Analysis of Context-Aware Security for Insider Attacks in Smart Offices

(ACASIA)

User Manual

Created by:

Azevedo, Marquus Joss R.

Del Castillo, Jose Mari L.

Dumas, Francisco Emmanuel T.

Malia, Gideon Andrew L.

Advised by:

Ms. Arlyn Verina O. Tiu

Table of Contents

1. Prerequisites

ACASIA requires to be run in Windows and Python libraries (not in standard library) to be installed in order to run. Two programming IDEs will be recommended in order to run the project and view how it works in real-time.

1.1. Python Version

Python 3.12.4: Download here.

1.2. Python Libraries

There will be multiple Python libraries required to be installed which are:

- pandas, matplotlib, seaborn, scikit-learn, scipy, numpy, umap-learn, and filelock

These can be installed by being in the ACASIA main project folder and executing "***pip install -r reqs.txt***" in the terminal.

1.3. Programming IDE Recommendations

It is recommended to use the following programming IDEs at the same time for a better experience:

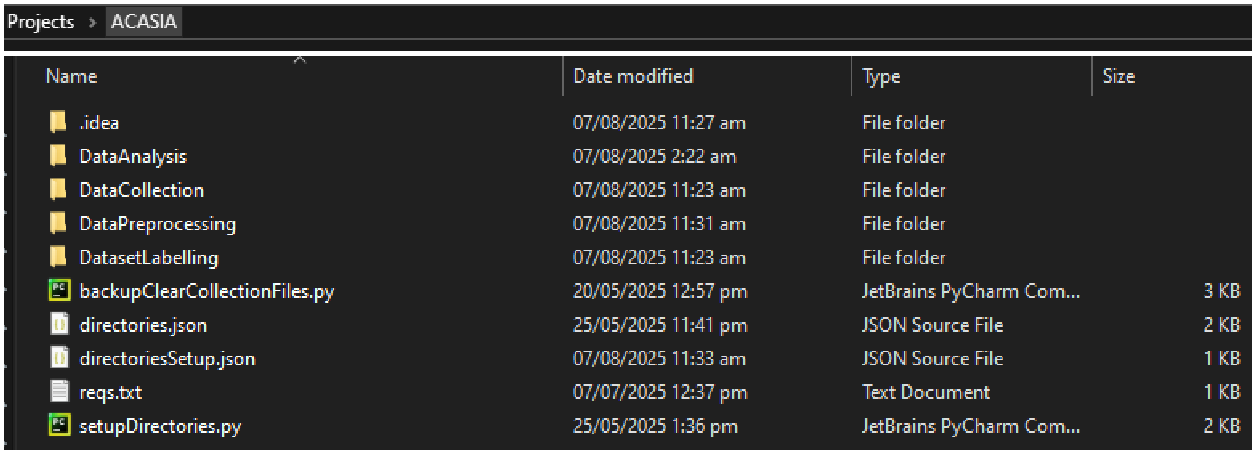
- PyCharm Community Edition (PyCharm)
    - Data collection, preprocessing and labelling will best perform in PyCharm.
- Visual Studio Code (VS Code)
    - Best suited for real-time viewing of collection as VS Code can refresh file viewing. Also used to execute the data analysis module once collection, preprocessing, and labelling is finished.

2. Project Folder File Structure

The ACASIA project can be downloaded from:

ACASIA Download Link

It will contain the setup files and folders as the main components of the project.

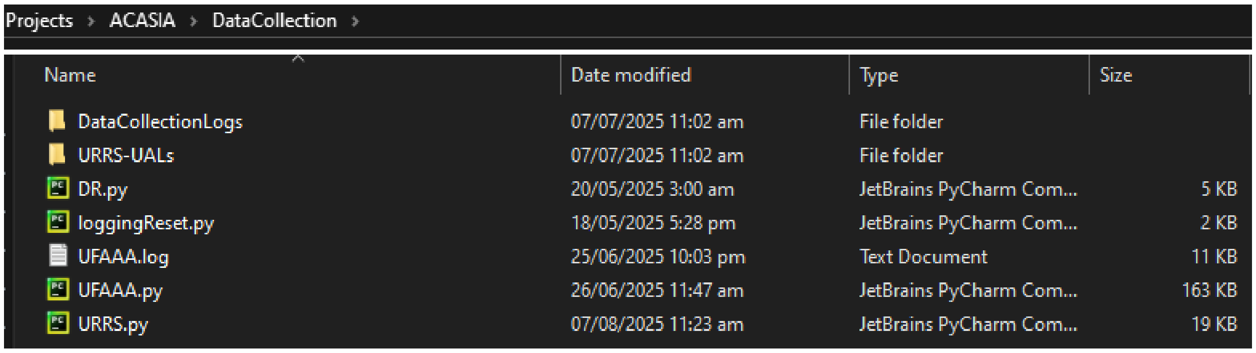


Projects > ACASIA

Name	Date modified	Type	Size
.idea	07/08/2025 11:27 am	File folder	
DataAnalysis	07/08/2025 2:22 am	File folder	
DataCollection	07/08/2025 11:23 am	File folder	
DataPreprocessing	07/08/2025 11:31 am	File folder	
DatasetLabelling	07/08/2025 11:23 am	File folder	
backupClearCollectionFiles.py	20/05/2025 12:57 pm	JetBrains PyCharm Com...	3 KB
directories.json	25/05/2025 11:41 pm	JSON Source File	2 KB
directoriesSetup.json	07/08/2025 11:33 am	JSON Source File	1 KB
reqs.txt	07/07/2025 12:37 pm	Text Document	1 KB
setupDirectories.py	25/05/2025 1:36 pm	JetBrains PyCharm Com...	2 KB

Filename	Description
reqs.txt	Contains a list of Python dependencies required for the project to run. This file can be used with pip install -r reqs.txt to automatically install all necessary libraries.
directoriesSetup.json	Defines relative paths to key project files and directories, organized by function (e.g., logging, data setpreprocessing, dataset labelling). This file is used to initialize absolute paths during project setup.
directories.json	Generated from directoriesSetup.json, this file contains the absolute paths to all critical project components on the local machine. It ensures the application references the correct locations based on the user's environment.
backupClearCollectionFiles.py	Backing up important data directories (excluding specific files like labeller.py). Clearing contents of certain folders used for logging and data labelling. This script helps maintain a clean working environment while preserving historical data.
setupDirectories.py	Initial setup script that: • Converts the relative paths in directoriesSetup.json into absolute paths stored in directories.json. • Updates a configuration file (config.json) used in data preprocessing with correct file paths. • This ensures that all components of the project reference the correct local paths before execution

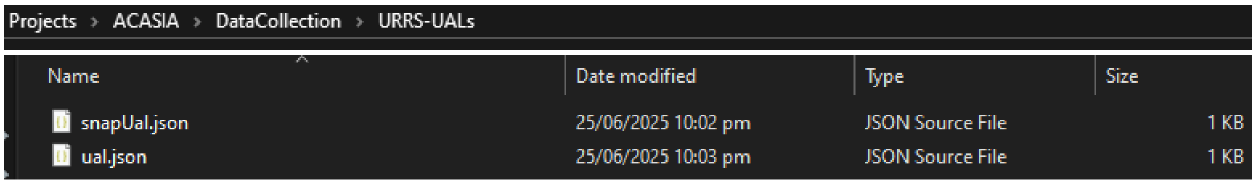
2.1. Data Collection



Projects › ACASIA › DataCollection ›

Name	Date modified	Type	Size
DataCollectionLogs	07/07/2025 11:02 am	File folder	
URRS-UALs	07/07/2025 11:02 am	File folder	
DR.py	20/05/2025 3:00 am	JetBrains PyCharm Com...	5 KB
loggingReset.py	18/05/2025 5:28 pm	JetBrains PyCharm Com...	2 KB
UFAAA.log	25/06/2025 10:03 pm	Text Document	11 KB
UFAAA.py	26/06/2025 11:47 am	JetBrains PyCharm Com...	163 KB
URRS.py	07/08/2025 11:23 am	JetBrains PyCharm Com...	19 KB

Filename	Description
DR.py	Data Record Module – Combines the file access attempts logs with the corresponding physical attributes collected from the snapUal.json (User Room Relocation Simulator Module).
loggingReset.py	Resets key logging-related files to their default or empty states by: • Replacing the main logging.json and breakTracker.json files with their corresponding reset versions. Clearing contents of logs.json and dataRecordLogs.json. This script ensures the data collection environment starts from a clean, default state.
UFAAA.py	User File Access Attempt Automator Module – Generates the file access attempt logs (virtual attributes)
URRS.py	User Room Relocation Simulator Module – Simulates the room relocation of users inside the virtual smart office (physical attributes)

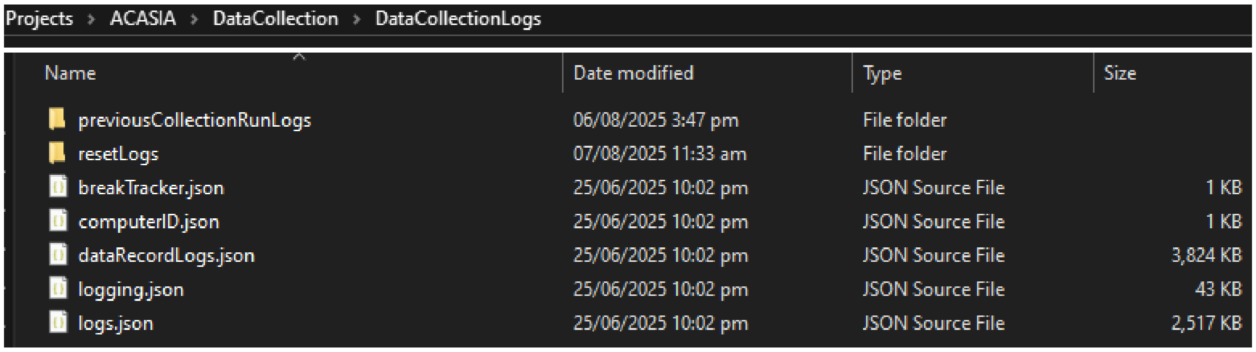


Projects › ACASIA › DataCollection › URRS-UALs

Name	Date modified	Type	Size
snapUal.json	25/06/2025 10:02 pm	JSON Source File	1 KB
ual.json	25/06/2025 10:03 pm	JSON Source File	1 KB

Filename	Description
ual.json	A JSON file containing the room location of each user in the virtual smart office environment.
snapUal.json	A JSON file containing the snapshot or copy (snap shot done by the module) of the room location of each user (ual.json) in the virtual smart office environment which is used to attain the physical attributes of the nearby users (near the computer), computer user location, and computer location when combining

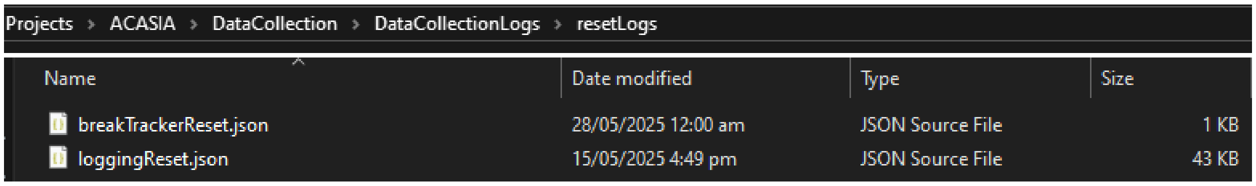
	with the virtual attributes collected from the respective file access attempt log through the Data Record Module.



