CSCI 370 Final Report

Salesforce Squadron - Data Harmonization

William Shellberg
Calvin Ko
Bryce Merritt
Ian Marchbank

Revised June 15, 2024

CSCI370 Summer 2024

Professor Kathleen Kelley

Table of Contents

I. Introduction

Our client, Salesforce, develops and offers software, such as Customer Relationship Management (CRM) software, via the Software as a Service (SaaS) model to businesses to increase sales, improve marketing, and build trust with their customers. For the past 20 years, the users of Salesforce software have been allowed to enter freeform, free-text address information into the user interfaces of existing Salesforce solutions. They have since moved to a standardized picklist, or dropdown list, to capture address information from about 6% of the user base. Salesforce is trying to increase the number of users following the new picklist format, but an undesirable challenge for Salesforce is that the existing freeform entries do not match up to the new picklists conveniently. They have tasked our team to develop an ISO Code Address Classifier solution prototype that maximizes the number of conversions of existing freeform address entries into the new standardized picklist format. This enhancement provides Salesforce the option to move away from allowing user input via the old freeform entry method and into their new standardized system.

While we maximize the number of successful, accurate conversions, we also need to maintain a clean and quick experience for our users, while keeping their data secure and private. We met this requirement by developing a new solution from the ground up, not by modifying the currently-used, outdated software. The user experience is optimized by the intelligent use of a web app that allows existing users to see suggested address conversions, and either accept or modify those conversions. This means the user has a direct and persistent interface with our solution, integrating some of the required software maintenance via user-defined conversion rules, which are unique to each Salesforce customer.

II. Functional Requirements

Our solution must be able to connect to, or take in a database of user-input freeform address entries, apply conversion methods and present them to the user in a sandbox environment where they can manually edit conversion rules, see pain points in their data and otherwise tinker and analyze before it is shipped to a production environment. We are striving for maximum possible conversion and ease of user experience through minimization of clicks and a clean, informative web app interface.

III. Non-Functional Requirements

The final solution should reflect a level of optimization that allows the maximum conversion to be completed in a relatively short amount of time, providing a seamless experience to the user. It also needs to demonstrate a level of documentation such that engineers at Salesforce can easily convert it into a solution that integrates well with their ecosystem of existing solutions. For security reasons, our solution must rely exclusively on connection to the client's database, and cannot connect to any third party sources during its operation.

IV. Risks

There are a couple technical risks that came with this project. The first being that our emulated database may not match what the clients database looks like as we were not able to look at any client data for testing. This is moderately likely, but this would be easy to solve by the engineers rebuilding our solution in the salesforce ecosystem. Our algorithm may also mislabel data. This is highly likely to occur, especially with the later filters that utilize machine learning techniques. Luckily, this is expected in large datasets and can be resolved by removing the usage of the later filters, though this will impact our solution's success if applied in future work.

There are also some skill risks that came along with this project. No one on our team knew anything about web apps. This poses a huge risk because we could have spent too much time learning web development and still come out with an inferior solution. Only a couple of us had experience with databases and none of us knew how to connect databases to an external application. This could have led to our team being underutilized while the database is being worked on.

Finally, there were some ethical risks. The solution may end up with a biased dataset in the end if it can convert some countries and addresses more than others. There are also security risks of dealing and interfacing with sensitive client databases. Because this is a prototype, the main security feature of our solution is that we did not implement external calls to third parties.

V. Definition of Done

The final solution combs through the data and gives a confidence level from 1-5 to each conversion. Conversion rules above a certain confidence threshold can be automatically approved to reduce manual user interaction. The user should then be able to scroll through all of the data that has been matched, and confirm whether or not the addresses should be converted according to the automatically generated rules. The user should also be able to do conversions based on different imported rulesets, but in a separate environment so that data can be reviewed before committing permanent changes.

