Use Case Analysis Tutorial

Following the steps:

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

<https://www.bluestacks.com/>

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>

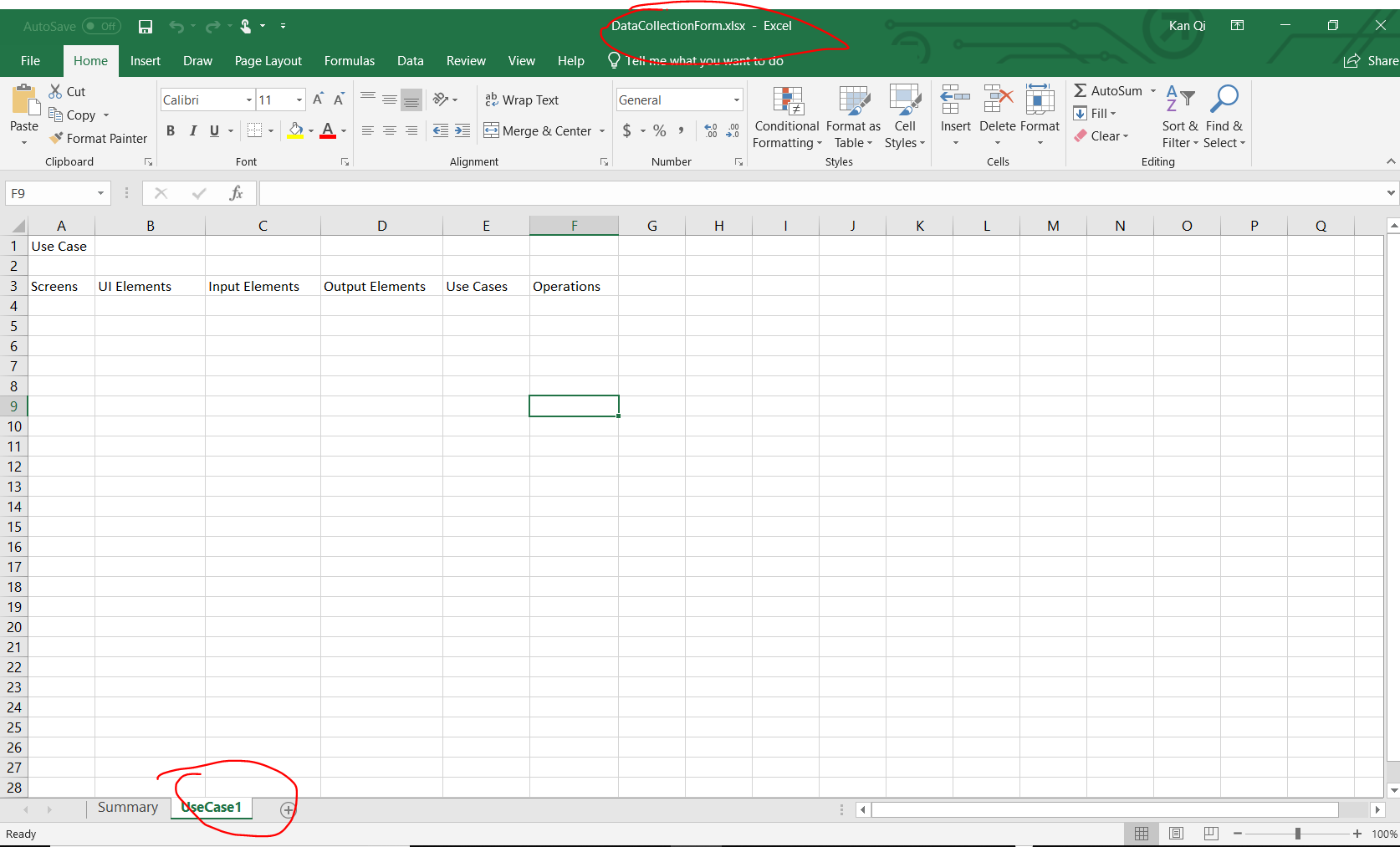
“In software and systems engineering, a use case is a list of actions or event steps typically defining the interactions between a role (known in the Unified Modeling Language (UML) as an actor) and a system to achieve a goal. The actor can be a human or other external system. In systems engineering, 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.”

Keypoints:

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 the goal is a use case.

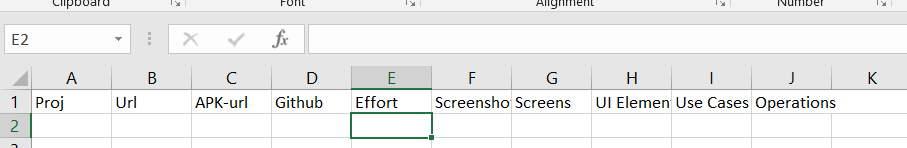
Examples of use cases.

1. For each of the use case, create a sheet as below in the DataCollectionForm.xlsx



The template provides the example fields needed:

1. Screens. Take screenshot for each screen 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.



[no need for step 3]

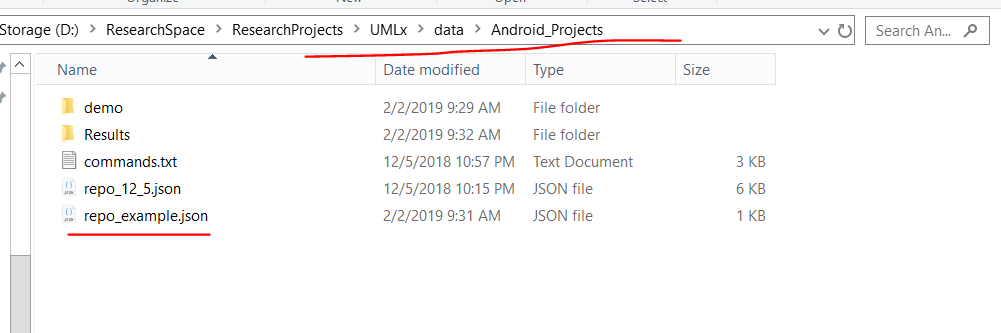
1. Run the following command to derive data from github for the effort:

