Use Case Analysis Tutorial

Following the steps:

1. Install the APK. It is best that you could have android phone, if not, please use the emulator that comes with android studio:

If you install the APK on Android phone, you can plugin your phone and install Vysor (https://chrome.google.com/webstore/detail/vysor/gidgenkbbabolejbgbpnhbimgjbffefm?hl=en-US), which is a chrome plugin app such that you can operate your phone from your computer for better efficiency.

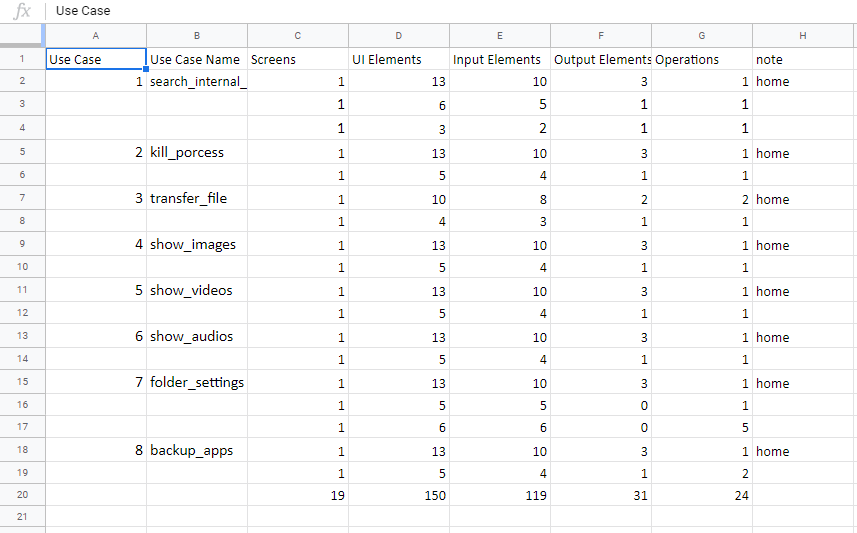
1. Run the Android app and identify use cases. Please see the definition of use cases here:

<https://en.wikipedia.org/wiki/Use_case>

“In software and systems engineering, a use case is a list of actions or event steps typically defining the interactions between a role (known in the Unified Modeling Language (UML) as an actor) and a system to achieve a goal. The actor can be a human or other external system. In systems engineering, use cases are used at a higher level than within software engineering, often representing missions or stakeholder goals. The detailed requirements may then be captured in the Systems Modeling Language (SysML) or as contractual statements.”

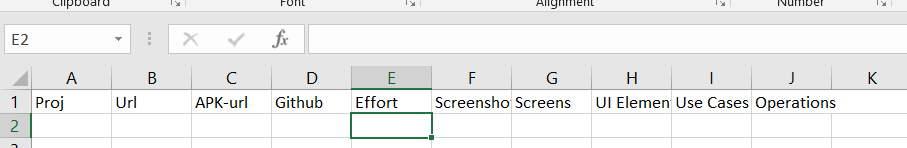
Key points:

1. Identify your role (unregister user, registered user, admin, etc).
2. The goal you are trying to achieve, for example, registration, login, reset password, read articles, etc. Each of such goal is a use case.
3. For each of APK, create a sheet as below in the “Use\_Case\_Analysis\_Results” google spreadsheet: <https://docs.google.com/spreadsheets/d/1D5R1_8BrvwRv33AdQ_LawW0c6JUT4Rk1SaEoj5WdZZc/edit#gid=0>



The template provides the example fields needed:

1. Screens. Count number of screens in the app to realize a use case. You can use the emulator and Vysor to capture the screenshot which should be convenient. Name the screenshot following the rule: UseCaseName-1.png. “1” represents the order you see the screen for the use case.
2. Input Elements. The number of input fields in the screen for the use case.
3. Output Elements. The number of views that display the information you need for the use case.
4. UI Elements. The total number of UI elements, which is a sum of the Input elements and output elements.
5. Operations. The number of movements on the screen for you to complete the use case.
6. Sum up the use cases to update the columns in the “summary” tab.