VI. System Architecture

Our solution combines three separate systems: a Web App, a DatabaseManager and a Classifier. The user interacts exclusively with the Web App. Once the user initially reaches the Web App (hosted locally for now, cloud hosting needed for deployment), they are prompted to connect to their database. This connection is then established and handled by the DatabaseManager we created for our solution.

The DatabaseManager contains useful methods such as get_next_n() which generates a batch of addresses to process and initial error handling such as verifying a connection can be established. This DatabaseManager is responsible for transporting the unprocessed addresses to the Classifier system of our solution.

From here, the Classifier does all of the processing necessary on the input addresses to pass it through its internal filter stack. These filters check for matches and similarity to user-imported and user-defined rules (defined during the UI phases of the application flow) as well as a master list of all country and state alternates names and misspellings. These filters run sequentially, and we have implemented strong logic to only continue using computation power on filtering if necessary. The Classifier finally then returns the results of the filtering system in the form of a probable mapping ISO 3166 alpha-2 country or state code and a confidence for that mapping and the number of times that state, country or address occurs in the database.

For each different stage of processing (Country, State, and Address) (Figure 2b-2d), the user is prompted with a UI that allows them to tinker with the results of that stage's filters, retrieved from the intermediate database. A paramount goal of our solution was to minimize user clicks, so we group all similar occurrences of states, countries and addresses, allowing the user to approve or edit the mapping for all of those occurrences at once.

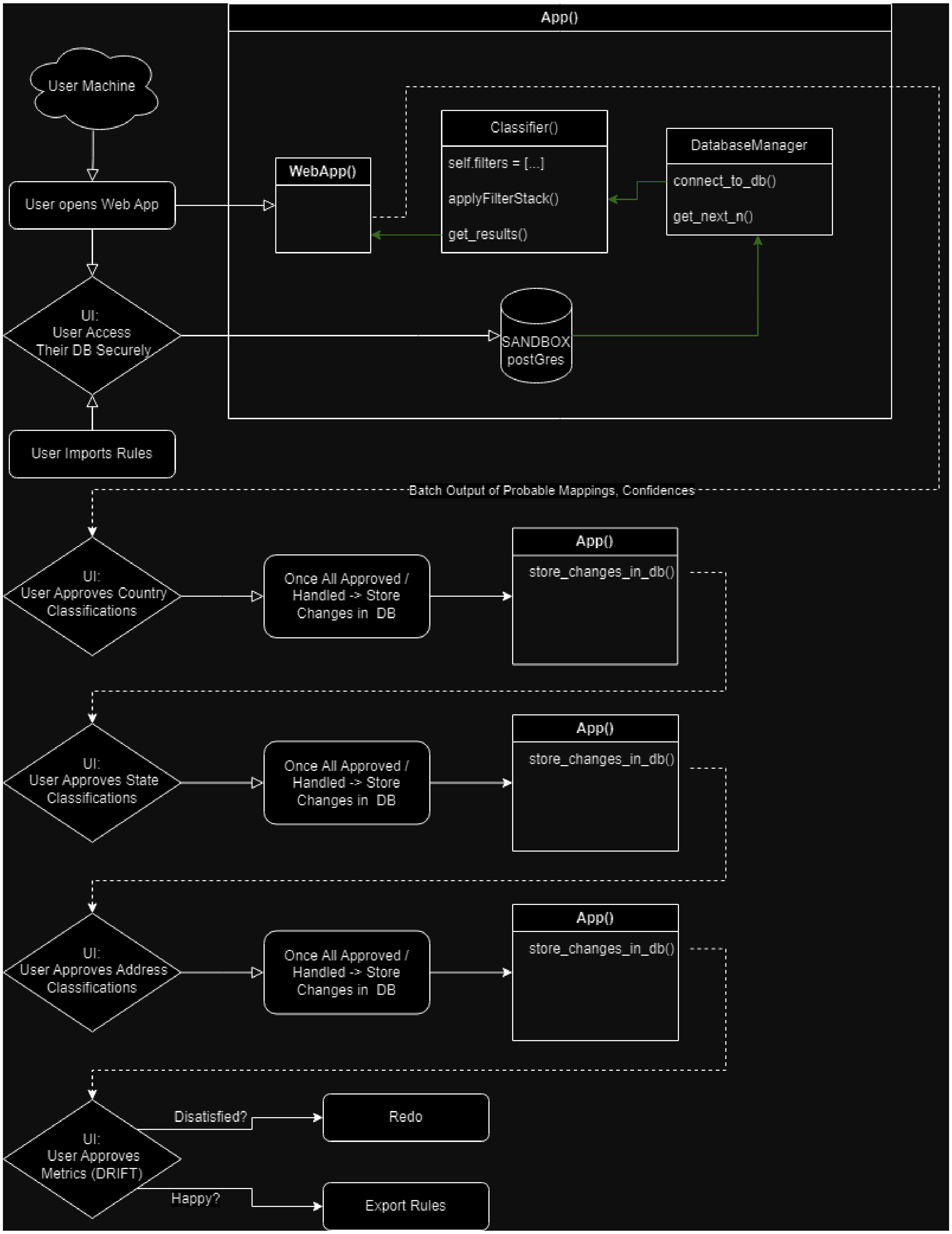
If you follow the Application Flow Diagram below (Figure 2), a sample run through would start the user on the Web App landing page UI (Figure 2a), prompting the user for their database connection and some configuration options. The database connection would then be established and maintained in the DatabaseManager class for the entirety of runtime. The App would then prompt the DatabaseManager to pass one batch of addresses to the Classifier to begin processing.

