Allowlist/Droplist Challenge

Testing Summary + Bug Reports

**Test Objective**

I focused testing on the complex variables and interdependencies of the program, in particular using a decision table to map out an extensive range of scenarios to run through. Through this approach I aimed to prioritise testing of both the example scenarios in the program documentation, as well as the most common use cases of the product.

I also explored each of the error messages outlined in the documentation, ensuring they were all performing as expected.

I used a script to generate data for 16 different test cases, exploring an additional 6 cases with no specific test data requirements. In addition, I used another script to generate data to explore a bug in more detail.

**Areas Covered**

Testing focused on the most common use cases of the product:

* Interactivity between different variables outlined in the documentation (originals, updates, allowlist, droplist)
* Expected errors outlined in documentation (missing files, duplicated files, incorrect commands, incorrect formatting)

Testing initially covered a small number of files (8 original, 8 updated), then expanded to a larger number to explore a specific bug (20 original, 20 updated).

A list of specific test cases can be found [here](https://docs.google.com/spreadsheets/d/1UFoVghHd6_JwoIrt0flou56PYCe9eqoE_t9Eb5-hjEY/edit?usp=sharing).

**Areas Not Covered**

I did not incorporate the following into my testing:

* Thorough boundary testing of allowlist and droplist files (e.g. high number of rows in file), or of originals and updates directories (e.g. high volume of files run)
* Thorough boundary testing of names, including long and unusual names, and less common characters.
* Thorough boundary testing of addresses, including long and unusual formats.
* Testing focused on UK names and addresses only. Note that other countries will likely have longer or alternately formatted names, and a variety of different address formats.

**Overall Summary**

Generally the product worked as expected, with a few exceptions:

* Original files not consistently added to finals directory, despite being included in allowlist.
* Final files become corrupted when run with a droplist, where updated address line lengths are shorter than original address line lengths. **Bug report below.**
* When generating test data, it’s important to not include the filetype in the file name, e.g. “Baxter.txt”, as the final files are generated using the entire name, e.g. “Baxtertxt”. It did not cause issues with testing, but is noteworthy feedback.
* File names also were found to not incorporate unusual characters, e.g. “-” found in double-barrelled last names. Some further testing here would be beneficial.
* If being used, allowlist file cannot be blank and raises an error.
* Error handling for address formatting needs some adjustment, as it’s currently based on the address length, rather than specific issues with the formatting. **Bug report below.**

Finally, further testing is required in a few key areas, to ensure the limitations of the product are fully understood. I’d recommend the boundary testing outlined in the Areas Not Covered section above, as this will help ensure a more robust product, and one that is able to handle less common names and addresses that may appear in the data, even within the UK.

Bug reports

**Bug report 1** - Corruption of final files when run with a droplist

**Severity:** High

**Priority:** High

**Observed Behaviour:** When testing the product with file names present in both the originals directory and the updates directory, and where a droplist is present that does not contain those file names, we found the resulting files generates in the finals directory were corrupted, containing an amalgamation of both addresses.

When exploring further, it appears that the updates files are overwriting the originals files, but only by characters per line. As a result, if an updates file contains lines of text that are shorter in length than their corresponding originals file, the additional text is not replaced, and both become merged.

**Examples:**

| **originals** | **updates** | **finals** |
| --- | --- | --- |
| Mr Jordan Cunningham  Flat 93M  Kimberley springs  South Kim  TR85 8LB | Mr Jordan Cunningham  9 Gallagher crest  Jasonbury  BL5 8SB | Mr Jordan Cunningham  9 Gallagher crest  Jasonbury springs  BL5 8SBim  TR85 8LB |
| Naomi Dunn  Studio 91M  Pauline green  East Pauline  W6S 3FW | Naomi Dunn  9 Douglas pines  Callumborough  LA0 9TA | Naomi Dunn  9 Douglas pines  Callumborough  LA0 9TAuline  W6S 3FW |
| Gavin Gough  Flat 84  Richard gateway  Sharpchester  WN8 6FX | Gavin Gough  578 Jill motorway  Gemmachester  CT01 1XN | Gavin Gough  578 Jill motorway  Gemmachesterway  CT01 1XNster  WN8 6FX |

In contrast, the following appears correct, as original address lines are all shorter in length than their updated counterparts, so are entirely over-written.

| Rebecca Bishop  0 Gail drive  Shaneville  LL5 6LQ | Rebecca Bishop  524 Arnold port  Lake Kerryview  HX9B 0WA | Rebecca Bishop  524 Arnold port  Lake Kerryview  HX9B 0WA |
| --- | --- | --- |

More examples can be found from row 57 down in [this document](https://docs.google.com/spreadsheets/d/1UFoVghHd6_JwoIrt0flou56PYCe9eqoE_t9Eb5-hjEY/edit?usp=sharing).

**Expected Behaviour**

In all the above cases, since the files are contained in both the originals directory and the updates directory, and are not included in the droplist file, the final directory should contain all the updated versions.

**Reproduction Steps**

1. Generate test data for a user in both originals and updates directories
2. Ensure address lines in originals file are longer in length than address lines in updates file (feel free to use above examples if this is beneficial)
3. Ensure the dropall file is present, but does not contain the test file name
4. Run the program, and check the files in the finals file
5. You should see files with a combination of both addresses, where additional characters from the original address are added on the end of the updated address.

**Bug report 2** - Error handling for address formatting needs refinement

**Severity:** High

**Priority:** Medium

**Observed Behaviour:** When running the program with unusually formatted addresses, the error handling is behaving incorrectly. It appears that the error is flagged when an address is fewer than 4 rows, or greater than 5 rows. Whilst this aligns with the majority of UK-based addresses, it raises issues for valid addresses that fall outside of those boundaries, as well as invalid addresses that fall within them.

**Examples:**

The following is invalid due to length, but a valid UK address:

Helen Adams

Studio 11x

The Haven

Church Road

Stockport

Greater Manchester

FK5 7YR

The following is valid, but an invalidly formatted address:

Ruth Baker

Studio 64E

Alice valley

New Debra, B05

1FJ7LJ

**Expected Behaviour**

Addresses with fewer than 4 lines, or greater than 5 lines should be considered valid, provided they contain no formatting inconsistencies.

Addresses with between 4 and 5 lines should not be automatically considered valid, and should be flagged if they contain formatting inconsistencies.

**Reproduction Steps**

1. Generate test data for a user in originals directory, where address is under 4 lines, for example:

Helen Adams

12 Walney Road

M22 9TL

1. Do not generate any file in the updates directory.
2. Ensure the dropall file is present, but does not contain the test file name
3. Run the program
4. You should receive an error message along the lines of “ERROR: document " Adams " doesn't contain an appropriately formatted address”
5. Generate test data for a user in the originals directory, where address is 4 or 5 lines in length, for example:

Ruth Baker

Studio 64E&^(@)!

Alice valley

New Debra,B05

1FJ7LJ

1. The program will run as expected, and a file will be generated in the finals directory, with the above input, despite being formatted incorrectly.