Download the instrumented apps:

1. Find the list of android projects from:

<https://docs.google.com/spreadsheets/d/1D5R1_8BrvwRv33AdQ_LawW0c6JUT4Rk1SaEoj5WdZZc/edit#gid=0>

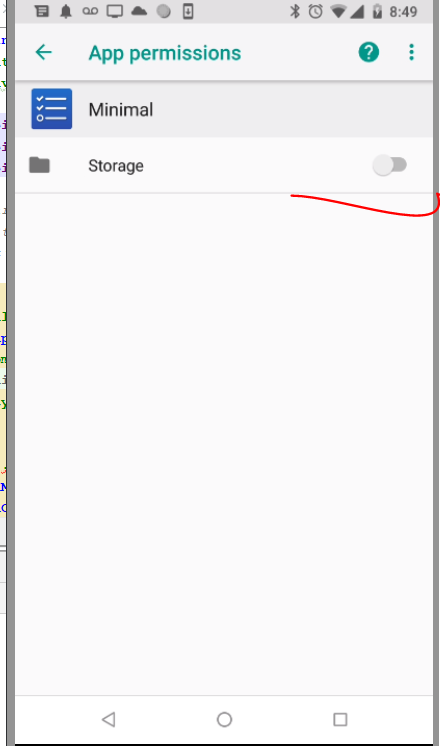
1. Download the instrumented apk from:

<https://drive.google.com/drive/u/1/folders/1AeJv3FJ3epeP-zrSkUy_5cPeLYNZjAh5>

1. Deploy the instrumented app onto your phone or emulator using command:

adb install app.apk

Make sure to give the written permission to the app.



1. Operate the apps according to the use cases that you identified before. Record the time that you operate the use cases, for example, according to the following format:

2/5/2019 12:51:27 Start Use\_Case\_Name

2/5/2018 12:52:21 End Use\_Case\_Name

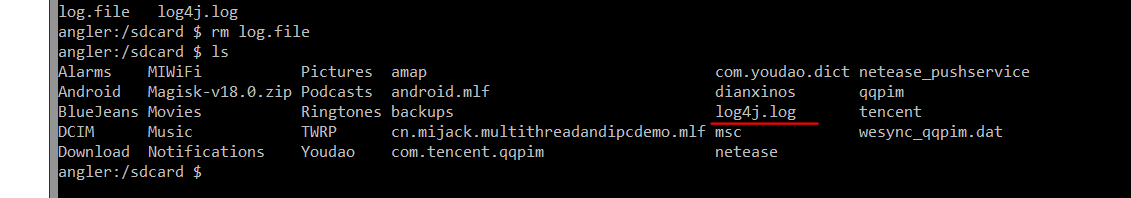
\*The time should precise as to “seconds”.

1. Download the logs from your phone or emulator:

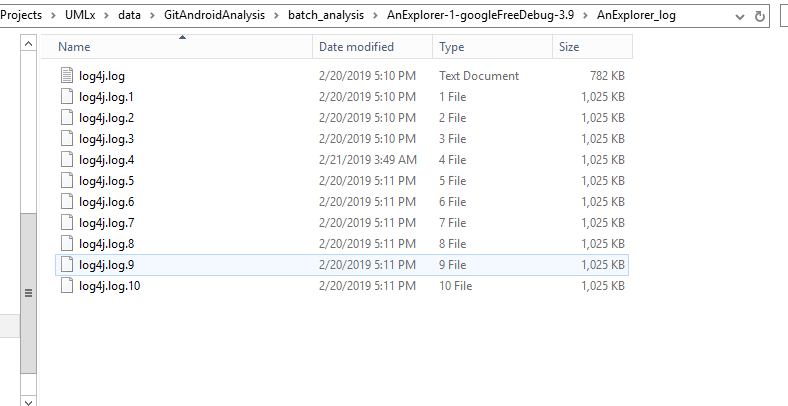
Run this command:

adb pull /sdcard/log4j.log

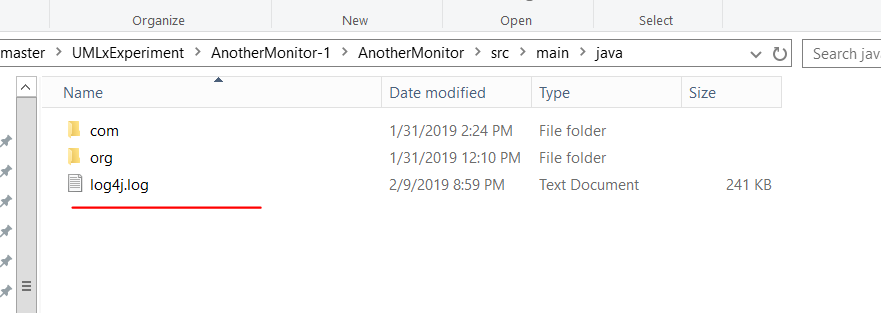
There may be multiple files which share the similar names (have different index numbers). Download them all.



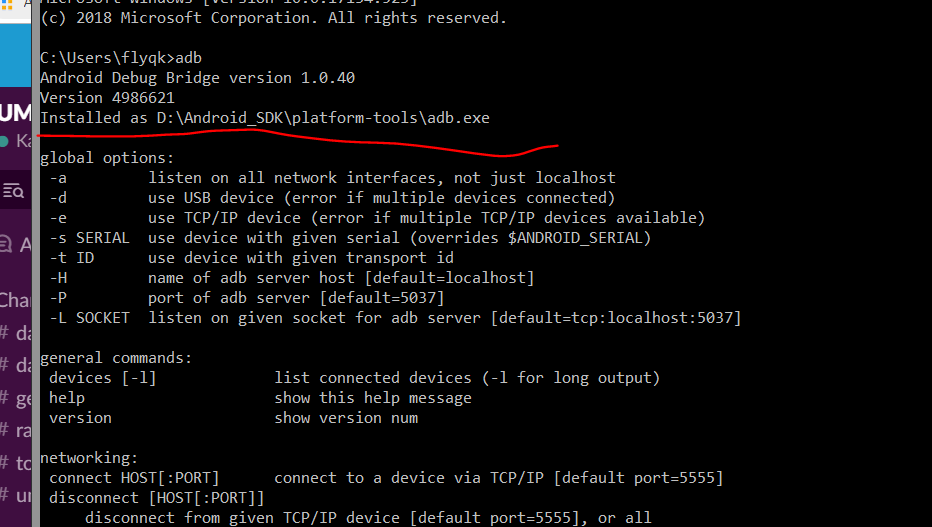
Multiple log files would be generated in the /SDcard folder, for example:



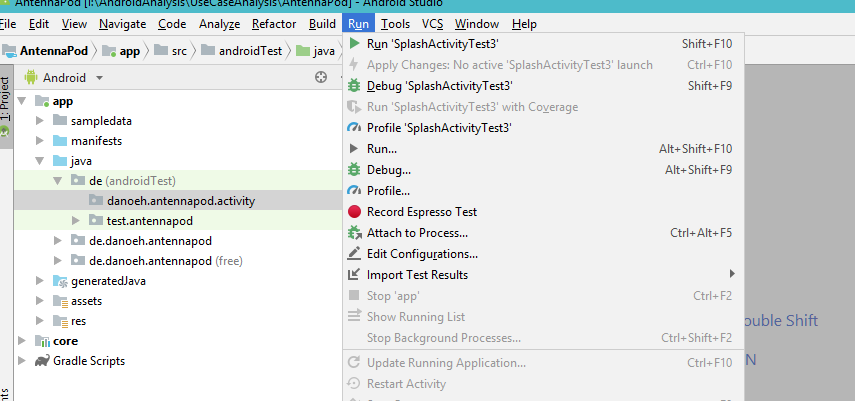
You will find the log in your local drive:

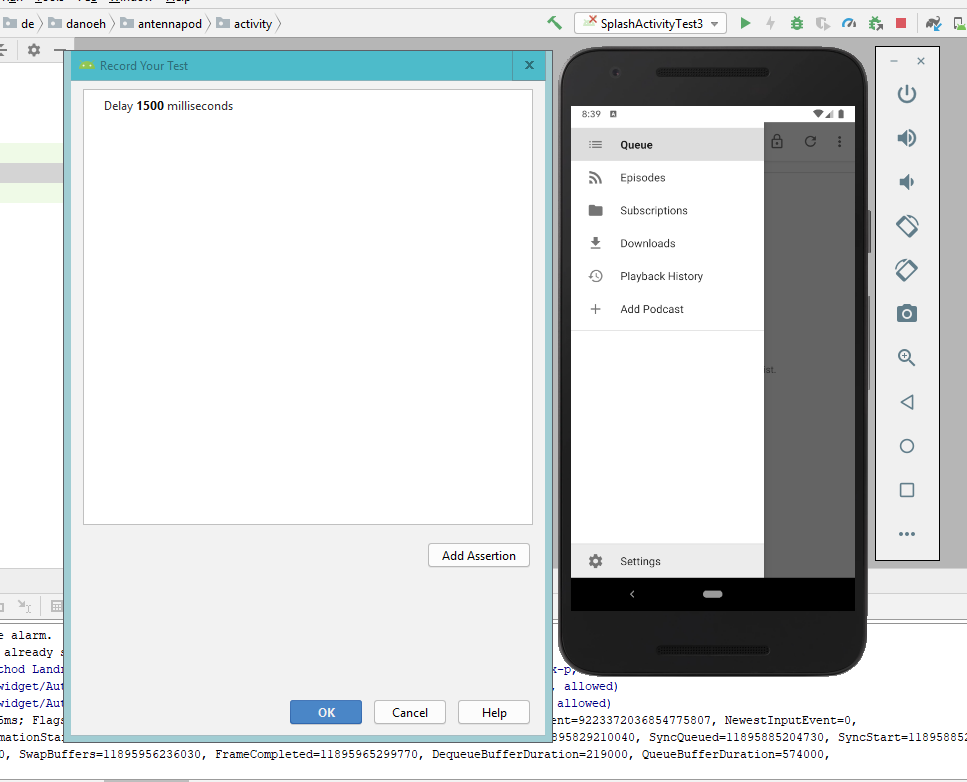


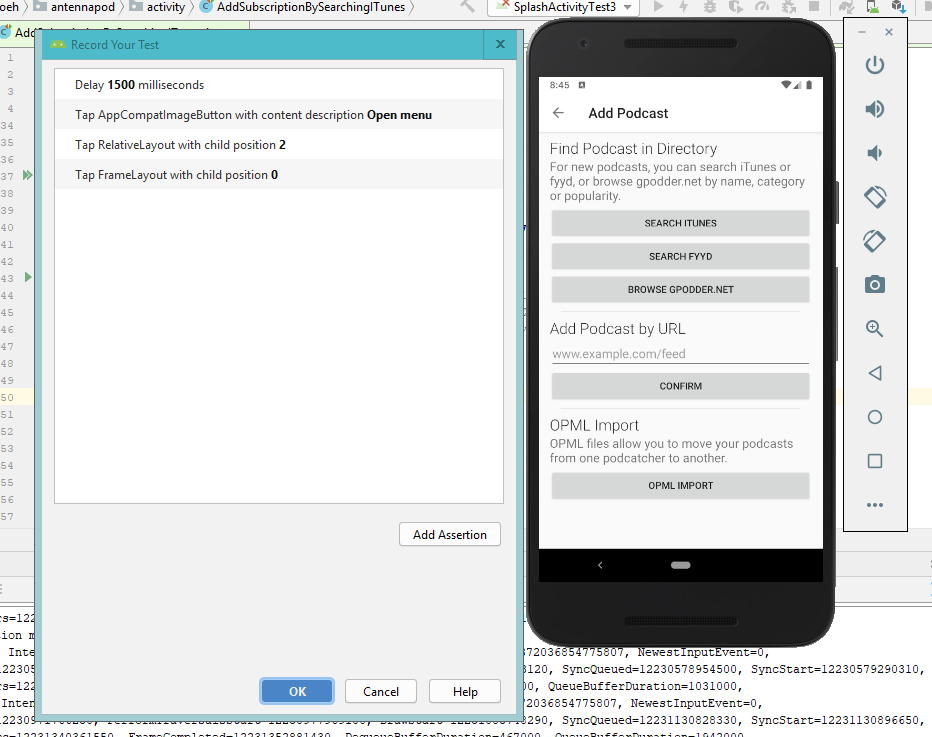
if you meet the problem of not finding command "adb”, please find and add the path to environmental variables.