The classifier runs a series of filters and returns the results to the Web App to display to the user for approval or edits (Figure 2). Edits created are saved for this runtime so that future batches do not require you to redefine the same rule. Once all are approved or edited for each stage of the filter stack, the solution generates some overall metrics: total confidence, percentage conversion, and drift from solution-defined and user-defined rules.

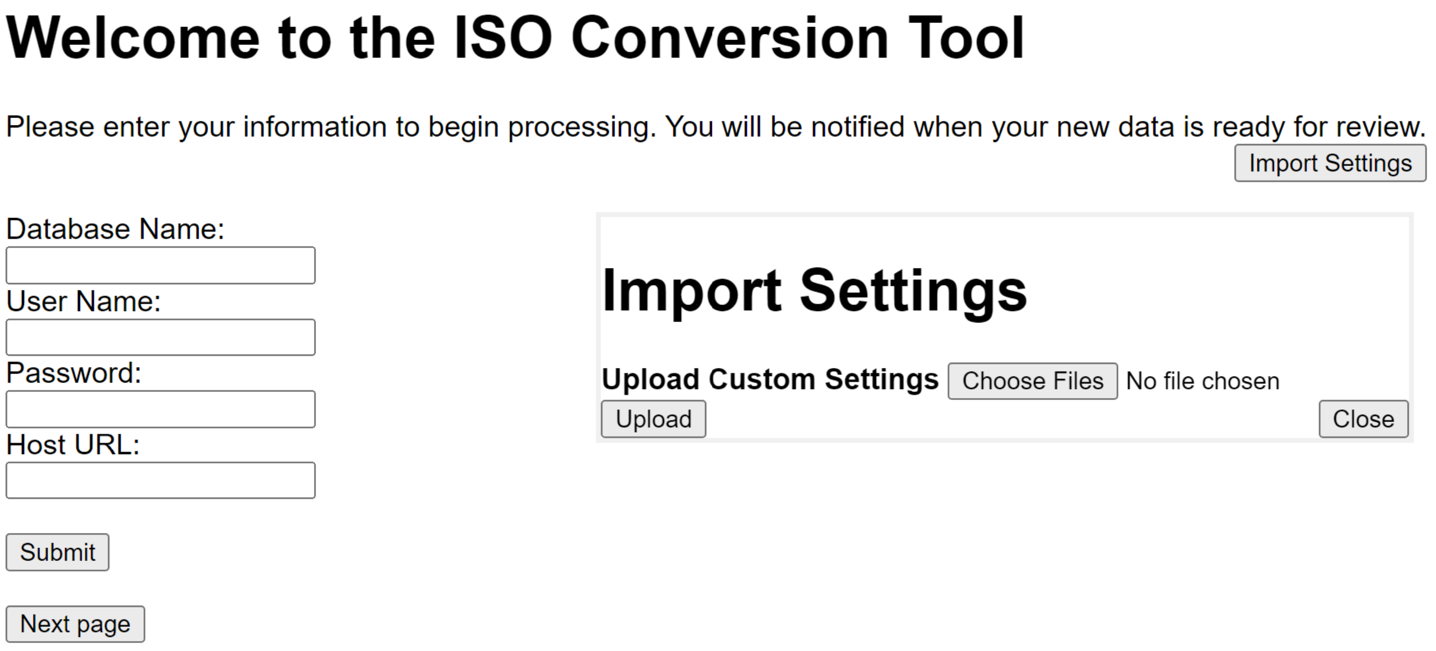
If the user is satisfied with the stage's results, the rules are saved and the next stage is called for. This is just a call to the next section of stored results because all of the processing is done beforehand, simply the results are stored in an intermediate database by stage of filtering as described above.. At the end of the process, the user can then decide to export their settings and reports from this sandbox environment, or restart the whole process.