Projects > ACASIA > DataCollection > DataCollectionLogs

Name	Date modified	Type	Size
previousCollectionRunLogs	06/08/2025 3:47 pm	File folder	
resetLogs	07/08/2025 11:33 am	File folder	
breakTracker.json	25/06/2025 10:02 pm	JSON Source File	1 KB
computerID.json	25/06/2025 10:02 pm	JSON Source File	1 KB
dataRecordLogs.json	25/06/2025 10:02 pm	JSON Source File	3,824 KB
logging.json	25/06/2025 10:02 pm	JSON Source File	43 KB
logs.json	25/06/2025 10:02 pm	JSON Source File	2,517 KB

Filename	Description
breakTracker.json	A JSON file used to track the break status of each user at a given simulated timestamp. Each user is marked as either working normal hours, on break, or outside their shift. This attribute is also recorded with every file access attempt log during data collection.
computerID.json	A JSON file containing the computer ID based on the selected occupied computer room during a simulated file access attempt.
dataRecordLogs.json	A JSON file used to contain the file access attempt logs from the Data Record Module.
logging.json	A JSON file used to keep a categorized tally of how many times each type of file access operation has been generated per user and access type. This helps with debugging and ensuring coverage of different authorized and unauthorized file access attempt scenarios.
logs.json	A JSON file used for storing the simulated file access attempt logs which only contain the virtual attributes during data collection.

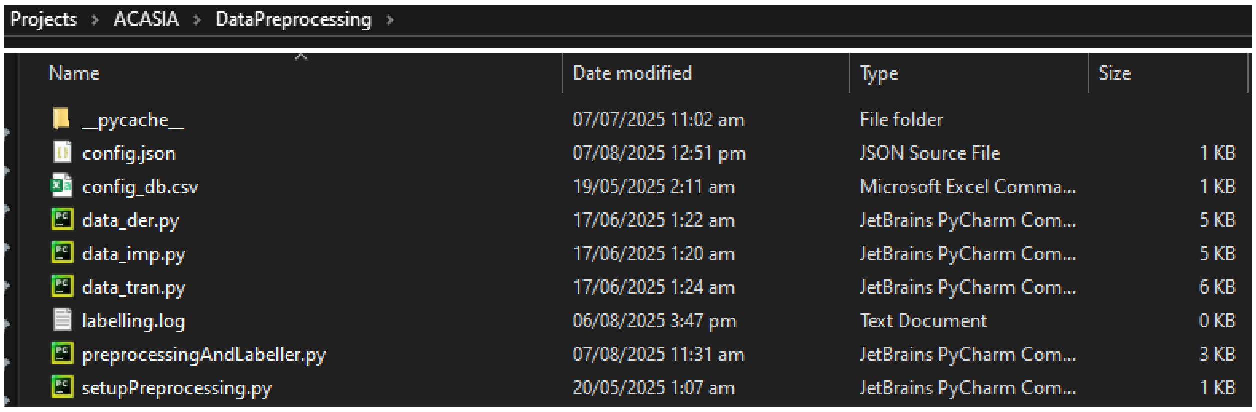


Projects > ACASIA > DataCollection > DataCollectionLogs > resetLogs

Name	Date modified	Type	Size
breakTrackerReset.json	28/05/2025 12:00 am	JSON Source File	1 KB
loggingReset.json	15/05/2025 4:49 pm	JSON Source File	43 KB

Filename	Description
breakTrackerReset.json	A JSON file that contains the initial state of the breakTracker.json in which it replaces the breakTracker.json's contents whenever loggingReset.py is run in order to reset breakTracker.json
loggingReset.json	A JSON file that contains the initial state of the logging.json in which it replaces the logging.json's contents whenever loggingReset.py is run in order to reset logging.json

2.2. Data Preprocessing

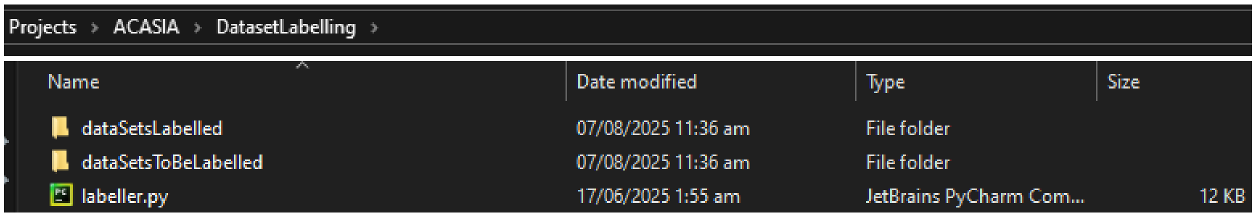

Projects > ACASIA > DataPreprocessing >

Name	Date modified	Type	Size
__pycache__	07/07/2025 11:02 am	File folder	
config.json	07/08/2025 12:51 pm	JSON Source File	1 KB
config_db.csv	19/05/2025 2:11 am	Microsoft Excel Comma...	1 KB
data_der.py	17/06/2025 1:22 am	JetBrains PyCharm Com...	5 KB
data_imp.py	17/06/2025 1:20 am	JetBrains PyCharm Com...	5 KB
data_tran.py	17/06/2025 1:24 am	JetBrains PyCharm Com...	6 KB
labelling.log	06/08/2025 3:47 pm	Text Document	0 KB
preprocessingAndLabeller.py	07/08/2025 11:31 am	JetBrains PyCharm Com...	3 KB
setupPreprocessing.py	20/05/2025 1:07 am	JetBrains PyCharm Com...	1 KB

Filename	Description
config.json	Configuration file used by the data preprocessing pipeline, specifying: • Input/output file paths (e.g., raw, imputed, derived, and transformed datasets) • Logging and tracking files • The path to the main log record JSON used in data collection This file ensures consistent, centralized control over file references during preprocessing operations.
config_db.csv	Defines the user scheduling configuration, including: • User names and roles (e.g., Administrative Staff, Manager, Director) • Corresponding shift start and end times This file supports role-based tracking and analysis during data collection and preprocessing.
data_der.py	Data Derivation Submodule – derives context attributes from the dataset such as nearby users, file type, and file destination type from information within the existing dataset.
data_imp.py	Data Imputation Submodule – removes rows with incomplete values from the dataset.
data_tran.py	Data Transformation Submodule – converts derived categorical and contextual attributes into numerical representations to make the dataset fit for exploratory data analysis.

labelling.log	Logs the amount of authorized and unauthorized file access attempts after labelling the dataset.
preprocessingAndLabeller.py	Preprocessing Module and Automated Labeler Module – Uses the preprocessing submodules (imputation, derivation, transformation) and labels every row of the dataset as authorized or unauthorized.
setupPreprocessing.py	Clears the preprocessing files produced from the preprocessingAndLabeller.py.

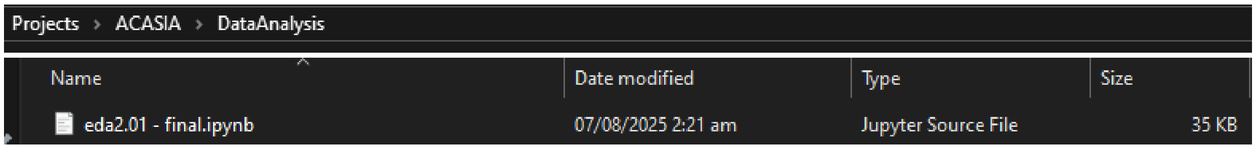
2.3. Data Labelling



```
Name                    Date modified        Type                        Size
dataSetsLabelled        07/08/2025 11:36 am  File folder
dataSetsToBeLabelled    07/08/2025 11:36 am  File folder
labeller.py             17/06/2025 1:55 am   JetBrains PyCharm Com...    12 KB
```

Filename	Description
labeller.py	Executed by the preprocessingAndLabeller.py to label every row of the dataset as authorized or unauthorized.

2.4. Data Analysis

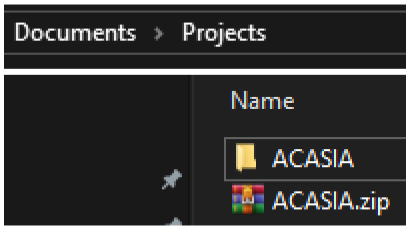


```
Name                  Date modified        Type                 Size
eda2.01 - final.ipynb 07/08/2025 2:21 am   Jupyter Source File  35 KB
```

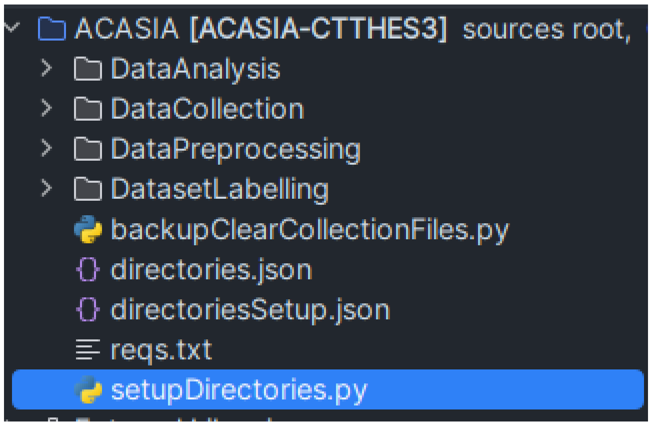
Filename	Description
eda2.01 - final.ipynb	Using the generated data set after the data labelling, this Jupyter Notebook will be run with the dataset to perform exploratory data analysis.

