Selecting Philips Micro-app in Android Studio**

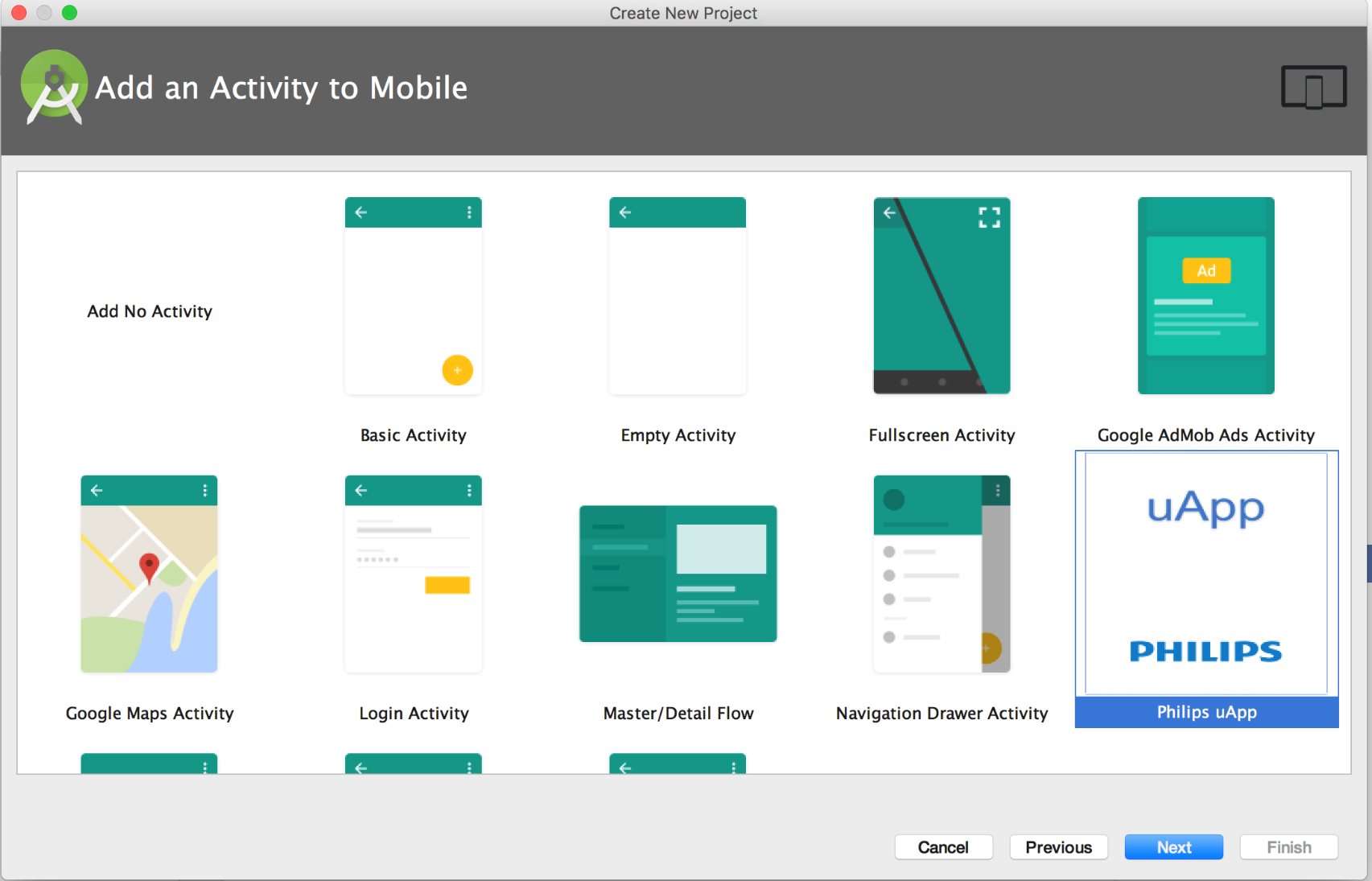
1. Create new project as show below



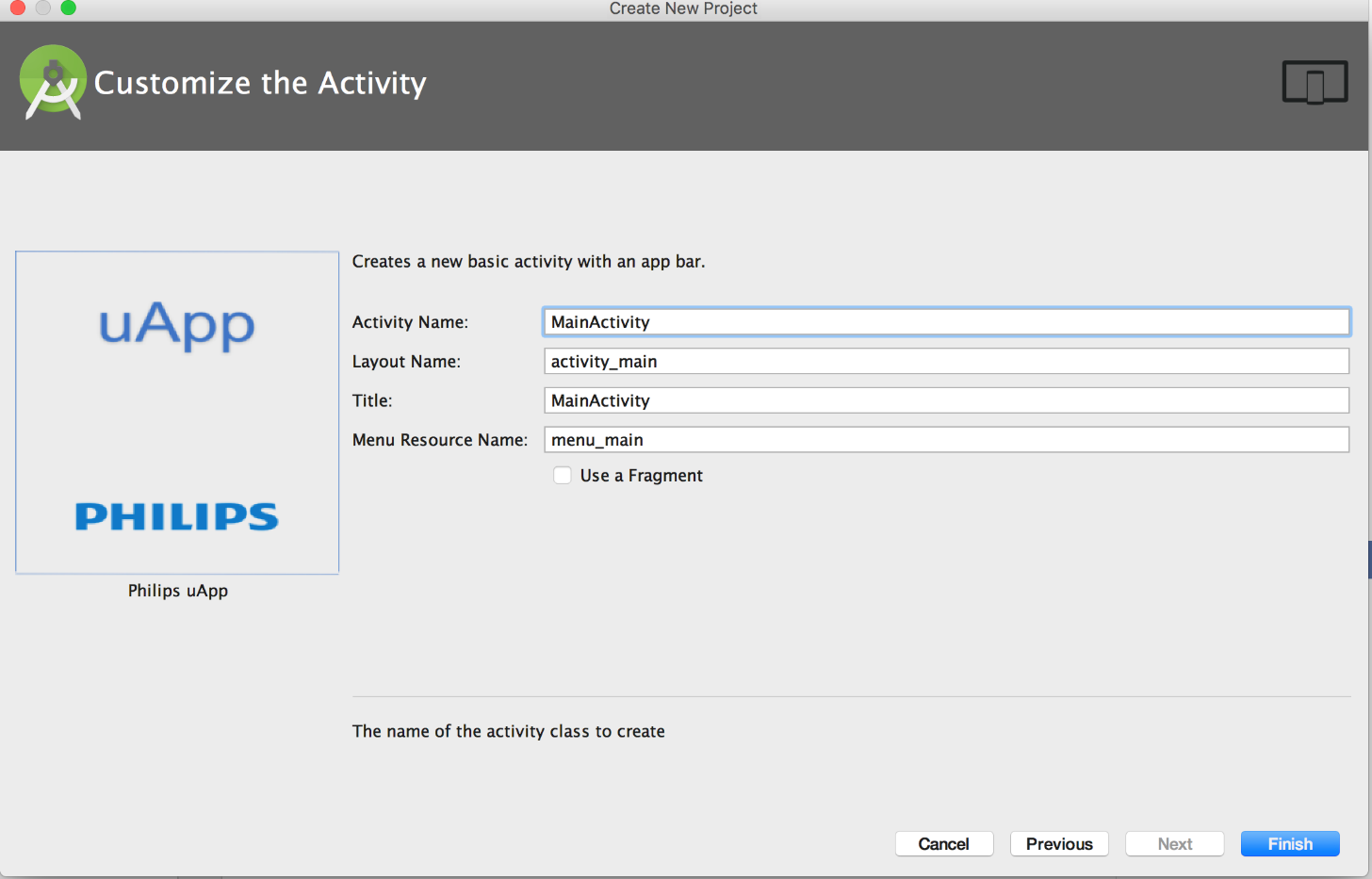
1. Provide Application name without space, as it concats app\_name to uApp dependencies java files as shown below



1. Select Philips uApp module as shown below



1. Provide Activity name as shown below and tapping finish will create Micro-app as per uApp defined standards.



**How to consume uApp Template:**

Once project is created via uApp Template, it will auto generate corresponding Dependencies, Interface, LaunchInput and Settings java classes.

Find the explanations for auto created classes below

**DemoAppDependencies**:

public class DemoAppDependencies extends UappDependencies {

public DemoAppDependencies(final AppInfraInterface appInfra) {

super(appInfra);

}

}

On the above code snippet, micro-app which was auto created requires App-infra as dependency hence injected accordingly

**DemoAppLaunchInput:**

public class DemoAppLaunchInput extends UappLaunchInput {

}

Kindly include the inputs which are required to launch your micro-app on LaunchInput class

**DemoAppSettings:**

public class DemoAppSettings extends UappSettings {

public DemoAppSettings(final Context applicationContext) {

super(applicationContext);

}

}

On the above code snippet, micro-app which was auto created requires Context, hence injected it under settings

**DemoAppInterface:**

Interface class can be used to initialize and launch your micro-app

public class DemoAppInterface implements UappInterface {

/\*\*

\* @param uappDependencies - App dependencies

\* @param uappSettings - App settings

\*/

@Override

public void init(final UappDependencies uappDependencies, final UappSettings uappSettings) {

}

Init method can be used to include initialization code when required

/\*\*

\* @param uiLauncher - Launcher to differentiate activity or fragment

\*/

@Override

public void launch (final UiLauncher uiLauncher, final UappLaunchInput uappLaunchInput) {

}

Launch method can be used to launch your micro-app based on the launcher type passed.

**Limitations**

1. Template app will not generated aar
2. Need to create component in Studio and add generated classes explicitly to generate aars