**Testing Approach for “BackSeatDrivers” Android application**

*Draft Prepared by* Sapana Belorkar *(live document, Please feel free to Edit/Update as needed)*

**Summary of the Product to be tested:**

The BackSeatDrivers application is intended to detect lane departure and signal the driver. That is, it determines when the vehicle is departing from its travel lane and warn the operator. It also recognizes the travel/driving lane.

**Team Members:**

* Belorkar, Sapana
* Burbidge, Adam
* Roseberry, Keith
* Varadaraju, Rakshith

**Resources Required for testing:**

Application “BackSeatDrivers” is an Android phone application. Minimal hardware resources required to test this application include team member’s personal computers for compiling, debugging and few emulators (will decide which ones soon) in testing scope.

Following Android smartphones will be used for testing on mobile devices:

* Samsung 5, Android 5.0
* Samsung 6, Android 7.0,
* HTC One M8, Android 6.0,
* Moto Z Play, Android 8.0.0

Please refer development plan on SSW-609 project GitHub site for tools used to develop the application.

**Defect tracking and fixes:**

The team will use GitHub/ZenHub for tracking testing issues. The developers will also function as the testing team. Issues fixes can be tracked by commit comments on GitHub.

Fixes are pushed to the developer’s branch until changes are tested and approved by pull request approver. Units tests should be passed locally before creating a pull request on GitHub.

**How to create unit tests in Android and run them from Android Studio:** *(usually not part of this document however since the Android application development is new for everyone involved in the project, I thought adding more details here will be useful)*

For testing Android apps, you typically create these types of automated unit tests:

* **Local tests:** Unit tests that run on the local machine only. These tests are compiled to run locally on the Java Virtual Machine (JVM) to minimize execution time. We use this approach to run unit tests that have no dependencies on the Android framework.

First of all make sure the dependencies section of the app module’s **build.gradle** file has

testImplementation **'junit:junit:4.12'**

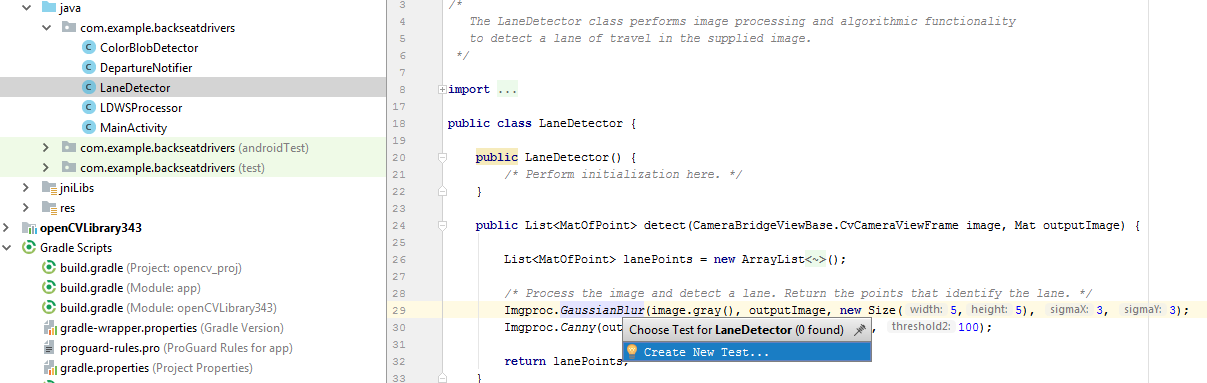


**Figure 1**

If you don’t have it, please go ahead and add it. (*I confirmed and it’s in our project already)*

To create local unit tests, you can create a new test for a specific class or method by following below steps:

* Open the Java file containing the code for which you want to create a unit test.
* Click the method or class name you want to test and press **Ctrl+Shift+T**
* You get the option to **Create new Test**, Click it.

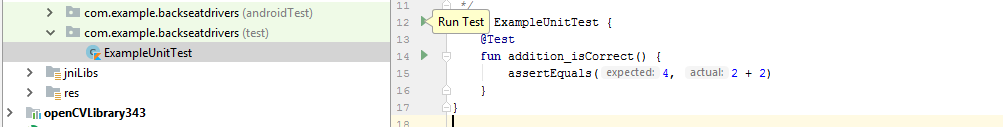


**Figure 2**

* In the **Create Test** pop up box, edit fields and select any methods to generate, and then click Ok.
* In the Choose **Destination Directory** pop up box, choose the below unit test folder and click OK.

*..app\src\****test****\java\com.example.backseatdrivers(test)*

* This will create a skeleton class for your test where now we can fill in the testing logic *(please refer SSW-567 notes to write testing logic)*
* Click **Run** Test to run your unit test



**Figure 3**

* **Instrumented tests:** Unit tests that run on an Android device or emulator. These tests have access to instrumentation information, such as the [Context](https://developer.android.com/reference/android/content/Context.html) for the app under test. Use this approach to run unit tests that have Android dependencies which cannot be easily filled by using mock objects.

First of all make sure the dependencies section of the app module’s **build.gradle** file has

androidTestImplementation 'com.android.support:support-annotations:28.0.0'  
androidTestImplementation 'com.android.support.test:runner:1.0.2'

androidTestImplementation 'com.android.support.test:rules:1.0.2'

*(Need to add androidTestImplementation 'com.android.support:support-annotations:28.0.0'*

*and androidTestImplementation 'com.android.support:test:rules:1.0.2' in our project)*

To create and run new Instrumented unit test, follow the steps given in Figure 2 and Figure 3 except source files for instrumented tests will be saved at

*..app\src\****test****\java\com.example.backseatdrivers(androidTest)*

Instrumented tests are essential when it’s required to evaluate your app's behavior against actual device hardware. Since we will be using actual devices to test this app Instrumented tests may not require at this time.

**Test Project structure:**

BackSeatDrivers Application

InstrumentationTestRunner

Test Package (test and AndroidTest)

Mock Objects

Junit Test class

Instrumentation Test Class

Team will verify functions with different images (saved in GitHub) to cover most scenarios on physical devices. There will be integration test performed on images and video after each sprint.