3. Data Processing Pipeline Execution and Data Analysis

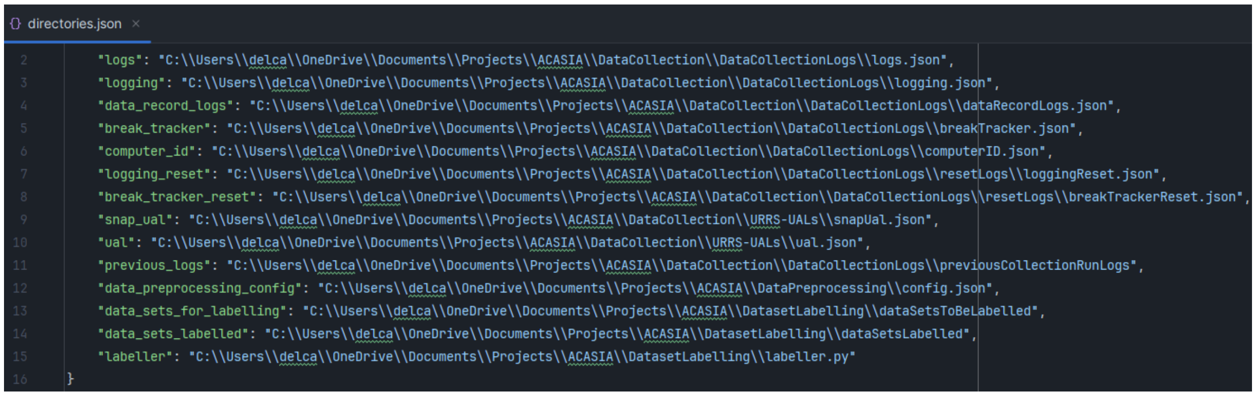
3.1. Project Setup



After extracting (Extract Here) the ACASIA project folder to your desired folder, the program "***setupDirectories.py***" located in the ACASIA main project folder must be run in order for the project to run properly with regards to your desired folder.



As seen below, the proper directories for the log files, dataset folders, scripts, and other files needed for data collection, data preprocessing, and labelling are configured based on the current Windows environment.

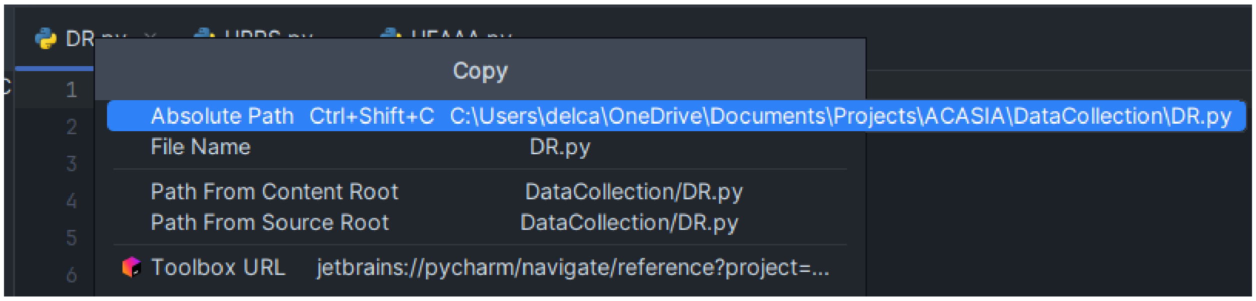


```
    "logs": "C:\\Users\\delca\\OneDrive\\Documents\\Projects\\ACASIA\\DataCollection\\DataCollectionLogs\\logs.json",
    "logging": "C:\\Users\\delca\\OneDrive\\Documents\\Projects\\ACASIA\\DataCollection\\DataCollectionLogs\\logging.json",
    "data_record_logs": "C:\\Users\\delca\\OneDrive\\Documents\\Projects\\ACASIA\\DataCollection\\DataCollectionLogs\\dataRecordLogs.json",
    "break_tracker": "C:\\Users\\delca\\OneDrive\\Documents\\Projects\\ACASIA\\DataCollection\\DataCollectionLogs\\breakTracker.json",
    "computer_id": "C:\\Users\\delca\\OneDrive\\Documents\\Projects\\ACASIA\\DataCollection\\DataCollectionLogs\\computerID.json",
    "logging_reset": "C:\\Users\\delca\\OneDrive\\Documents\\Projects\\ACASIA\\DataCollection\\DataCollectionLogs\\resetLogs\\loggingReset.json",
    "break_tracker_reset": "C:\\Users\\delca\\OneDrive\\Documents\\Projects\\ACASIA\\DataCollection\\DataCollectionLogs\\resetLogs\\breakTrackerReset.json",
    "snap_ual": "C:\\Users\\delca\\OneDrive\\Documents\\Projects\\ACASIA\\DataCollection\\URRS-UALs\\snapUal.json",
    "ual": "C:\\Users\\delca\\OneDrive\\Documents\\Projects\\ACASIA\\DataCollection\\URRS-UALs\\ual.json",
    "previous_logs": "C:\\Users\\delca\\OneDrive\\Documents\\Projects\\ACASIA\\DataCollection\\DataCollectionLogs\\previousCollectionRunLogs",
    "data_preprocessing_config": "C:\\Users\\delca\\OneDrive\\Documents\\Projects\\ACASIA\\DataPreprocessing\\config.json",
    "data_sets_for_labelling": "C:\\Users\\delca\\OneDrive\\Documents\\Projects\\ACASIA\\DatasetLabelling\\dataSetsToBeLabelled",
    "data_sets_labelled": "C:\\Users\\delca\\OneDrive\\Documents\\Projects\\ACASIA\\DatasetLabelling\\dataSetsLabelled",
    "labeller": "C:\\Users\\delca\\OneDrive\\Documents\\Projects\\ACASIA\\DatasetLabelling\\labeller.py"
}
```

Open three command prompts to prepare to run the three Python scripts of "***DR.py***" (Data Record Module), "***URRS.py***" (User Room Relocation Simulator Module), and "***UFAAA.py***" (User File Access Attempt Automator Module).

3.2. Data Collection

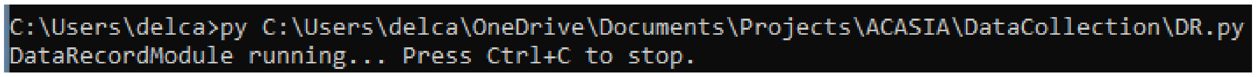
With the three Python scripts open in PyCharm, copy their absolute path and run in each of the respective command prompts



```
Absolute Path  Ctrl+Shift+C  C:\Users\delca\OneDrive\Documents\Projects\ACASIA\DataCollection\DR.py
File Name                    DR.py
Path From Content Root       DataCollection/DR.py
Path From Source Root        DataCollection/DR.py
Toolbox URL    jetbrains://pycharm/navigate/reference?project=...
```

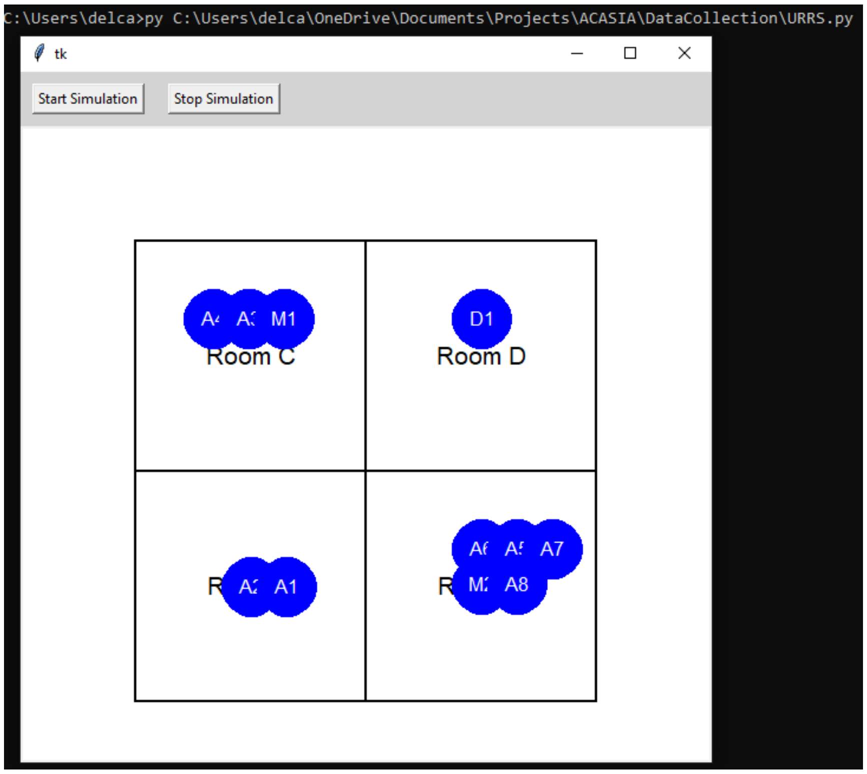
Then run in each of the command prompts with '***py "<absolute path>"***'.

Data Record Module (DR) should display this output.



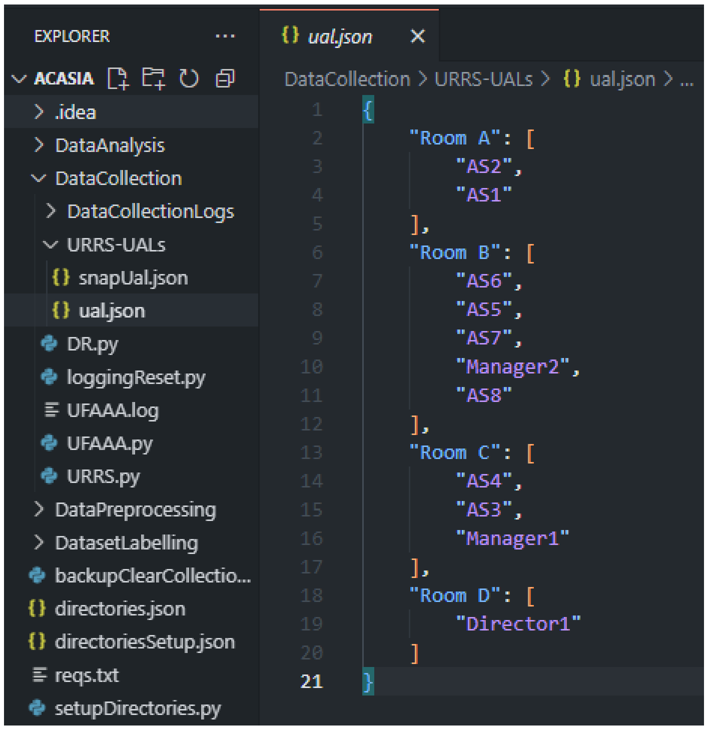
```
C:\Users\delca>py C:\Users\delca\OneDrive\Documents\Projects\ACASIA\DataCollection\DR.py
DataRecordModule running... Press Ctrl+C to stop.
```

User Room Relocation Simulator Module (URRS) should display this output.