In the UI, the user is presented with a series of different screens that display info about what needs to be changed. As stated before, these screens come in different stages, starting with the home page, then country approval stage, moving to the state approval, then address approval, and finally the statistics and confirmation page. In the home page, the user can access the option to import settings, which allows them to import their own rules for conversion in addition to the default ones implemented. In the approval screens, the layout is similar in all three with minor differences depending on which page they are on. Data is presented in different accordion tabs, with the parent being the country, displayed with the respective country ISO code. Inside the parent tab, different data is displayed. In the country approval stage, only identified country strings are displayed. In the state and address approval stage, the parent tab contains the corresponding state tabs that have the identified strings or unconverted addresses. In all stages, the user is able to go through the organized tabs and make manual changes, or just accept the conversions that the filter system has done, facilitating the process and making it faster.

At each stage, the processed data is displayed in a table that gives the address or string, the confidence, an approval checkbox, and a dropdown list that allows the user to manually change the country and/or state code. In the country and state approval stages, there is also an occurrences column, notifying the user of how many times a certain spelling occurred, indicating it might be a spelling that should be added to their own user rules. At the bottom the UI also has a search bar available, in case the user has a certain case that they would like to search for in order to manually change it. Shown below are screenshots of the UI.


App()
User Machine
User opens Web App
WebApp()
Classifier()
self.filters = [...]
applyFilterStack()
get_results()
DatabaseManager
connect_to_db()
get_next_n()
UI: User Access Their DB Securely
SANDBOX postGres
User Imports Rules
Batch Output of Probable Mappings, Confidences
UI: User Approves Country Classifications
Once All Approved / Handled -> Store Changes in DB
App()
store_changes_in_db()
UI: User Approves State Classifications
Once All Approved / Handled -> Store Changes in DB
App()
store_changes_in_db()
UI: User Approves Address Classifications
Once All Approved / Handled -> Store Changes in DB
App()
store_changes_in_db()
UI: User Approves Metrics (DRIFT)
Disatisfied?
Redo
Happy?
Export Rules