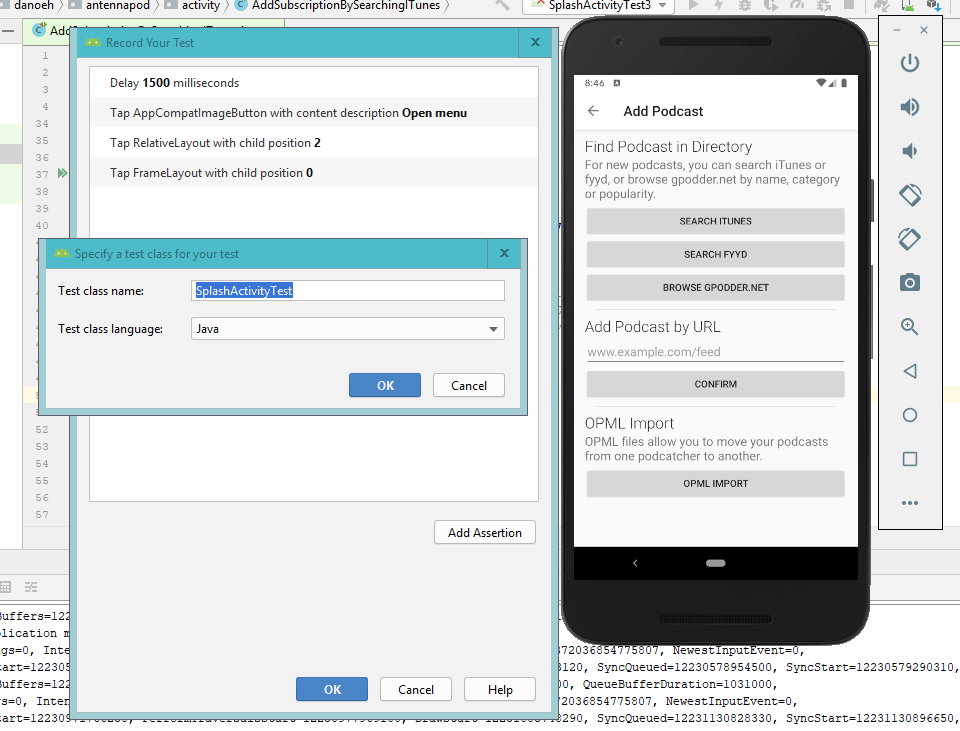


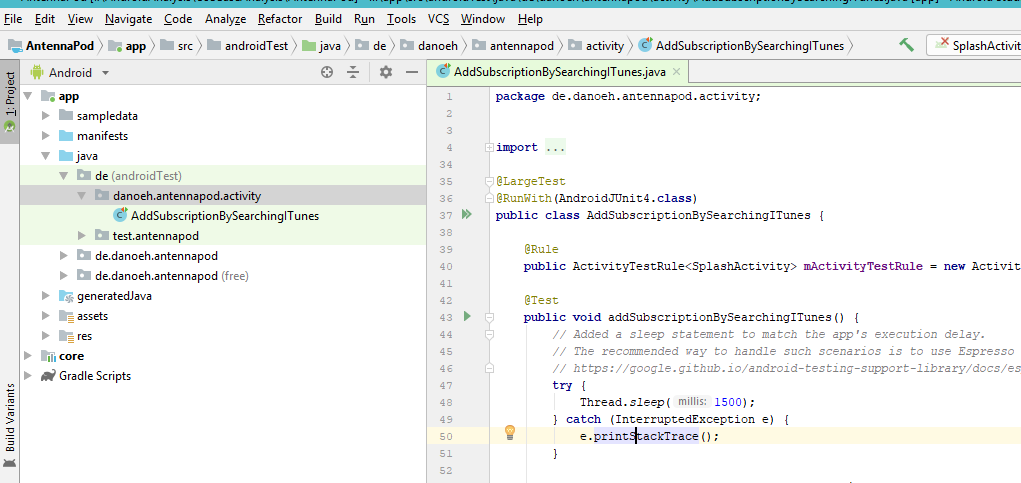
1. Create espresso test cases for the use cases you identified. For each use case, creae one espresso test case