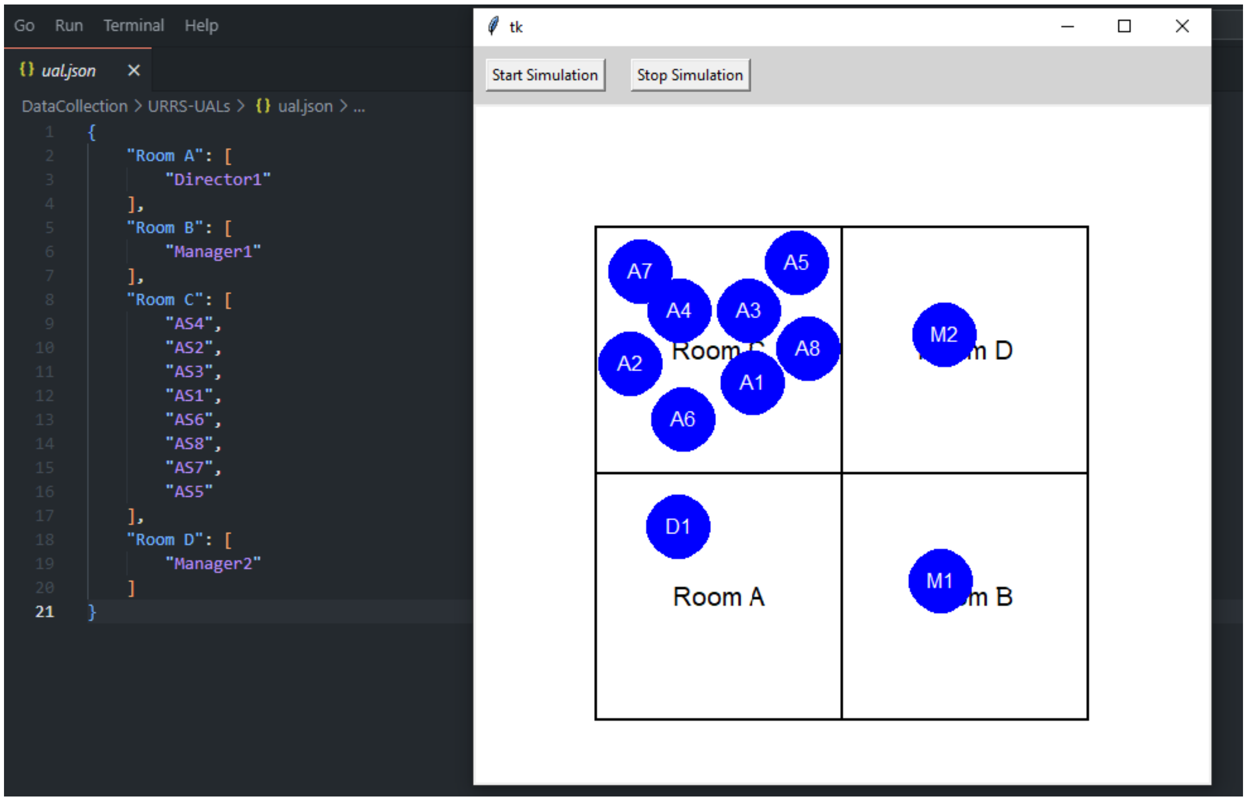
```
C:\Users\delca>py C:\Users\delca\OneDrive\Documents\Projects\ACASIA\DataCollection\URRS.py
```

To confirm if URRS is working properly, open the project in VS Code, go to the directory "***DataCollection\URRS-UALs***" and open "***ual.json***" and click on "***Start Simulation***" in the GUI and compare the GUI and "***ual.json***" by checking both of their movements to confirm it is working correctly. You can also check by just moving a user node instead to another room to confirm.



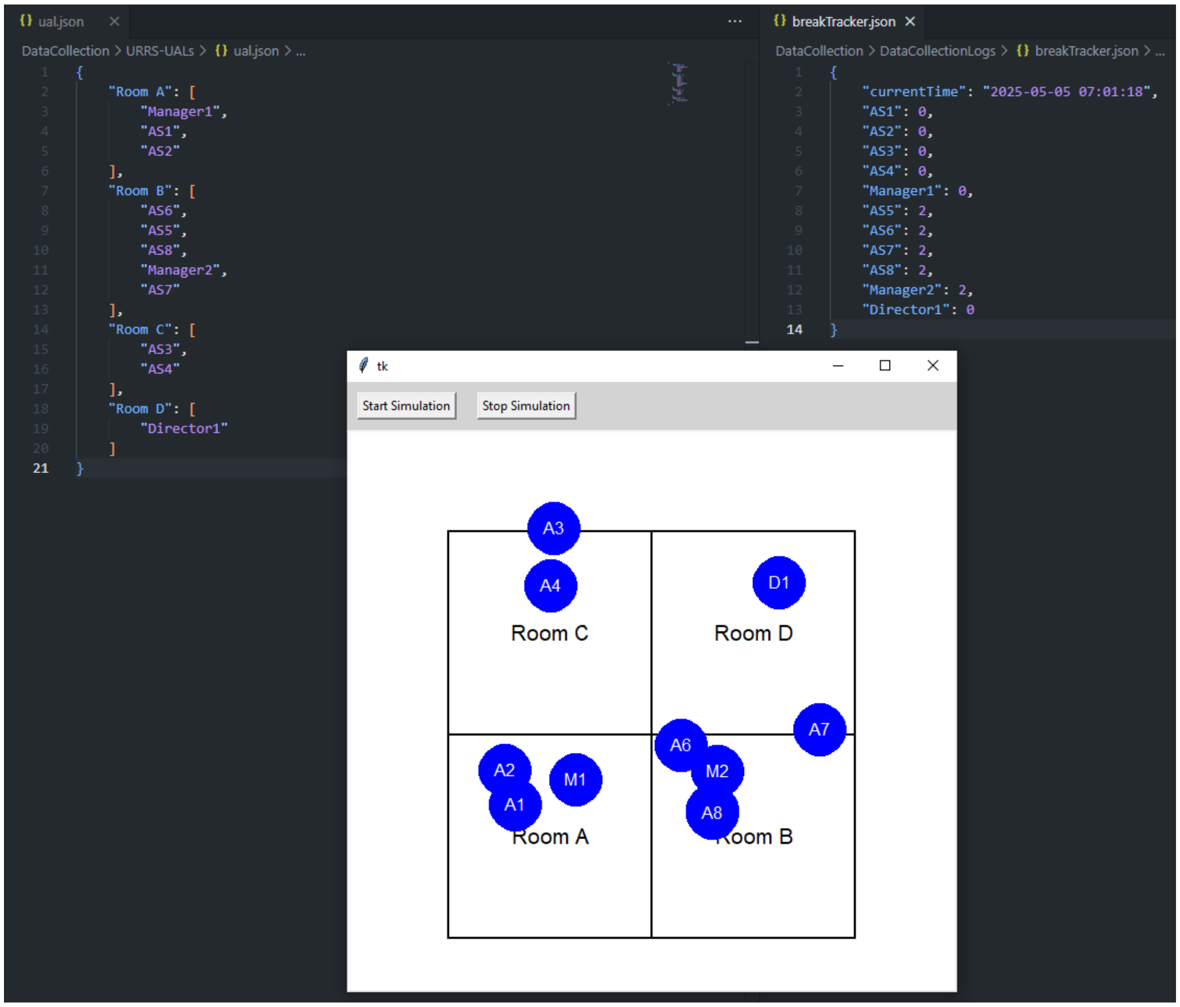
```
{
    "Room A": [
        "AS2",
        "AS1"
    ],
    "Room B": [
        "AS6",
        "AS5",
        "AS7",
        "Manager2",
        "AS8"
    ],
    "Room C": [
        "AS4",
        "AS3",
        "Manager1"
    ],
    "Room D": [
        "Director1"
    ]
}
```

Manual Check:



```
{
    "Room A": [
        "Director1"
    ],
    "Room B": [
        "Manager1"
    ],
    "Room C": [
        "AS4",
        "AS2",
        "AS3",
        "AS1",
        "AS6",
        "AS8",
        "AS7",
        "AS5"
    ],
    "Room D": [
        "Manager2"
    ]
}
```

Checking depending on break status (0 [normal working hours], 1 [on break], 2 [out of shift]:

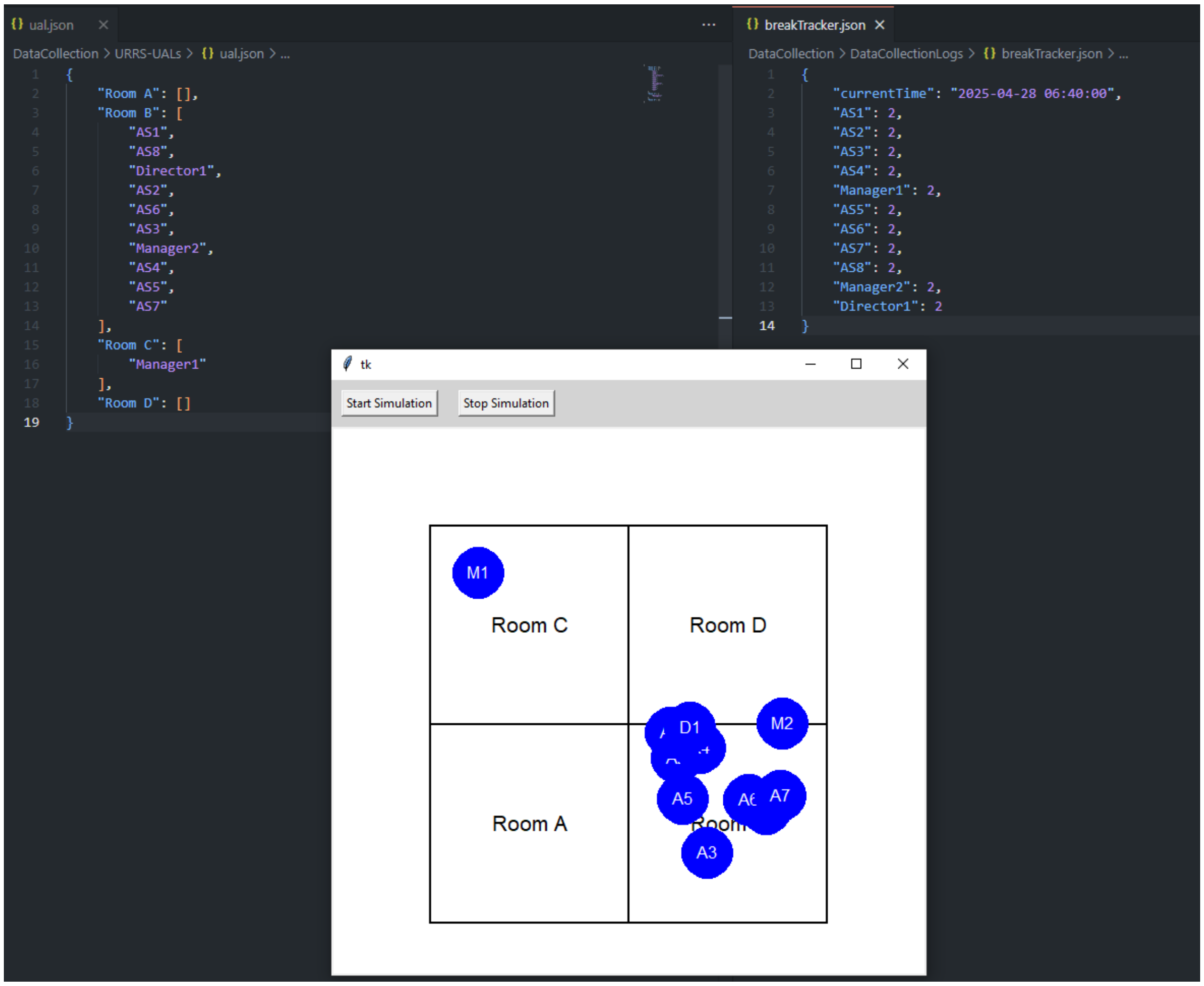


```
{
    "Room A": [
        "Manager1",
        "AS1",
        "AS2"
    ],
    "Room B": [
        "AS6",
        "AS5",
        "AS8",
        "Manager2",
        "AS7"
    ],
    "Room C": [
        "AS3",
        "AS4"
    ],
    "Room D": [
        "Director1"
    ]
}
```

```
{
    "currentTime": "2025-05-05 07:01:18",
    "AS1": 0,
    "AS2": 0,
    "AS3": 0,
    "AS4": 0,
    "Manager1": 0,
    "AS5": 2,
    "AS6": 2,
    "AS7": 2,
    "AS8": 2,
    "Manager2": 2,
    "Director1": 0
}
```

ual.json
DataCollection > URRS-UALs > ual.json > ...
breakTracker.json
DataCollection > DataCollectionLogs > breakTracker.json > ...
tk
Start Simulation
Stop Simulation
Room C
Room D
Room A
Room B
A3
A4
D1
A7
A6
M2
A8
A2
M1
A1

By default since the "***currentTime***" field will start at 6:40 AM, out of shift access of users in the office will have low probabilities.

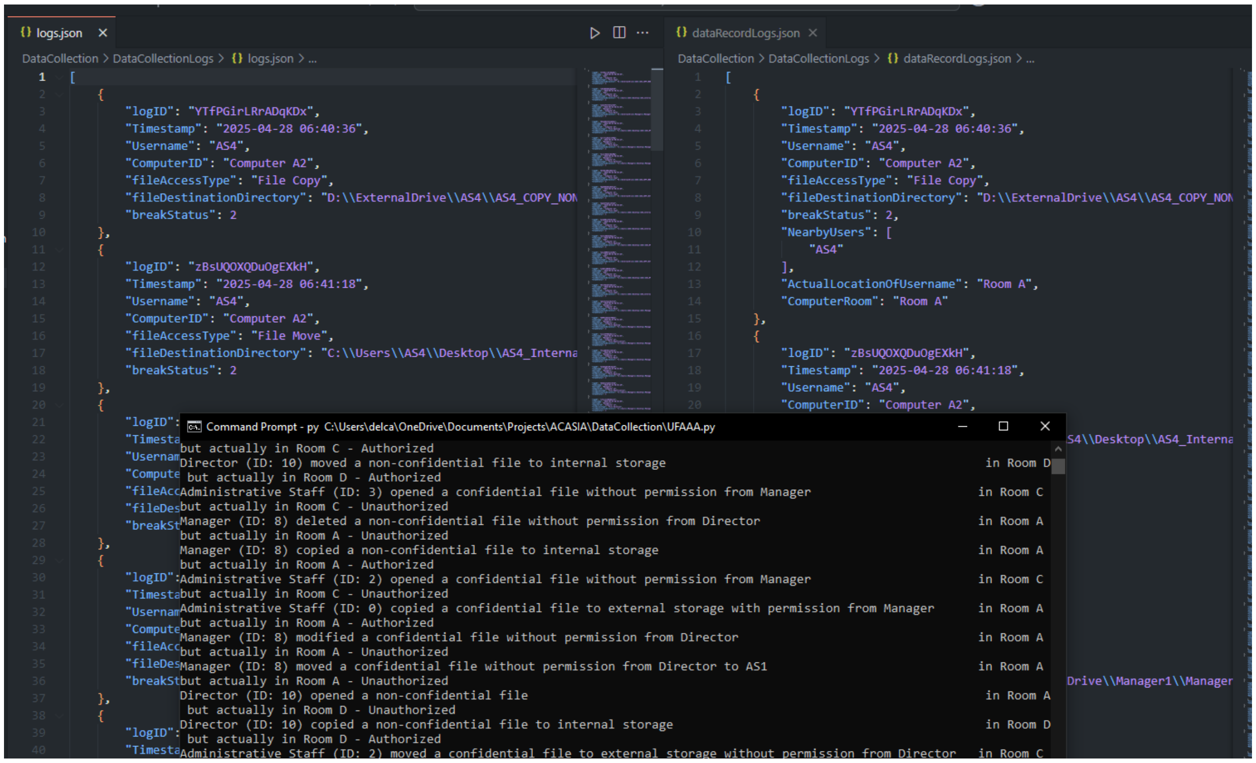


```
{
    "Room A": [],
    "Room B": [
        "AS1",
        "AS8",
        "Director1",
        "AS2",
        "AS6",
        "AS3",
        "Manager2",
        "AS4",
        "AS5",
        "AS7"
    ],
    "Room C": [
        "Manager1"
    ],
    "Room D": []
}
```

```
{
    "currentTime": "2025-04-28 06:40:00",
    "AS1": 2,
    "AS2": 2,
    "AS3": 2,
    "AS4": 2,
    "Manager1": 2,
    "AS5": 2,
    "AS6": 2,
    "AS7": 2,
    "AS8": 2,
    "Manager2": 2,
    "Director1": 2
}
```

ual.json
DataCollection > URRS-UALs > ual.json > ...
breakTracker.json
DataCollection > DataCollectionLogs > breakTracker.json > ...
tk
Start Simulation
Stop Simulation
M1
Room C
Room D
D1
M2
A5
A7
Room A
A3

User File Access Attempt Automator Module (UFAAA) should display this output (while checking the "***DataCollection\DataCollectionLogs\dataRecordLogs.json***" (also "***logs.json***") for correct outputs of the file access attempt logs. It can be seen that physical attributes (from "***DataCollection\URRS-UALs\snapUal.json***") are properly being combined below where the nearby users, actual location of user, and computer room are being appended to the respective logs in "***dataRecordLogs.json***".



logs.json

DataCollection > DataCollectionLogs > logs.json > ...

```
[
    {
        "logID": "YTfPGirLRrADqKDx",
        "Timestamp": "2025-04-28 06:40:36",
        "Username": "AS4",
        "ComputerID": "Computer A2",
        "fileAccessType": "File Copy",
        "fileDestinationDirectory": "D:\\ExternalDrive\\AS4\\AS4_COPY_NON
        "breakStatus": 2
    },
    {
        "logID": "zBsUQOXQDuOgEXkH",
        "Timestamp": "2025-04-28 06:41:18",
        "Username": "AS4",
        "ComputerID": "Computer A2",
        "fileAccessType": "File Move",
        "fileDestinationDirectory": "C:\\Users\\AS4\\Desktop\\AS4_Interna
        "breakStatus": 2
    },
    {
```