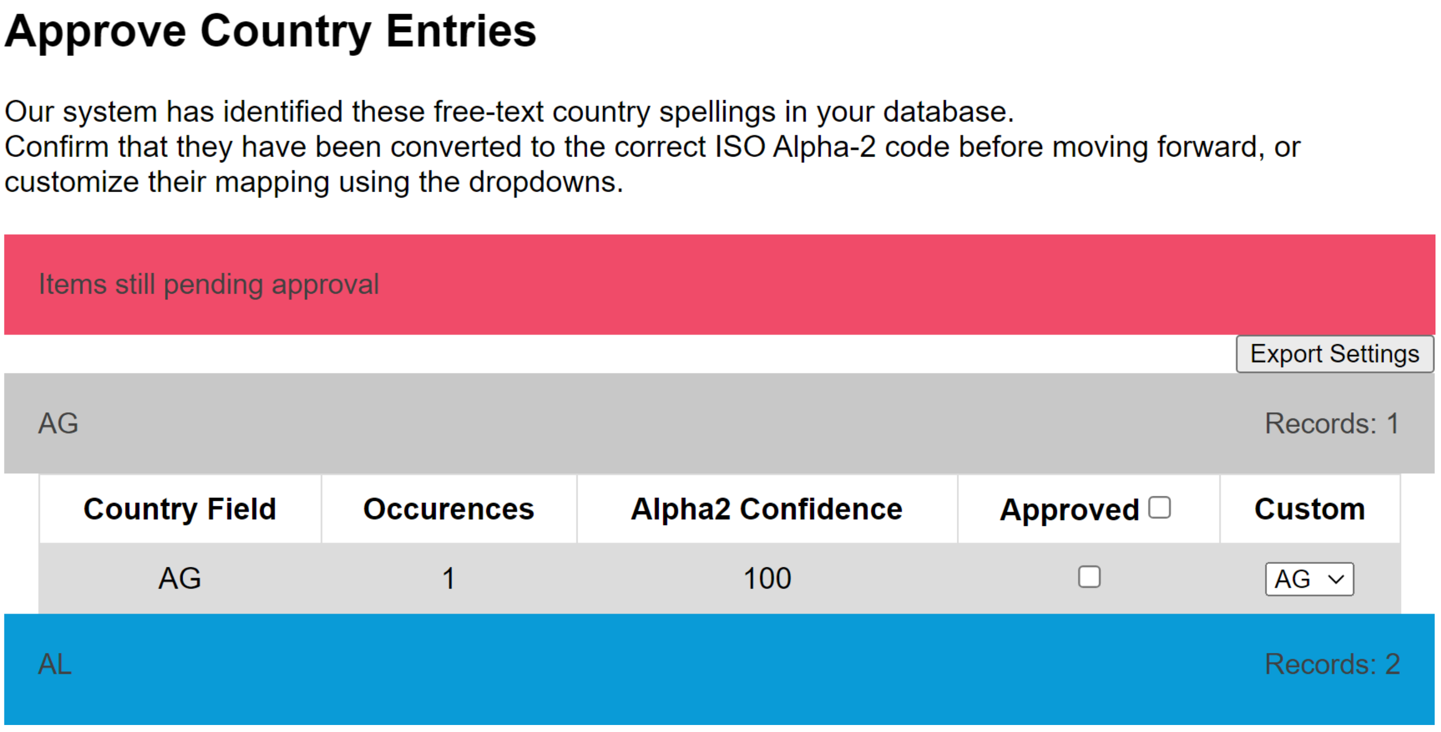
Backend Flow Diagram (Figure 1)



Welcome to the ISO Conversion Tool

Please enter your information to begin processing. You will be notified when your new data is ready for review.

Application Flow Diagram - Landing Page (Figure 2a)



Approve Country Entries

Our system has identified these free-text country spellings in your database.
Confirm that they have been converted to the correct ISO Alpha-2 code before moving forward, or customize their mapping using the dropdowns.