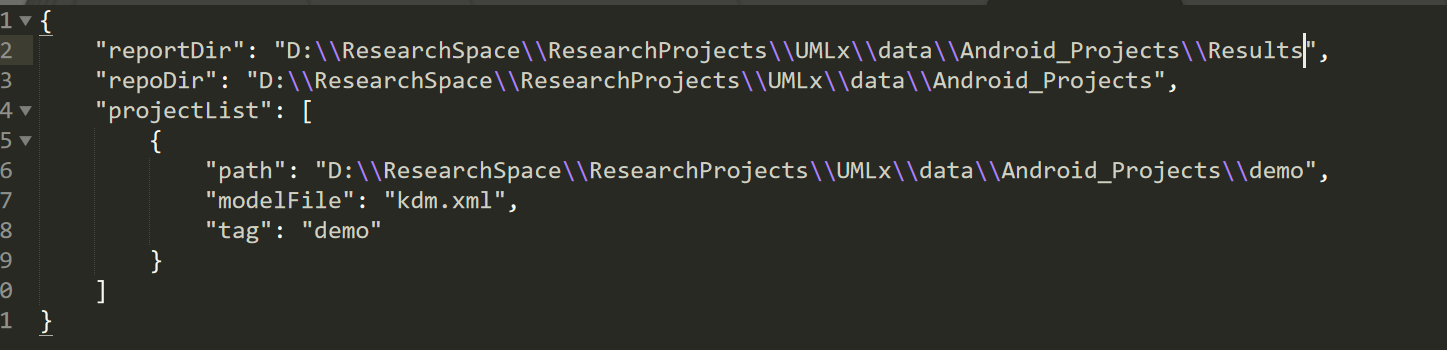
"C:/Program Files/R/R-3.2.5/bin/Rscript" ./UMLx/data/OpenSource/active\_contributors\_every\_30.R "Github-repo-api-url" "./temp/git\_effort\_request\_results.txt"

1. Run the command below to derive the data for lines of source code.

Command to source code of code.

1. Create a config file to reference the project source.





* You can add more projects to this configuration file to further do the analysis.

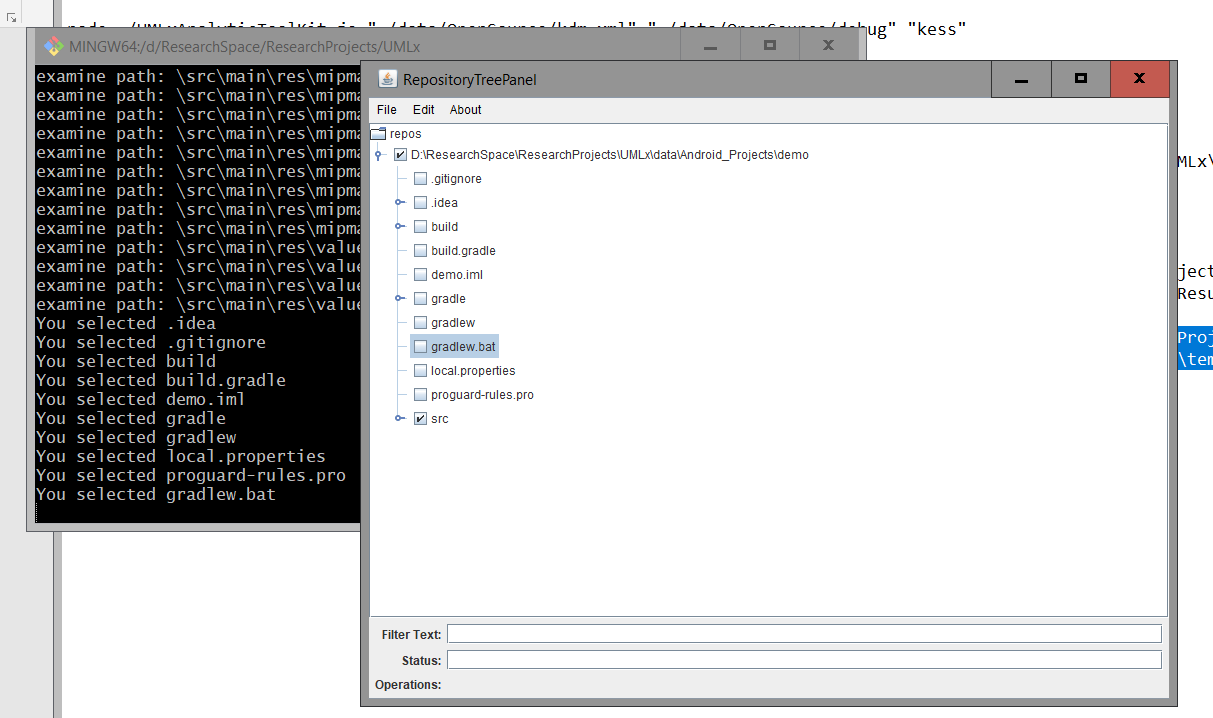
1. Scan the repo

node --max\_old\_space\_size=10240 ".\utils\OpenSourceProjectAnalysis.js" --scan-repo "D:\\ResearchSpace\\ResearchProjects\\UMLx\\data\\Android\_Projects\\repo\_example.json"