dataRecordLogs.json

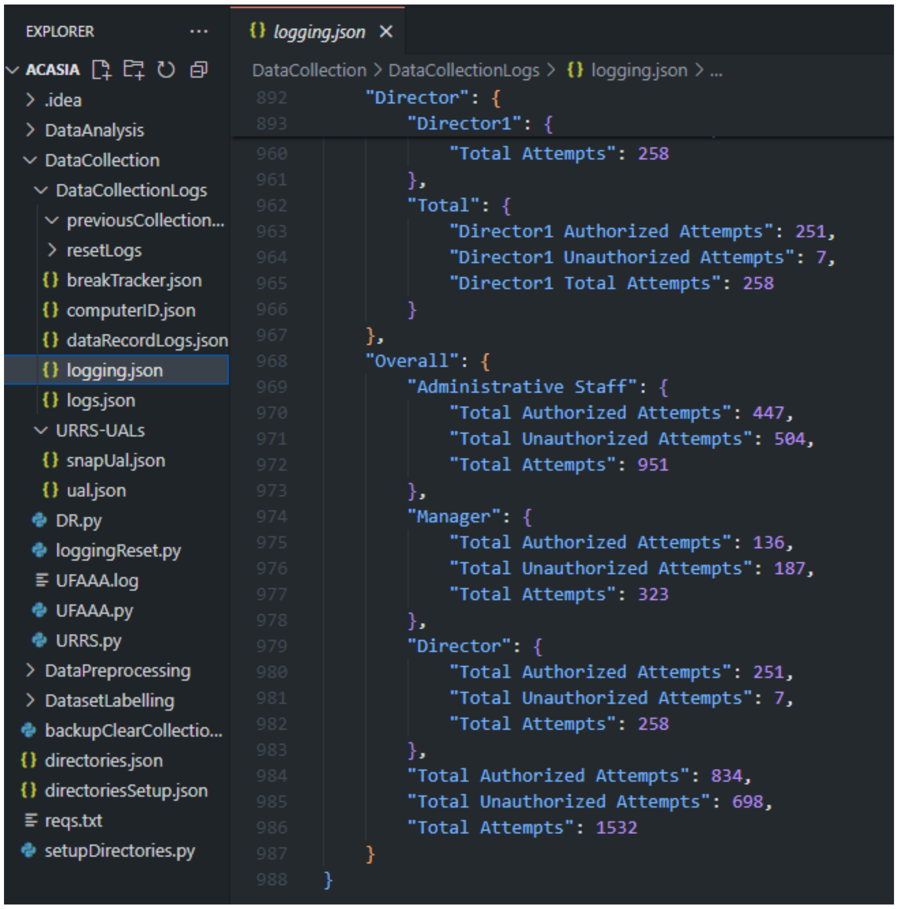
DataCollection > DataCollectionLogs > dataRecordLogs.json > ...

```
[
    {
        "logID": "YTfPGirLRrADqKDx",
        "Timestamp": "2025-04-28 06:40:36",
        "Username": "AS4",
        "ComputerID": "Computer A2",
        "fileAccessType": "File Copy",
        "fileDestinationDirectory": "D:\\ExternalDrive\\AS4\\AS4_COPY_NON
        "breakStatus": 2,
        "NearbyUsers": [
            "AS4"
        ],
        "ActualLocationOfUsername": "Room A",
        "ComputerRoom": "Room A"
    },
    {
        "logID": "zBsUQOXQDuOgEXkH",
        "Timestamp": "2025-04-28 06:41:18",
        "Username": "AS4",
        "ComputerID": "Computer A2",
```

Command Prompt - py C:\Users\delca\OneDrive\Documents\Projects\ACASIA\DataCollection\UFAAA.py

```
but actually in Room C - Authorized
Director (ID: 10) moved a non-confidential file to internal storage                                in Room D
 but actually in Room D - Authorized
Administrative Staff (ID: 3) opened a confidential file without permission from Manager           in Room C
but actually in Room C - Unauthorized
Manager (ID: 8) deleted a non-confidential file without permission from Director                  in Room A
but actually in Room A - Unauthorized
Manager (ID: 8) copied a non-confidential file to internal storage                                in Room A
but actually in Room A - Authorized
Administrative Staff (ID: 2) opened a confidential file without permission from Manager           in Room C
but actually in Room C - Unauthorized
Administrative Staff (ID: 0) copied a confidential file to external storage with permission from Manager      in Room A
but actually in Room A - Authorized
Manager (ID: 8) modified a confidential file without permission from Director                     in Room A
but actually in Room A - Unauthorized
Manager (ID: 8) moved a confidential file without permission from Director to AS1                 in Room A
but actually in Room A - Unauthorized
Director (ID: 10) opened a non-confidential file                                                   in Room A
 but actually in Room D - Unauthorized
Director (ID: 10) copied a non-confidential file to internal storage                               in Room D
 but actually in Room D - Authorized
Administrative Staff (ID: 2) moved a confidential file to external storage without permission from Director   in Room C
```

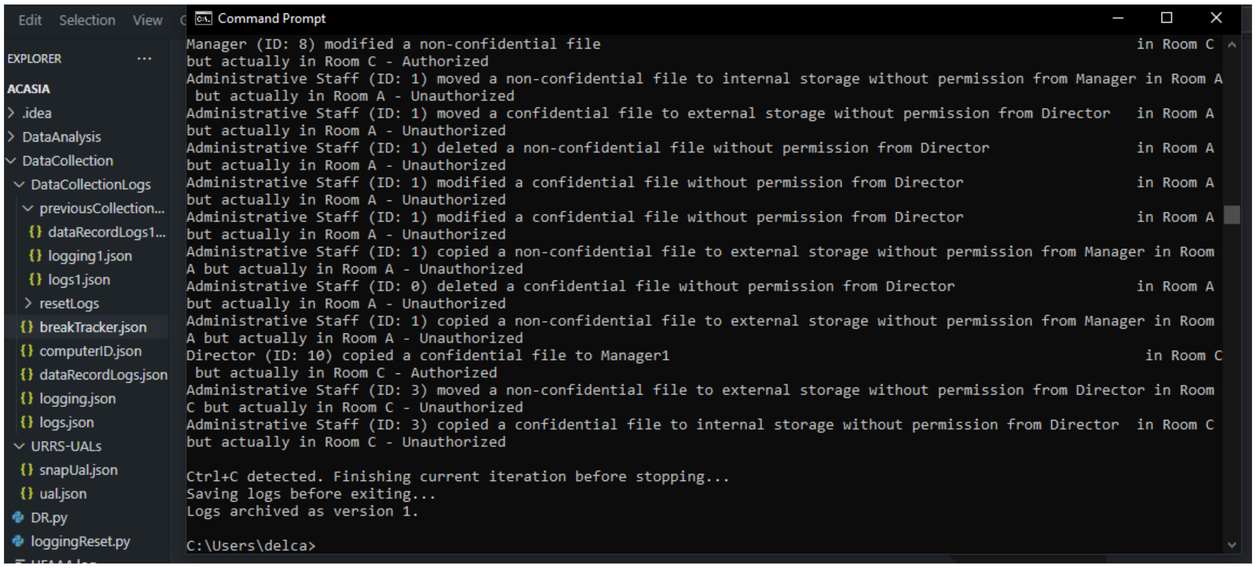
"***DataCollection\logging.json***" can also be checked to see how many logs were tallied per user role, how many authorized/unauthorized file access attempts, how many counts of each type of file access attempts were done per user.



```
    "Director": {
        "Director1": {
            "Total Attempts": 258
        },
        "Total": {
            "Director1 Authorized Attempts": 251,
            "Director1 Unauthorized Attempts": 7,
            "Director1 Total Attempts": 258
        }
    },
    "Overall": {
        "Administrative Staff": {
            "Total Authorized Attempts": 447,
            "Total Unauthorized Attempts": 504,
            "Total Attempts": 951
        },
        "Manager": {
            "Total Authorized Attempts": 136,
            "Total Unauthorized Attempts": 187,
            "Total Attempts": 323
        },
        "Director": {
            "Total Authorized Attempts": 251,
            "Total Unauthorized Attempts": 7,
            "Total Attempts": 258
        },
        "Total Authorized Attempts": 834,
        "Total Unauthorized Attempts": 698,
        "Total Attempts": 1532
    }
}
```

Once the three Python scripts are running, these can be left running in the background for hours. It is expected that collection (rate of logs) will slow down after a few hours. The UFAAA can be paused by just typing "***Ctrl C***" and wait for it to fully end as it is going to back up the previous logs in the "***DataCollection\previousCollectionRunLogs***" directory and by running it again with '***py "<absolute path of UFAAA>"***', it will continue from the same "***currentTime***" field and respective break statuses. It is however suggested to clear "***logs.json***" before re-running the UFAAA, to make the speed of collection similar to how the three scripts started. DR and URRS do not need to be re-run again since they do not take up that much memory as they are lighter programs compared to UFAAA.

Once finished with collection, and as stated before, typing "***Ctrl C***" in the UFAAA will end the collection and saved its logs to the "***DataCollection\previousCollectionRunLogs***" directory where in each log filename will be appended with the iteration count of the runs of the UFAAA.

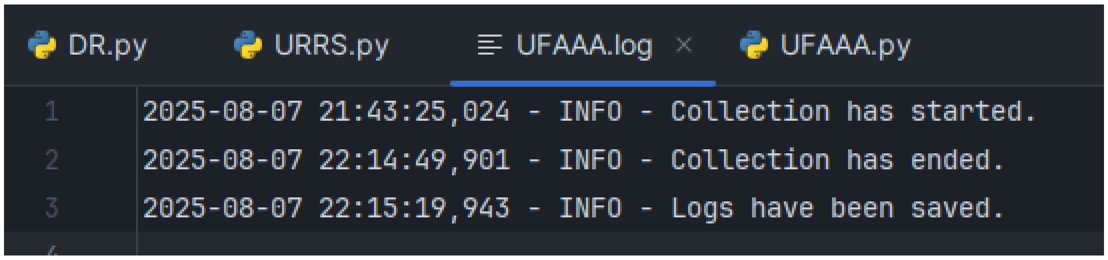


```
Manager (ID: 8) modified a non-confidential file                                          in Room C
but actually in Room C - Authorized
Administrative Staff (ID: 1) moved a non-confidential file to internal storage without permission from Manager in Room A
 but actually in Room A - Unauthorized
Administrative Staff (ID: 1) moved a confidential file to external storage without permission from Director   in Room A
but actually in Room A - Unauthorized
Administrative Staff (ID: 1) deleted a non-confidential file without permission from Director             in Room A
but actually in Room A - Unauthorized
Administrative Staff (ID: 1) modified a confidential file without permission from Director                in Room A
but actually in Room A - Unauthorized
Administrative Staff (ID: 1) modified a confidential file without permission from Director                in Room A
but actually in Room A - Unauthorized
Administrative Staff (ID: 1) copied a non-confidential file to external storage without permission from Manager in Room
A but actually in Room A - Unauthorized
Administrative Staff (ID: 0) deleted a confidential file without permission from Director                 in Room A
but actually in Room A - Unauthorized
Administrative Staff (ID: 1) copied a non-confidential file to external storage without permission from Manager in Room
A but actually in Room A - Unauthorized
Director (ID: 10) copied a confidential file to Manager1                                                   in Room C
 but actually in Room C - Authorized
Administrative Staff (ID: 3) moved a non-confidential file to external storage without permission from Director in Room
C but actually in Room C - Unauthorized
Administrative Staff (ID: 3) copied a confidential file to internal storage without permission from Director  in Room C
but actually in Room C - Unauthorized

Ctrl+C detected. Finishing current iteration before stopping...
Saving logs before exiting...
Logs archived as version 1.

C:\Users\delca>
```

It can also be seen in “***DataCollection\UFAAA.log***” the time the collection has started and ended and when the logs have been saved.