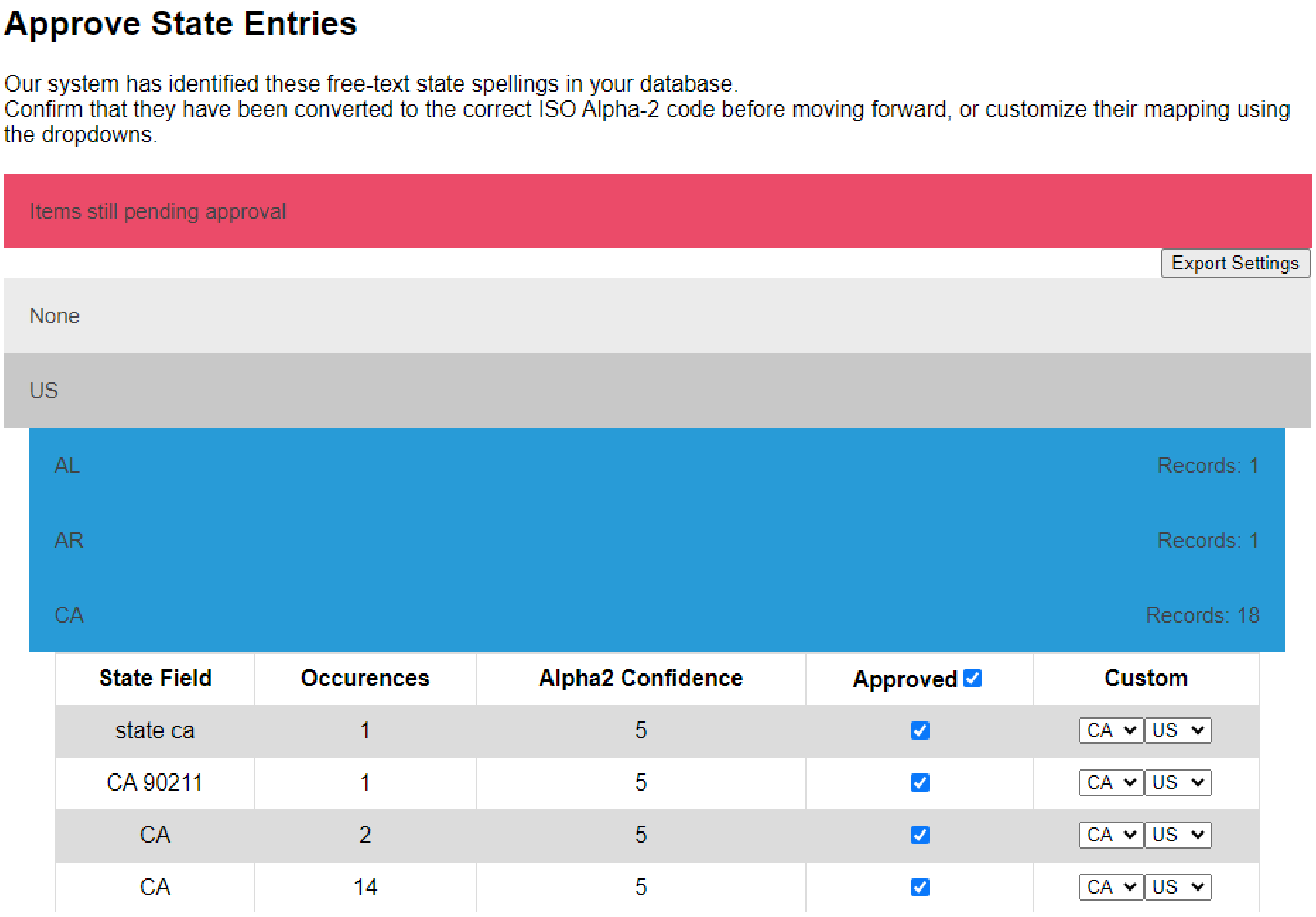
Items still pending approval

AG Records: 1

Country Field	Occurences	Alpha2 Confidence	Approved ☐	Custom
AG	1	100	☐	AG

AL Records: 2

Application Flow Diagram - Country Stage (Figure 2b)



Approve State Entries

Our system has identified these free-text state spellings in your database.
Confirm that they have been converted to the correct ISO Alpha-2 code before moving forward, or customize their mapping using the dropdowns.