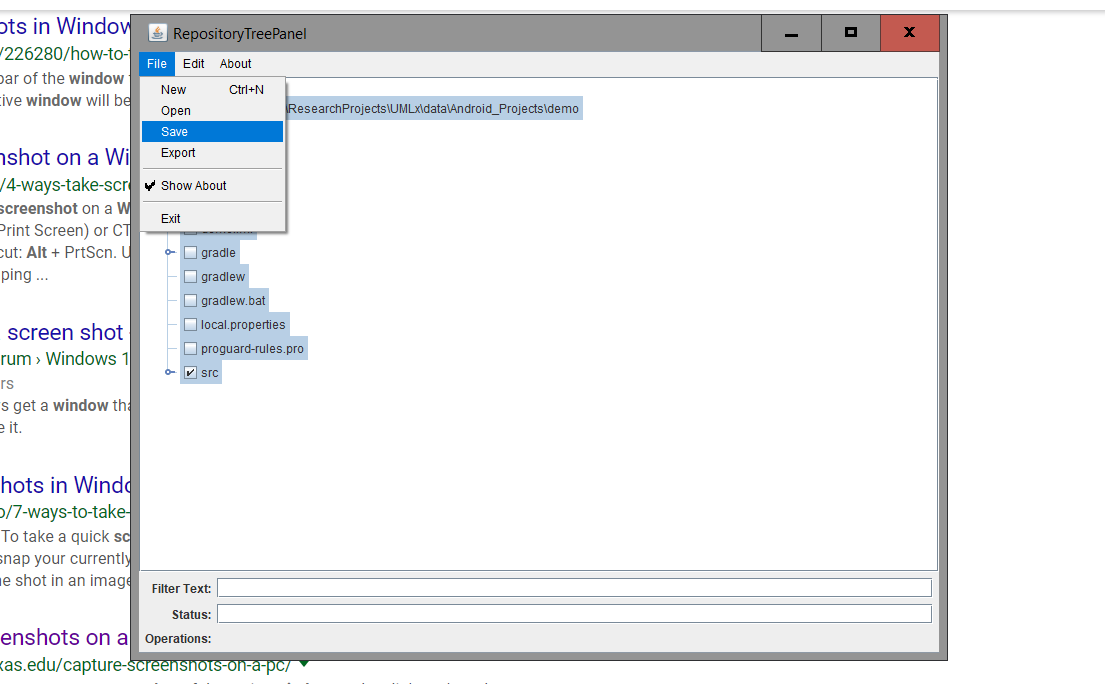
[skip this step]

1. Select the source code files using the following command:

node --max\_old\_space\_size=10240 ".\utils\OpenSourceProjectAnalysis.js" --select-files "D:\\ResearchSpace\\ResearchProjects\\UMLx\\data\\Android\_Projects\\repo\_example.json"

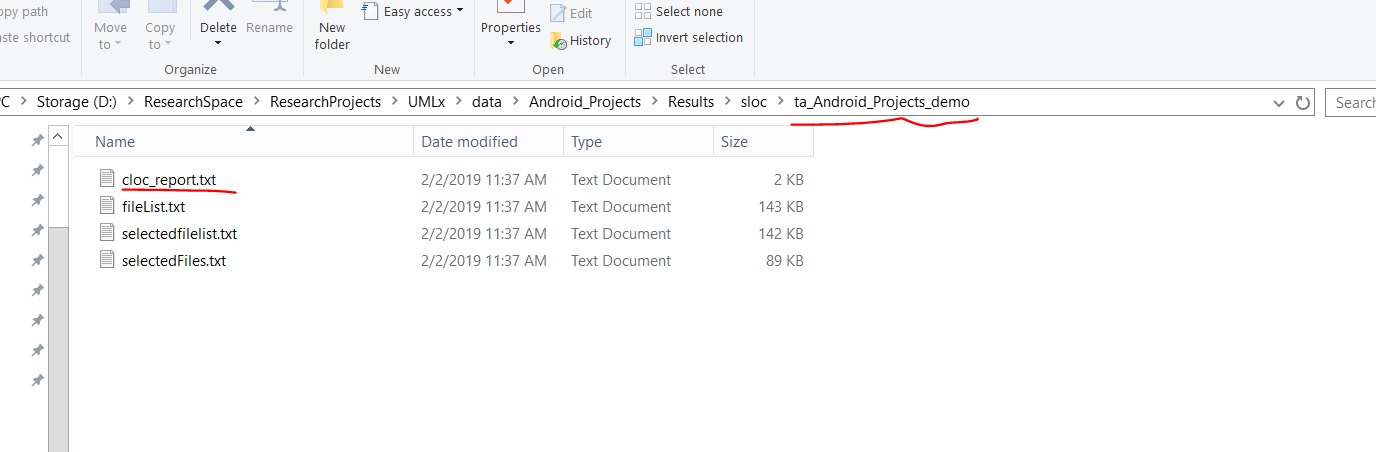


Save the selected files



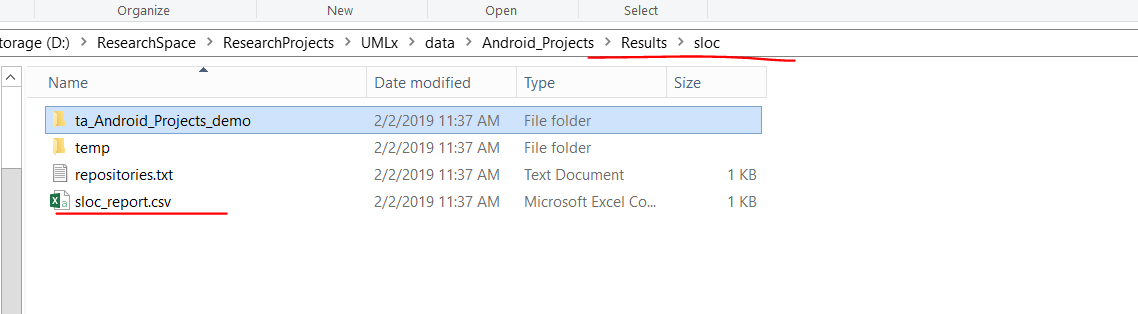
1. Analyse the lines of source code:

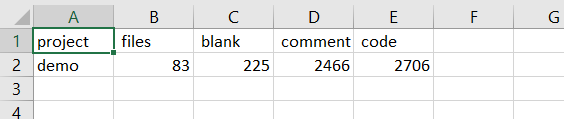
node --max\_old\_space\_size=10240 ".\utils\OpenSourceProjectAnalysis.js" --analyse-sloc "D:\\ResearchSpace\\ResearchProjects\\UMLx\\data\\Android\_Projects\\repo\_example.json"



1. Generate the report

node --max\_old\_space\_size=10240 ".\utils\OpenSourceProjectAnalysis.js" --generate-sloc-report "D:\\ResearchSpace\\ResearchProjects\\UMLx\\data\\Android\_Projects\\repo\_example.json"





1. Run the R script to understand the correlations and the stepwise model

UMLx\data\FeatureSelection\gzh\_report.Rmd

**Output operational logs [Updated]**

Download the instrumented apps:

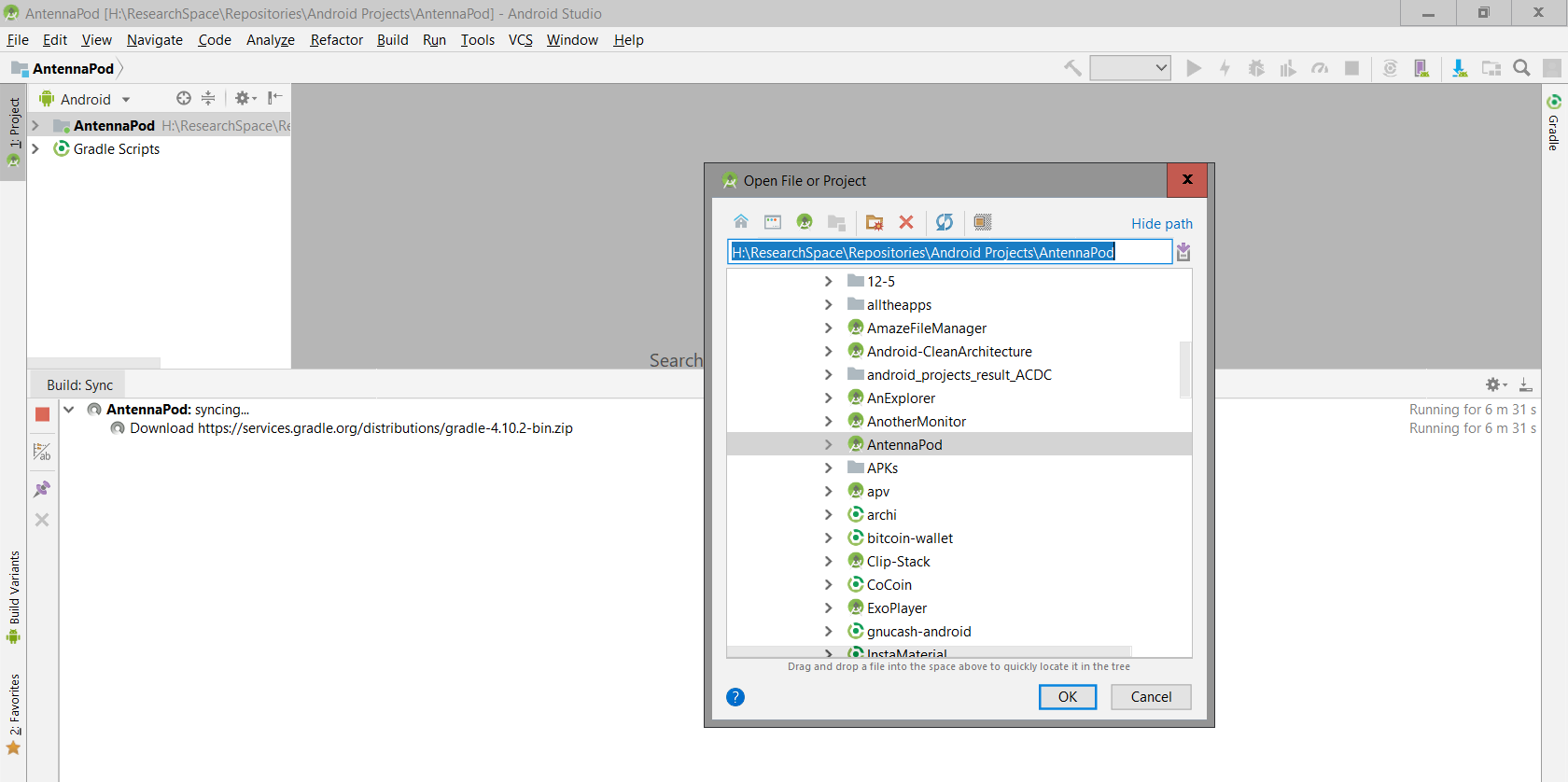
1. Find the list of android projects from:

https://docs.google.com/spreadsheets/d/14-YLB1cqJ7l67AZ-gfH-q4po2krPeQ5K68jnk5YevrM/edit#gid=0

