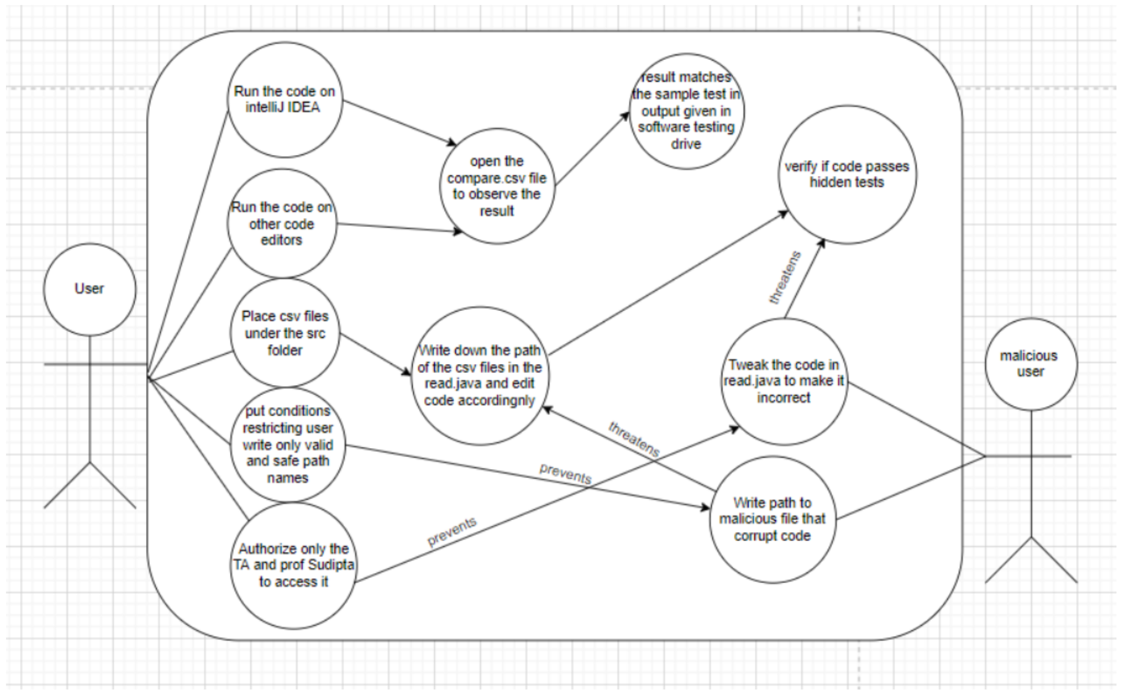
Equivalence Class Partitioning


User
Run the code on intelliJ IDEA
Run the code on other code editors
Place csv files under the src folder
put conditions restricting user write only valid and safe path names
Authorize only the TA and prof Sudipta to access it
open the compare.csv file to observe the result
result matches the sample test in output given in software testing drive
Write down the path of the csv files in the read.java and edit code accordingnly
verify if code passes hidden tests
threatens
Tweak the code in read.java to make it incorrect
threatens
prevents
prevents
Write path to malicious file that corrupt code
malicious user

From the use case diagram above, we can look at the requirement which is putting conditions restricting users to write only valid and safe path names. Hence we divide into classes, **one class with the valid path of csv files and another class with the invalid path of CSV files**

- Divide input conditions into two classes
  - Valid path of sample_file_1.csv & sample_file_3.csv
    - Ex : `"./src/sample_file_1.csv"`
    - Ex : `"./src/sample_file_3.csv"`
  - Invalid path of sample_file_1.csv & sample_file_3.csv
    - Ex : `"./xml/sample_file_1.csv"`
      - `(csv file not stored under xml)`
    - Ex : `"./.idea/sample_file_1.csv"`
      - `(csv file not stored under .idea)`
    - Ex : `"./Intellijideaprojects/sample_file_1.csv"`
      - `(invalid directory- Intellijideaprojects)`

Valid path of sample_file_1.csv & sample_file_3.csv
When we input valid path names of the CSV files in the scanner class, we can expect a **compare.csv** with the valid result with the **differences between sample_file_1.csv & sample_file_3.csv to be generated**
Invalid path of sample_file_1.csv & sample_file_3.csv
However, when we input the Invalid path names of the CSV files in the scanner class, **an empty compare.csv will be generated not listing the differences between sample_file_1.csv & sample_file_3.csv**.

Boundary Value Analysis

- Divide input conditions into two classes
    - Valid path of sample_file_1.csv & sample_file_3.csv
        - Middle value
            - `"./src/sample_file_1.csv"`
        - boundary value
            - `"C:\Users\sahana\IdeaProjects\ESC_Project_1\src\sample_file_1.csv"`
    - Invalid path of sample_file_1.csv & sample_file_3.csv
        - Middle value :Wrong path (sample_file_1.csv not under xml)
            - Ex : `"./xml/sample_file_1.csv"`
        - Boundary Value: when we pass the name of the file that does not exist
            - Ex : `sample_file_8.csv`

<u>The first class contains the valid path of sample_file_1.csv & sample_file_3.csv.</u>
The **middle value** is the **shortened path of csv file which is truncated** and starts with ./ to allow the file to be read.
The **boundary value** is the **maximum length of the path of the csv file** and the **actual full path** of the csv file.
In conclusion for both the middle and boundary value in the first class,when we input valid path names of the CSV files in the scanner class, we can expect a compare.csv with the valid result with the differences between sample_file_1.csv & sample_file_3.csv to be generated

<u>The second class Invalid path of sample_file_1.csv & sample_file_3.csv.</u>
The **middle value contains the wrong path with the wrong directory "XML"**: `"./xml/sample_file_1.csv"`.
The **boundary value is which is the name of csv file** `sample_file_8.csv` which does not exist
In conclusion for both the middle and boundary value in the second class when we input the Invalid path names of the CSV files in the scanner class, an empty compare.csv will be generated not listing the differences between sample_file_1.csv & sample_file_3.csv.