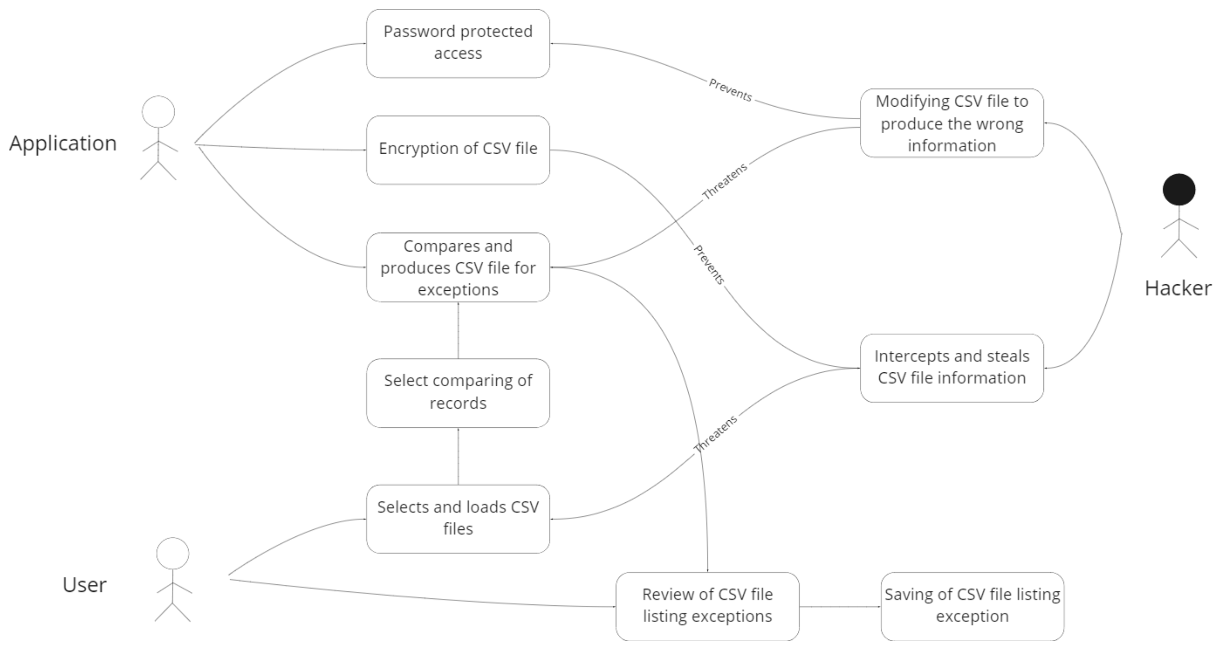
Use Case Diagram


Application
Password protected access
Encryption of CSV file
Compares and produces CSV file for exceptions
Select comparing of records
Selects and loads CSV files
User
Review of CSV file listing exceptions
Saving of CSV file listing exception
Prevents
Threatens
Prevents
Threatens
Modifying CSV file to produce the wrong information
Intercepts and steals CSV file information
Hacker

Equivalence Class & Boundary Values

1. **Input files into the software**

   a. When the user runs the software with input files, the file types are of to be .csv
   b. The file is also expected to be of the user's working directory
      - If the file is not in the user's working directory, an exception for file not found should be thrown

Below lists the equivalence class of file types that should be accepted

Test Scenario #	Test Scenario Description	Expected Outcome
1	Both files are of .csv type	System should accept
2	Only one file is of .csv type	System should not accept
3	Both files are not of .csv type	System should not accept

Boundary cases: Minor typo in the file type for example, .tsv

2. **Comparison of values in the CSV files**

   a. The software should detect all differences in the csv files, inclusive of significant figures and capitalizations
   b. One mismatched value should additionally flag the entire row of the data

Below lists the equivalence class of values that will be flagged

Test Scenario #	Test Scenario Description	Expected Outcome

1	File 1 row: {id: 1, balance: 1, …}
	File 2 row: {id: 1, balance: 1, …}
		System should return no mismatches

2	File 1 row: {id: 1, balance: 200, …}
	File 2 row: {id: 1, balance: 300, …}
		System should return a mismatch in the row for id 1 as the balances are different

3	File 1 row: {id: 1, balance: 200, …}
	File 2 row: {id: 1, balance: 200.00, …}
		System should return a mismatch in the row for id 1 as the balances significant figures are different

4	File 1 row: {id: 1, type: SAVINGS, …}
	File 2 row: {id: 1, type: savings, …}
		System should return a mismatch as capitalization is also considered during matching

Boundary cases: Boundary cases includes significant figures and capitalizations that should be considered when calculating for edge cases as the software is meant to compare exact matches in the rows

3. Comparison of order of headers and values

a. The order in which the headers and values are listed will be considered
b. In the case where each file has different header orders, the file should then be marked as mismatch
c. This should also take note of the capitalization of the headers as accordingly
   - This should not matter if the order is correct

Below lists the equivalence class of the outcomes depending on the headers

Test Scenario #	Test Scenario Description	Expected Outcome

1	File 1 header: {id, balance, …}
	File 2 header: {id, balance, …}
		System should return depending on the mismatch of the data

2	File 1 row: {balance, id, …}
	File 2 row: {id, balance, …}
		System should return a mismatch for all rows as the

3	File 1 row: {BALANCE, ID, …}
	File 2 row: {balance, id, …}
		System should return depending on the mismatch of the data

Fuzzy Testing

In this fuzzy testing test cases, there are 2 types of testing implemented.

1. Load testing
    a. The main end-to-end test has been put through 100 times of running test cases continuously with different test cases being generated
2. Edge cases
    a. Minor character swapping errors
    b. Single character differences

In the test cases, files stemming from “sample_file_1.csv” has been generated algorithmically using the Random library to swap characters around or change single characters in the file.

The changes in the sample file is then saved in a file named “expectedFuzzer.csv” and the new sample compared file is saved in “fuzzer.csv”.

The test will then run through the main function of the application and then produce the same result and compare the differences in the rows.

This test is run 100 times, varying between the 2 edge cases, which is also a representation of load testing for the system.