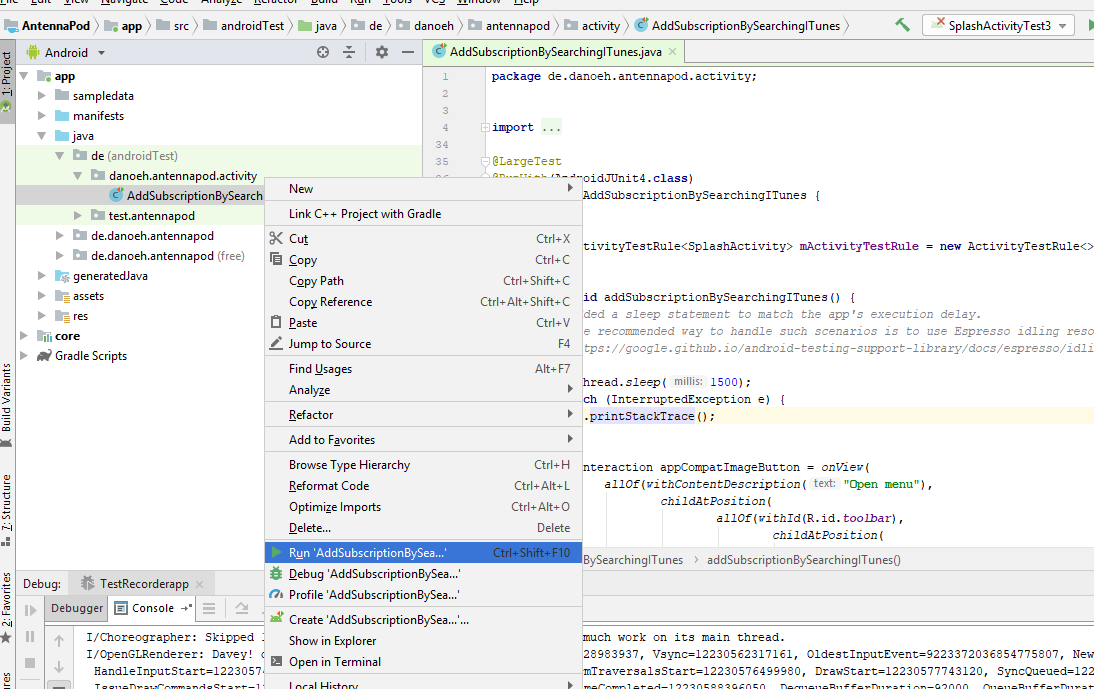












1. After getting the logs and use case records:

Please upload the logs into the folder (put it into a sub-folder using the app’s name):

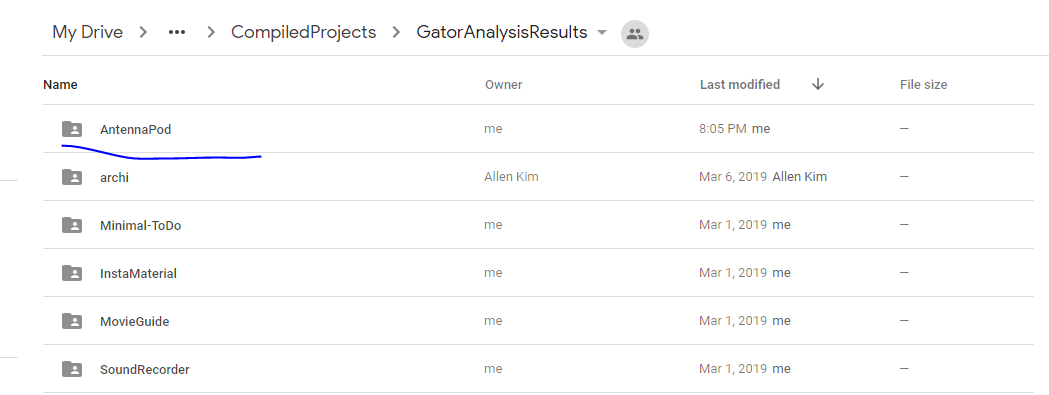
<https://drive.google.com/drive/u/1/folders/1hSNnY0TwRGkSdrN24STuBy_UZxyCt_AS>