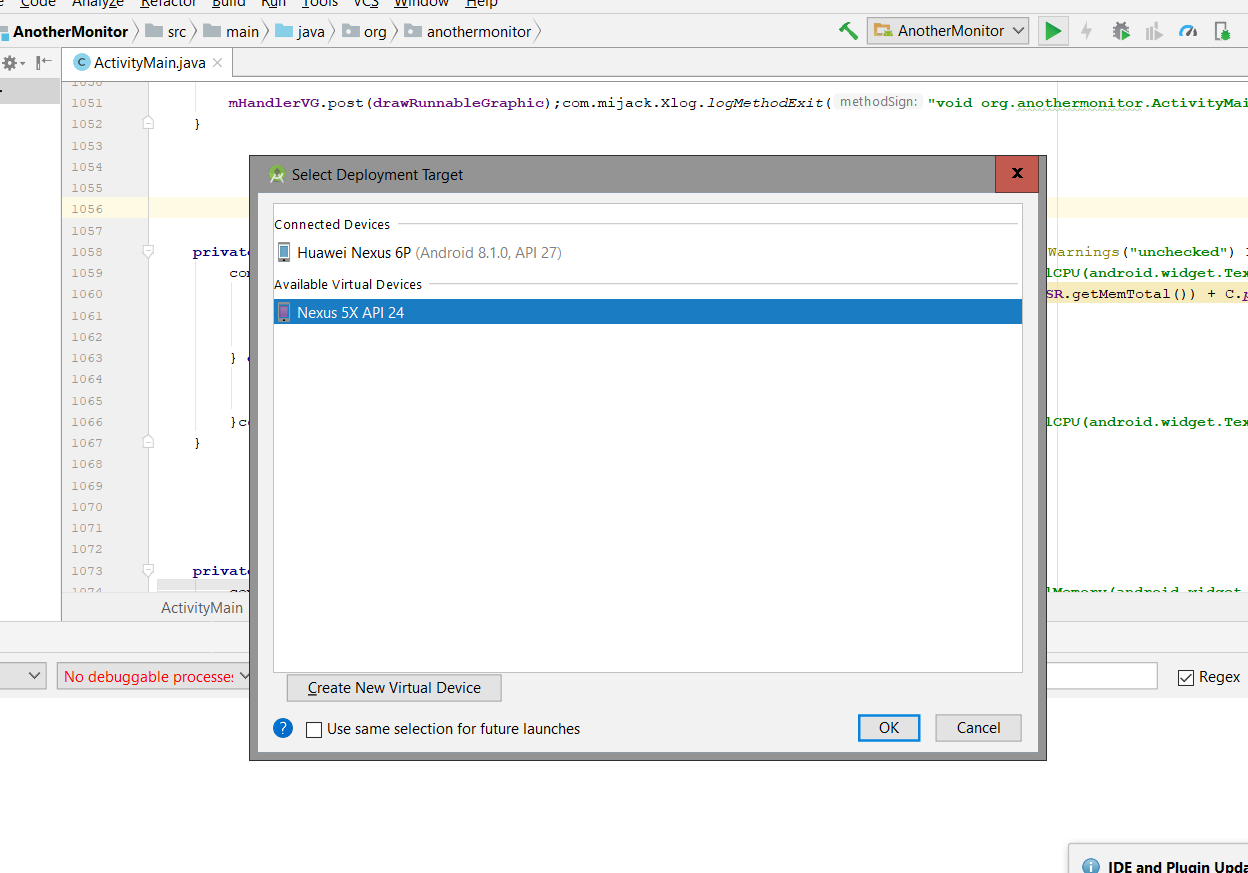
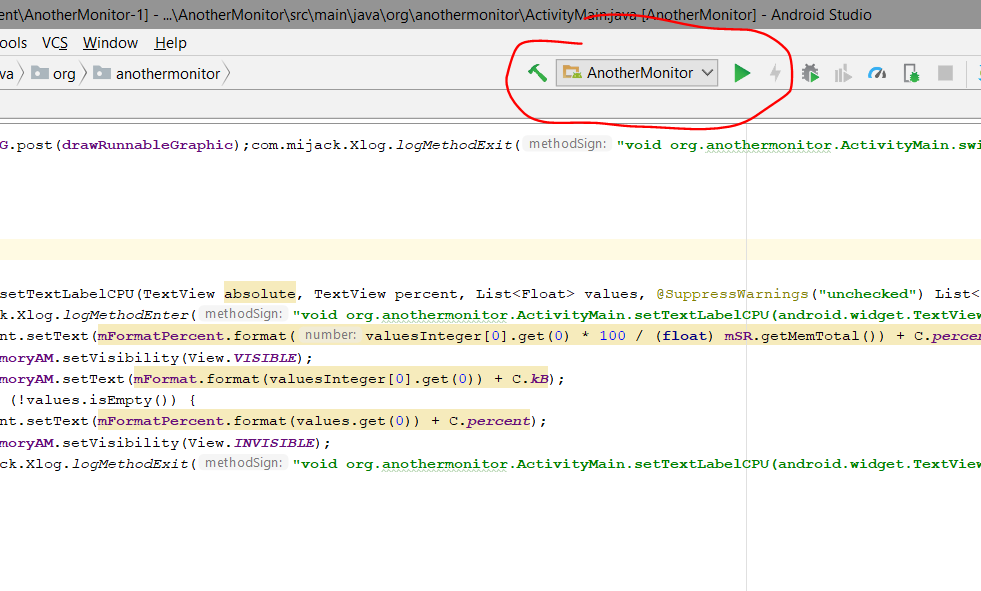
1. [no need for this step] Download the source code from Url

<https://drive.google.com/drive/u/1/folders/11-XYoIbd6IU1zN26S--g1E12od1WIU6F>

Import the android project into android studio:



Deploy the instrumented app onto your phone or virtual android phone.



