Review of tests by Yi Da:

Yi Da's original tests are considerably comprehensive, covering a wide range from inputs to fuzzy mutation tests. Most of the test were very comprehensive and showcased the possible edge cases of the plausible shortcoming of the application. This are good tests that covered major aspects of the application.

Review of tests by Jarron:

Jarron's original tests are also comprehensive, covering a wide range from inputs to fuzzy testings. Comparatively to Yi Da, Jarron does not employ mutation tests, however he employs randomized text that has been generated to test for strange or unexpected inputs. This is good as it covers unexpected outcomes due to the application and increase the coverage of code testing in another method.

Fuzzy Test by me:

For the fuzzy test, as the general end-to-end service of the app should be the same, the end-to-end fuzzer that has been used in my code has been reused for Jarron's and Yida's code.

The fuzzer uses the concept of mutation to randomly swap letters and trim letters from existing user files. As an add-on to Yi Da's test, my test runs repeatedly with randomized mutation applied to it, which can also add on to load testing.

However, some of the code is modified slightly to be able to fit the inputs/usage of their applications. The changes will not be discussed here as the changes are not imperative to the outcome of the actual results of the test cases.

Results for Yi Da:

```
"ID66","BOS2168166","SEK","SAVINGS","969346"
"IDp6","BOS2168166","SEK","SAVINGS","969346"
```

The test case that was run 100 times were successful such that it was able to identify the mismatch of the CSV as expected. Thus, no bugs have been found and Yi Da's end-to-end implementation is successful according to my tests. The load testing on Yi Da's implementation works well too and can process and detect the differences in the application quickly.

Results for Jarron:

```
"ID99","BOS8059799","SGD","CURRENT","208045"
"ID99","BOS8059799","SGD","CURRENT","208043"
```

The test cases that was run for Jarron has also been successful. As Jarron's original implementation did not take on mutation testing, this increased the coverage such that his implementation could detect minor changes and differences, thus proving it to be working as expected. Furthermore, his implementation worked quickly and was able to process the differences quickly and passed load testing.