and update the datasheet

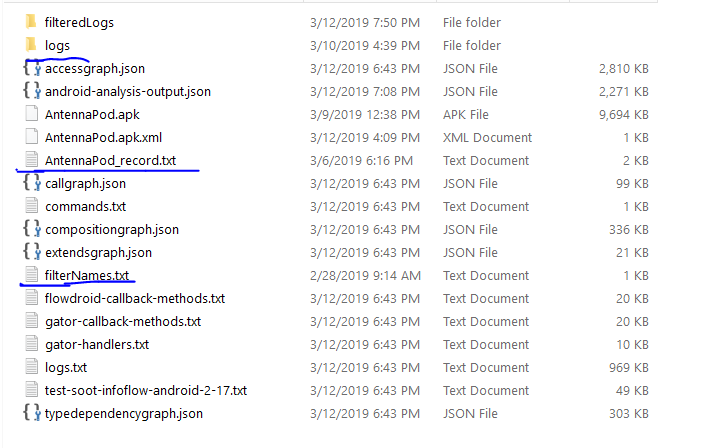
<https://docs.google.com/spreadsheets/d/1D5R1_8BrvwRv33AdQ_LawW0c6JUT4Rk1SaEoj5WdZZc/edit#gid=0>

1. Analyze the log to identify transactions.
   1. Download the corresponding analysis package from:

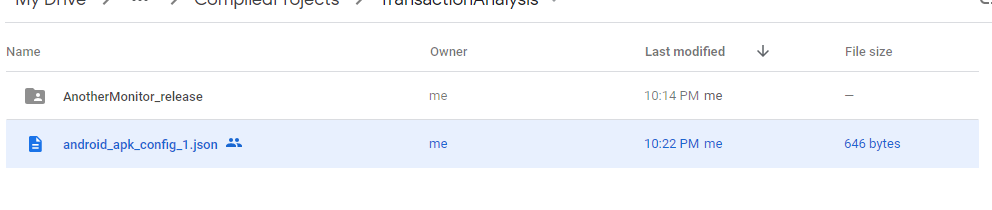
<https://drive.google.com/drive/u/1/folders/1fESD7ylHh0v5zFlRQV18eBbeWKqlhjjm>

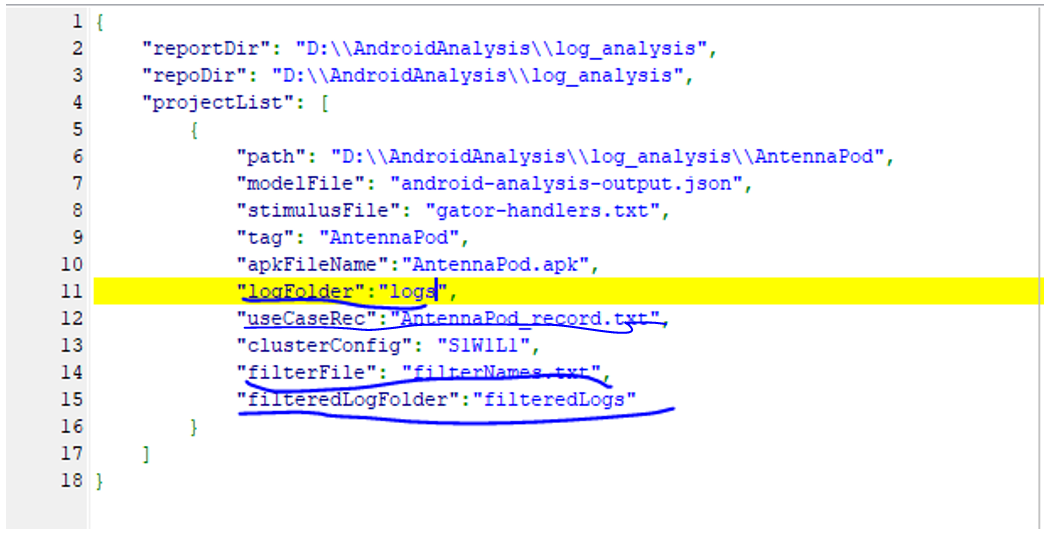


For example, for “AntennaPod”, there are list of files in the folder. Download the folder to your local machine and put the logs you generated from the previous steps.