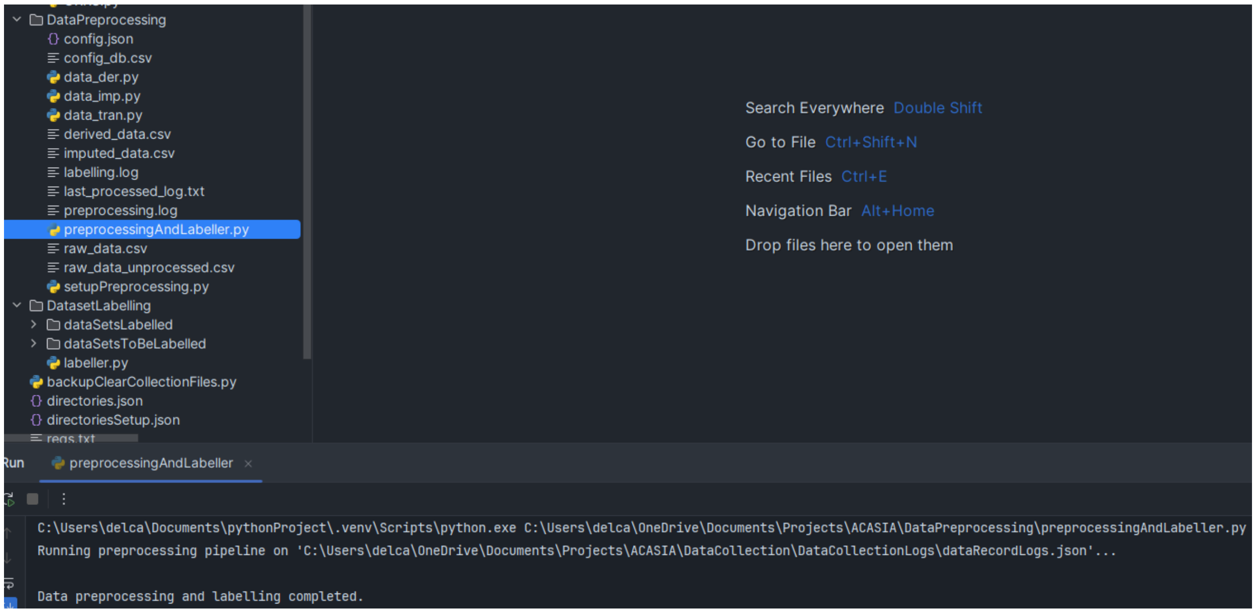


```
2025-08-07 21:43:25,024 - INFO - Collection has started.
2025-08-07 22:14:49,901 - INFO - Collection has ended.
2025-08-07 22:15:19,943 - INFO - Logs have been saved.
```

After the collection, data preprocessing and data labelling can already be done to prepare the data collected for exploratory data analysis.

3.3. Data Preprocessing

To perform data preprocessing and labelling the “***Data Preprocessing\preprocessingAndLabeller.py***” can be run in PyCharm.

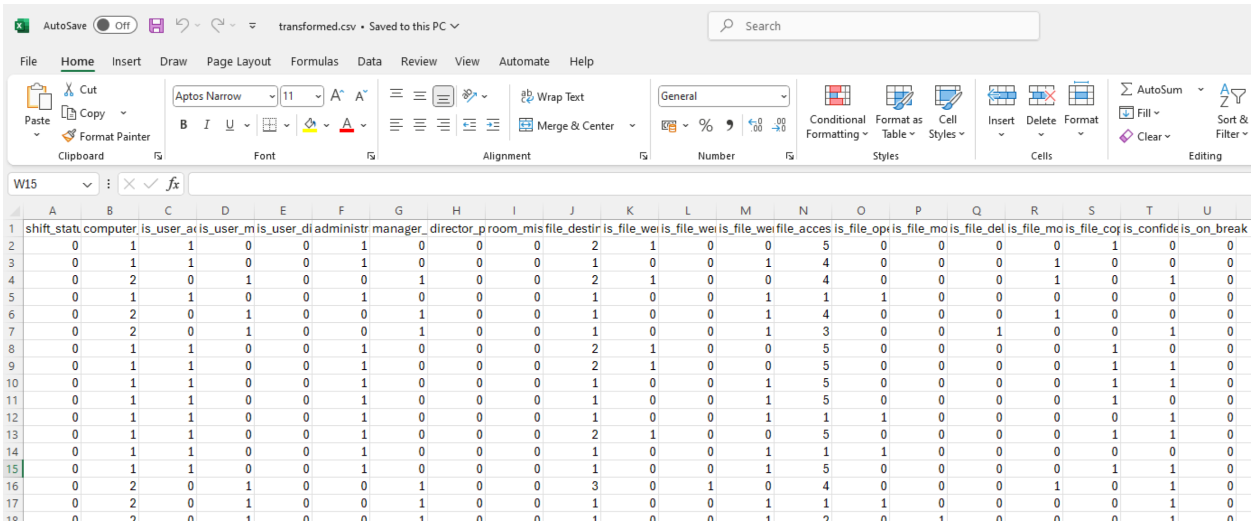


```
DataPreprocessing
    config.json
    config_db.csv
    data_der.py
    data_imp.py
    data_tran.py
    derived_data.csv
    imputed_data.csv
    labelling.log
    last_processed_log.txt
    preprocessing.log
    preprocessingAndLabeller.py
    raw_data.csv
    raw_data_unprocessed.csv
    setupPreprocessing.py
DatasetLabelling
    dataSetsLabelled
    dataSetsToBeLabelled
    labeller.py
backupClearCollectionFiles.py
directories.json
directoriesSetup.json
```

```
C:\Users\delca\Documents\pythonProject\.venv\Scripts\python.exe C:\Users\delca\OneDrive\Documents\Projects\ACASIA\DataPreprocessing\preprocessingAndLabeller.py
Running preprocessing pipeline on 'C:\Users\delca\OneDrive\Documents\Projects\ACASIA\DataCollection\DataCollectionLogs\dataRecordLogs.json'...

Data preprocessing and labelling completed.
```

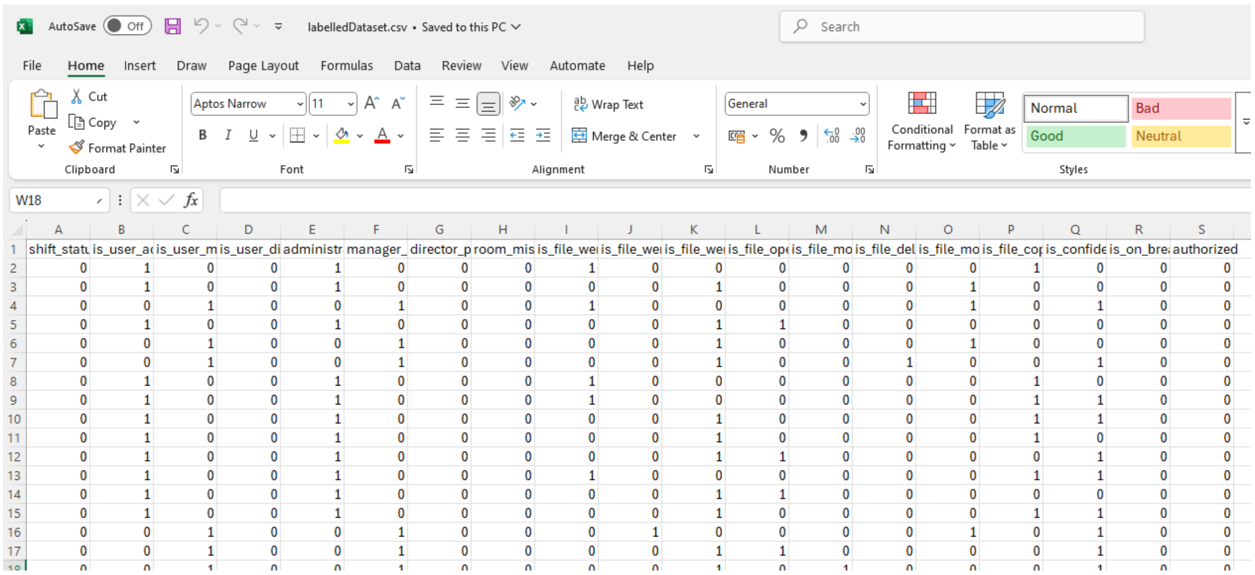
The transformed dataset (which is the last step of the preprocessing [data transformation]) will be appear in “***DatasetLabelling\dataSetsToBeLabelled\transformed.csv***”.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	shift_statu	computer_	is_user_ac	is_user_m	is_user_di	administr	manager_	director_p	room_mis	file_destir	is_file_wei	is_file_wei	is_file_wei	file_acces	is_file_ope	is_file_mo	is_file_del	is_file_mo	is_file_cop	is_confide	is_on_break
2	0	1	1	0	0	1	0	0	0	2	1	0	0	5	0	0	0	0	1	0	0
3	0	1	1	0	0	1	0	0	0	1	0	0	1	4	0	0	0	1	0	0	0
4	0	2	0	1	0	0	1	0	0	2	1	0	0	4	0	0	0	1	0	1	0
5	0	1	1	0	0	1	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0
6	0	2	0	1	0	0	1	0	0	1	0	0	1	4	0	0	0	1	0	0	0
7	0	2	0	1	0	0	1	0	0	1	0	0	1	3	0	0	1	0	0	1	0
8	0	1	1	0	0	1	0	0	0	2	1	0	0	5	0	0	0	0	1	0	0
9	0	1	1	0	0	1	0	0	0	2	1	0	0	5	0	0	0	0	1	1	0
10	0	1	1	0	0	1	0	0	0	1	0	0	1	5	0	0	0	0	1	1	0
11	0	1	1	0	0	1	0	0	0	1	0	0	1	5	0	0	0	0	1	0	0
12	0	1	1	0	0	1	0	0	0	1	0	0	1	1	1	0	0	0	0	1	0
13	0	1	1	0	0	1	0	0	0	2	1	0	0	5	0	0	0	0	1	1	0
14	0	1	1	0	0	1	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0
15	0	1	1	0	0	1	0	0	0	1	0	0	1	5	0	0	0	0	1	1	0
16	0	2	0	1	0	0	1	0	0	3	0	1	0	4	0	0	0	1	0	1	0
17	0	2	0	1	0	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0

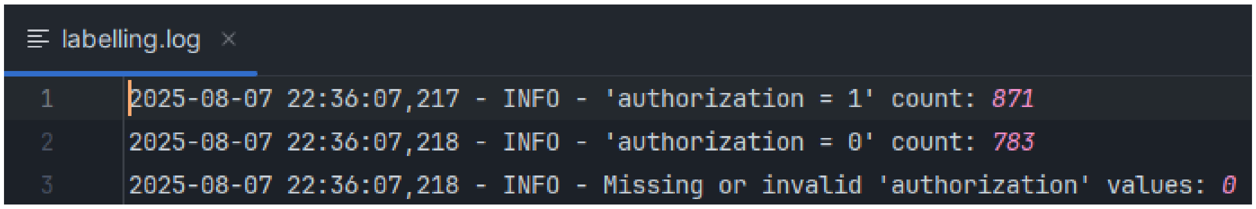
3.4. Data Labelling

As the data preprocessing and labelling are run at the same program, the labelled dataset will appear in “***DatasetLabelling\dataSetsLabelled\labelledDataset.csv***” where in it would have an “***authorized***” field. A value of 1 will indicate that it is an authorized file access attempt while a value of 0 will indicate that that is an unauthorized file access attempt based on the context attributes such as “***shift_status***”, “***room_mismatch***”, “***is_confidential***” and other attributes which will be used in the labelling rules.