1. Download the instrumented app from:

<https://drive.google.com/drive/u/1/folders/11-XYoIbd6IU1zN26S--g1E12od1WIU6F>

and run command:

adb install app.apk

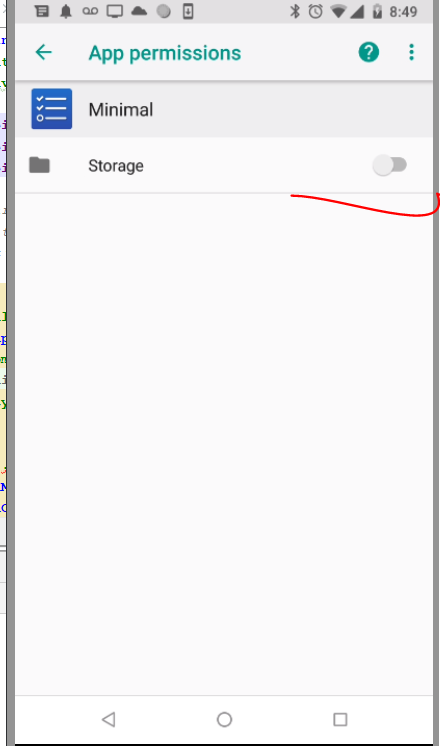
1. Operate the apps according to the use cases that you identified before. Record the time that you operate the use cases, for example, according to the following format:

2/5/2019 12:51:27 Start Login

2/5/2018 12:52:21 End Login

* Second is important to record.

1. Install the app and make sure to give the written permission to the app.

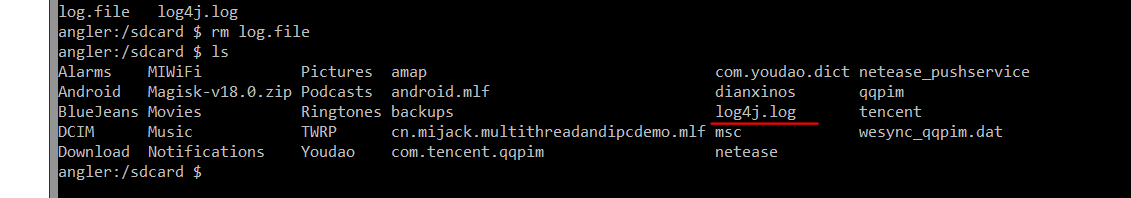


1. Download the logs from your phone or emulator:

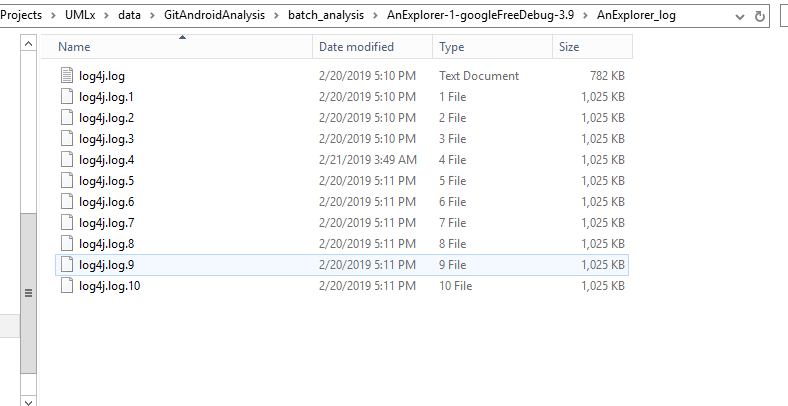
Run this command:

adb pull /sdcard/log4j.log

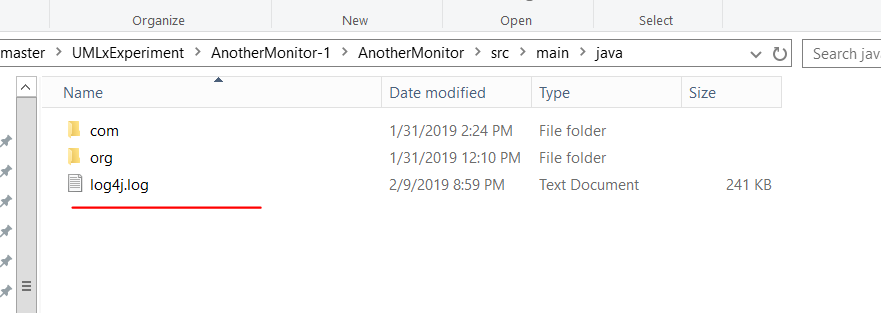
There may be multiple files which share the similar names (have different idex numbers). Download them all.



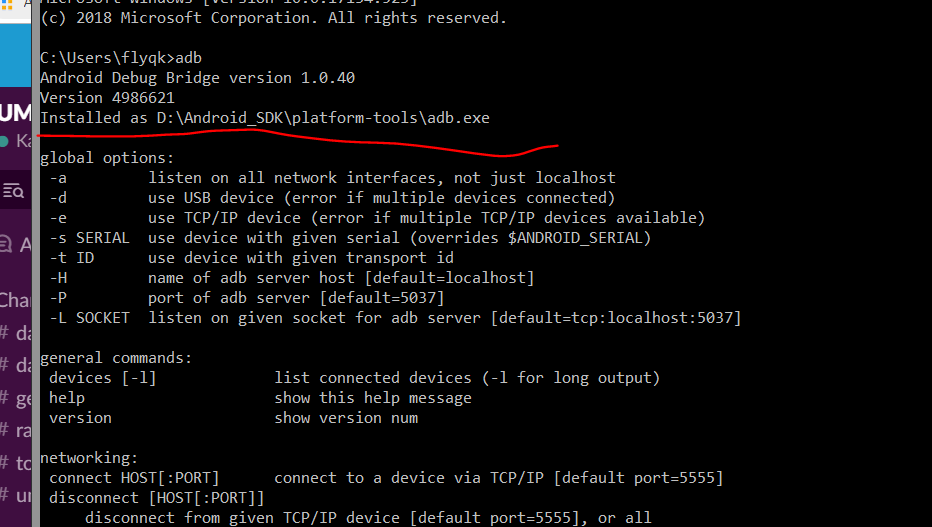
Multiple log files would be generated in the /SDcard folder, for example:



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. After getting the logs and use case records:

Please upload the logs into the corresponding folders:

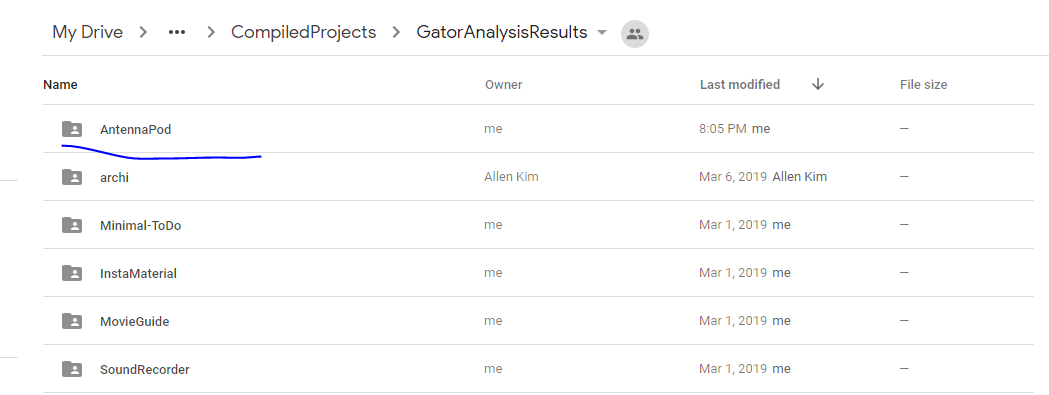
<https://drive.google.com/drive/u/1/folders/1ntO1K6klOA8buTm-NBhqVRr1TXvNg7da>

and update the datasheet

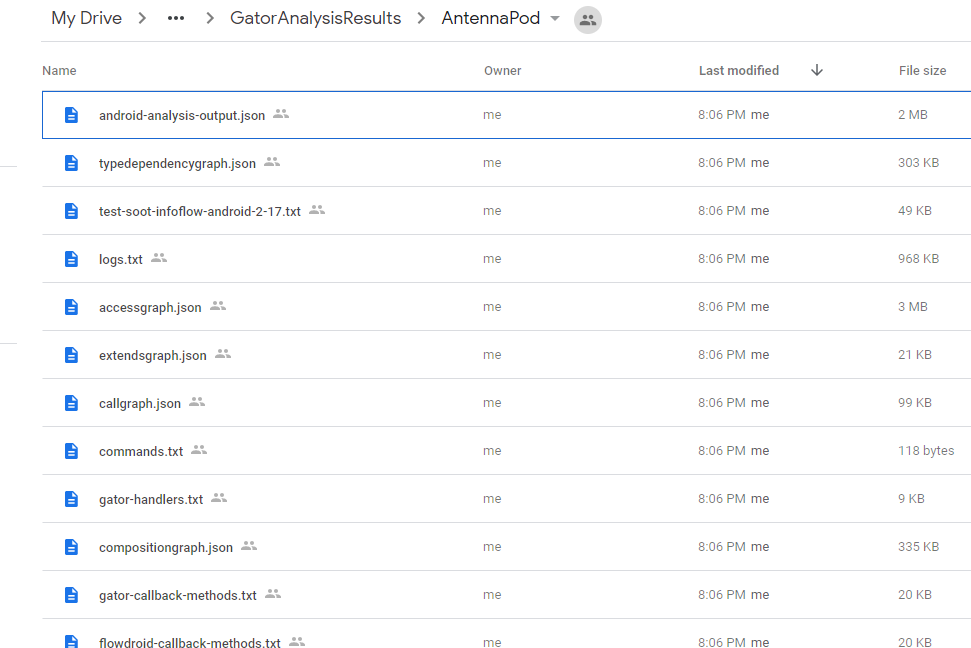
<https://docs.google.com/spreadsheets/d/14-YLB1cqJ7l67AZ-gfH-q4po2krPeQ5K68jnk5YevrM/edit>

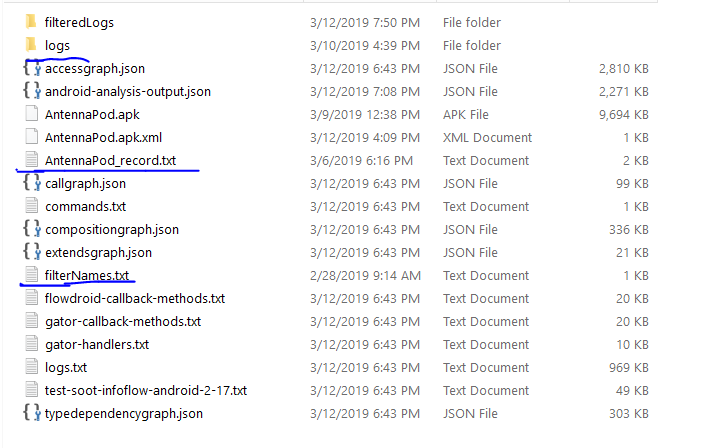
1. Analyze the log for identifying the transactions.
   1. Download the analysis package from:

<https://drive.google.com/drive/u/1/folders/1Y1-eWs4lL92z4RmCZkHWSPEnvpsNM8N2>

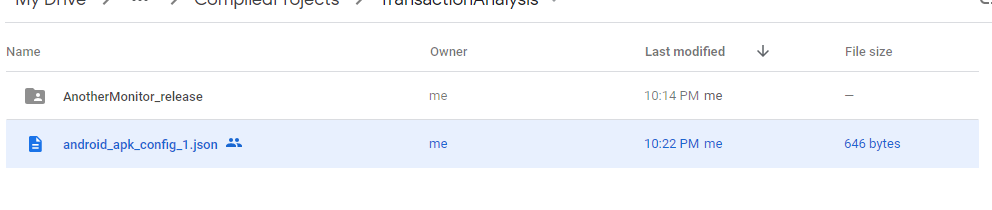


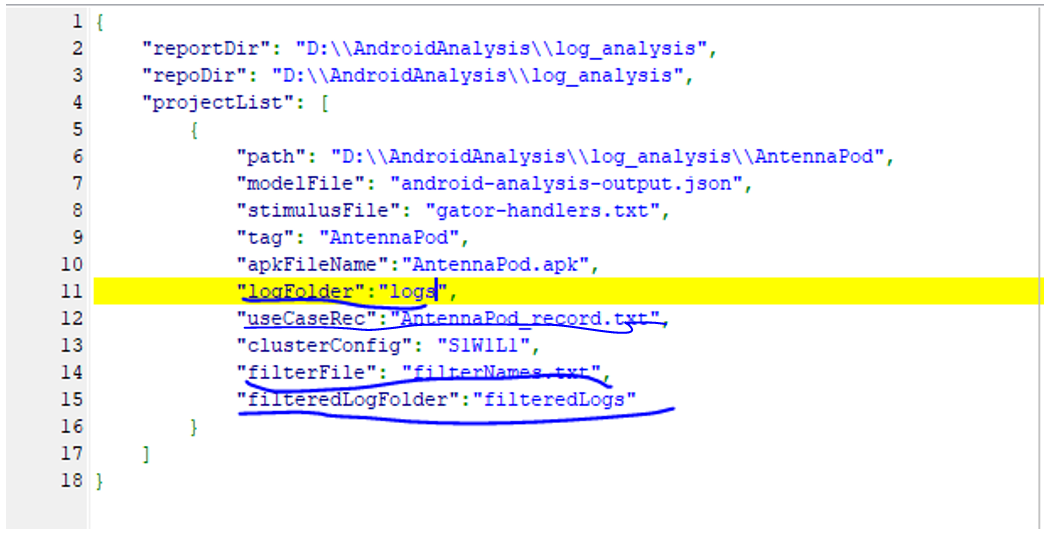
For example, for “AntennaPod”, there are list of files in the folder. Create a folder to put the logs you generated from the previous steps.



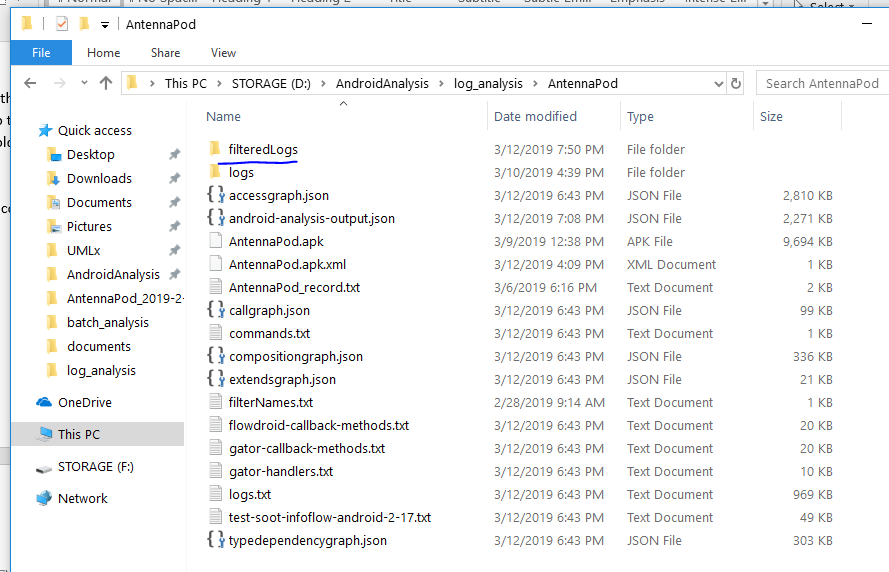


create a json configuration file to point to the files for the following analysis (an example json configuration file is provided in the google drive folder):





\*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.



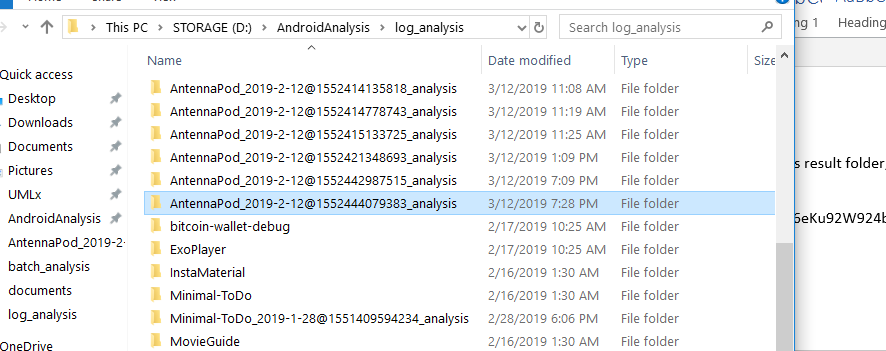
1. Preprocess the log.

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/1wPLoziL0QMa-rVvoVGO6eKu92W924blQ