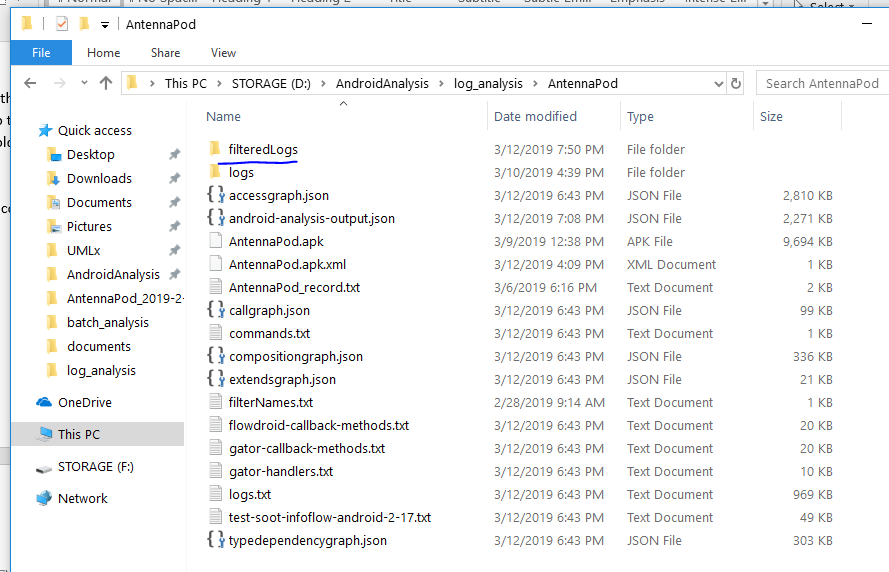


Modify the json file within the folder, and references to the log files for the following analysis(an example folder is provided in the google drive folder):





\*copy “filterNames.txt” file into the analysis package, which is provided in the google drive folder. “logFolder” should be pointed to the folder you created in last step that holds your log files. “filteredLogFolder” will be the folder hold the filtered log files. “useCaseRec” should be pointed to the use case record file you created in the previous use case analysis steps.



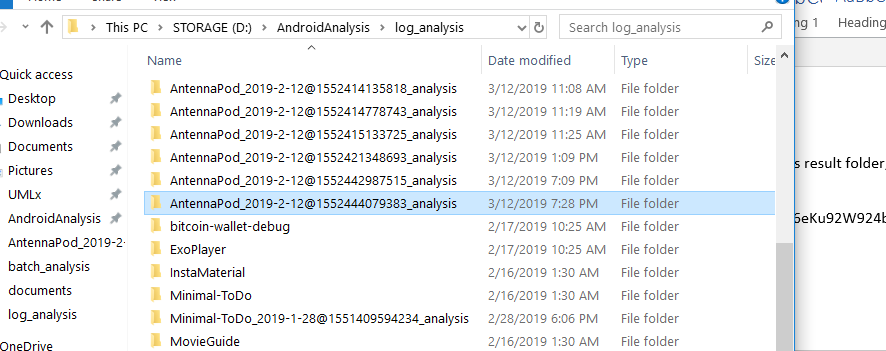
1. Preprocess the log.

node --max\_old\_space\_size=10240 ".\utils\AndroidProjectAnalysis.js" --filter-logs "D:\\AndroidAnalysis\\log\_analysis\\android\_apk\_config\_1.json"

\*replace the paths in the command line with your local path.

1. Apply the transaction analysis results.

node --max\_old\_space\_size=10240 ".\utils\AndroidProjectAnalysis.js" --analyse-android-projects "D:\\ResearchSpace\\ResearchProjects\\UMLx\\data\\GitAndroidAnalysis\\log\_analysis\\ android\_apk\_config\_1.json"



Upload the configuration file folder, the generated transaction analysis result folder, and the configuration file onto Vocareum at this location:

<https://drive.google.com/drive/u/1/folders/1XGECxoeEmUqdz_-0PUT5C_x9f1NKSFU0>