AutoSave Off
labelledDataset.csv • Saved to this PC
Search
File Home Insert Draw Page Layout Formulas Data Review View Automate Help
Cut Copy Format Painter Paste Clipboard
Aptos Narrow 11 Font
Wrap Text Merge & Center Alignment
General Number
Conditional Formatting Format as Table
Normal Bad Good Neutral Styles
W18

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	shift_statu	is_user_ac	is_user_m	is_user_di	administr	manager_	director_p	room_mis	is_file_we	is_file_we	is_file_we	is_file_ope	is_file_mo	is_file_del	is_file_mo	is_file_cop	is_confide	is_on_bre	authorized
2	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
3	0	1	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0
4	0	0	1	0	0	1	0	0	1	0	0	0	0	0	1	0	1	0	0
5	0	1	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0
6	0	0	1	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0
7	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0	1	0	0
8	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	1	1	0	0
10	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	1	1	0	0
11	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0
12	0	1	0	0	1	0	0	0	0	0	1	1	0	0	0	0	1	0	0
13	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	1	1	0	0
14	0	1	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0
15	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	1	1	0	0
16	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	1	0	0
17	0	0	1	0	0	1	0	0	0	0	1	1	0	0	0	0	1	0	0
18	0	0	1	0	0	1	0	0	0	0	1	0	1	0	0	0	1	0	0

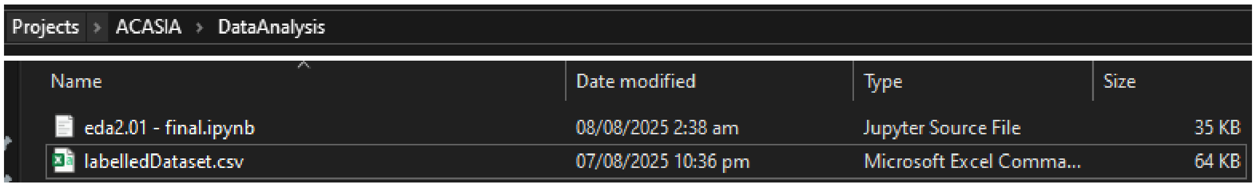
For the counts of authorized and unauthorized file access attempts it can be seen in "***DataPreprocessing\labelling.log***"



```
2025-08-07 22:36:07,217 - INFO - 'authorization = 1' count: 871
2025-08-07 22:36:07,218 - INFO - 'authorization = 0' count: 783
2025-08-07 22:36:07,218 - INFO - Missing or invalid 'authorization' values: 0
```

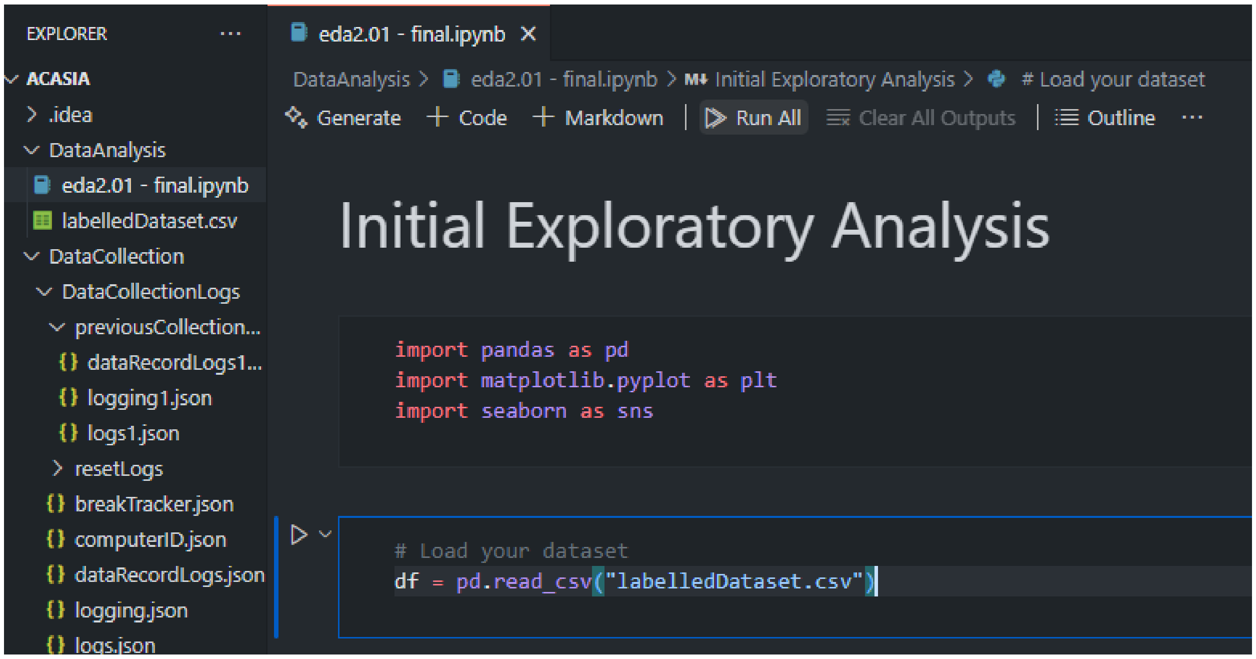
3.5. Data Analysis

After the dataset has been labelled, transfer or copy the "***labelledDataset.csv***" to the "***DataAnalysis***" folder.



Name	Date modified	Type	Size
eda2.01 - final.ipynb	08/08/2025 2:38 am	Jupyter Source File	35 KB
labelledDataset.csv	07/08/2025 10:36 pm	Microsoft Excel Comma...	64 KB

Open the "***eda2.01 - final.ipynb***" file in VS Code then click on "***Run All***"

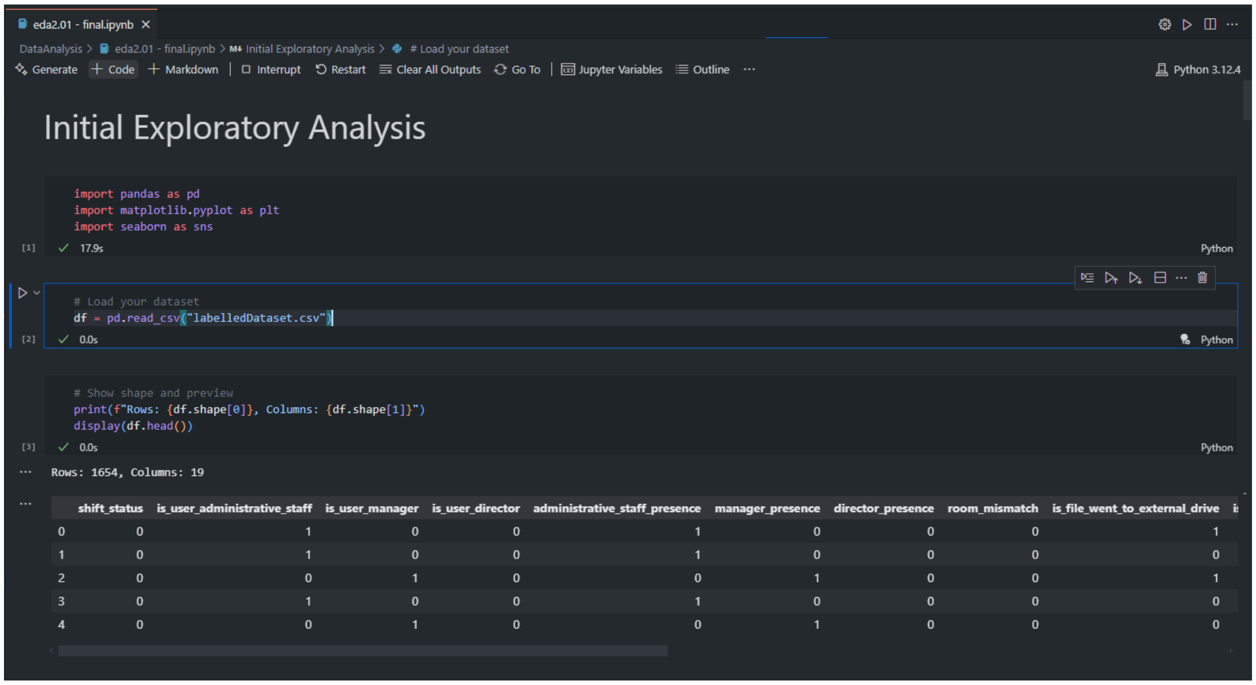


Initial Exploratory Analysis

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Load your dataset
df = pd.read_csv("labelledDataset.csv")
```

The Jupyter Notebook will then run to perform explanatory data analysis on the dataset produced from data collection which was then preprocessed and labelled.



Initial Exploratory Analysis

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[1] ✓ 17.9s Python

```
# Load your dataset
df = pd.read_csv("labelledDataset.csv")
```

[2] ✓ 0.0s Python

```
# Show shape and preview
print(f"Rows: {df.shape[0]}, Columns: {df.shape[1]}")
display(df.head())
```

[3] ✓ 0.0s Python

```
Rows: 1654, Columns: 19
```

	shift_status	is_user_administrative_staff	is_user_manager	is_user_director	administrative_staff_presence	manager_presence	director_presence	room_mismatch	is_file_went_to_external_drive
0	0	1	0	0	1	0	0	0	1
1	0	1	0	0	1	0	0	0	0
2	0	0	1	0	0	1	0	0	1
3	0	1	0	0	1	0	0	0	0
4	0	0	1	0	0	1	0	0	0