Items still pending approval

Export Settings

None

US

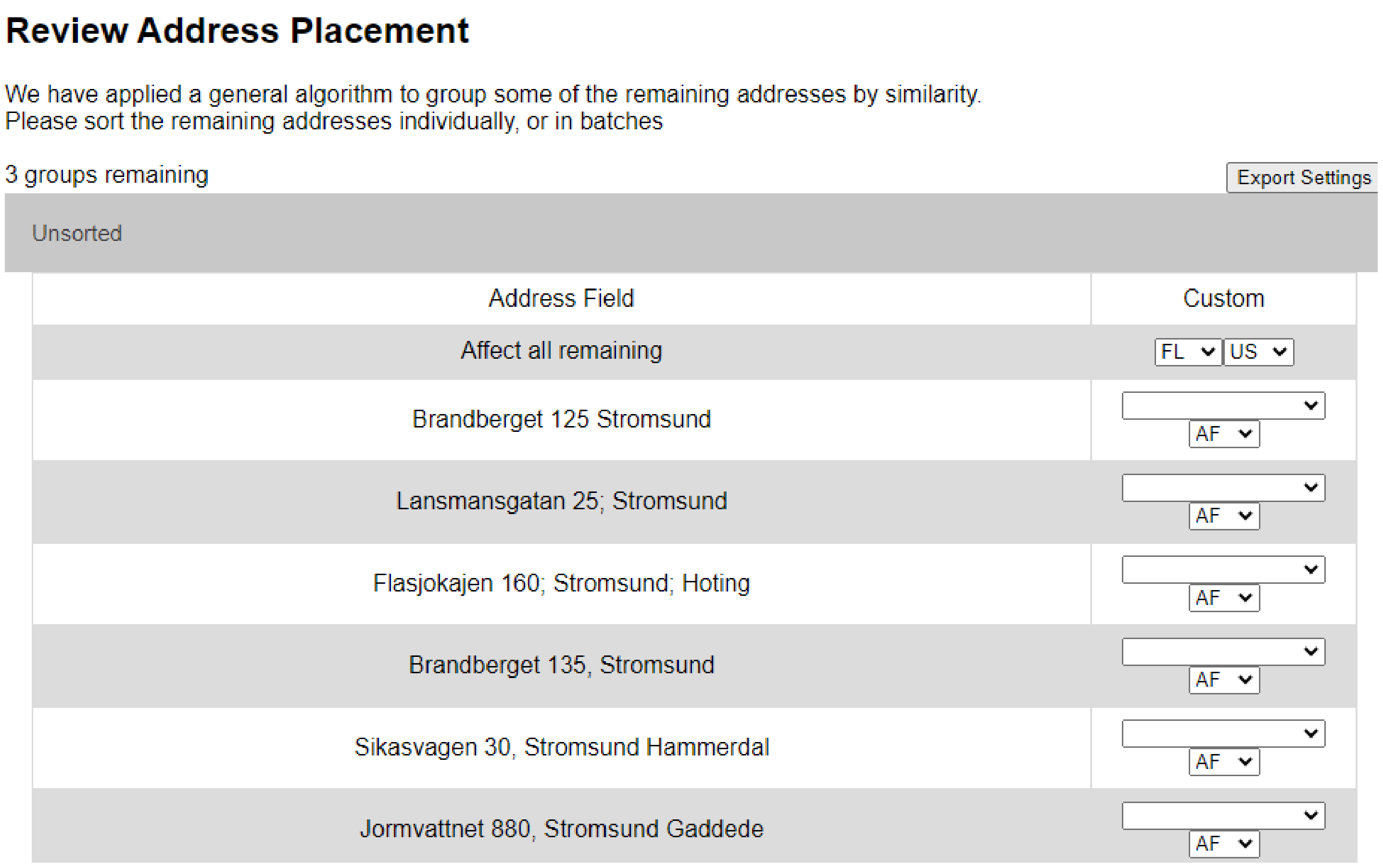
AL Records: 1

AR Records: 1

CA Records: 18

State Field	Occurences	Alpha2 Confidence	Approved ✓	Custom
state ca	1	5	✓	CA US
CA 90211	1	5	✓	CA US
CA	2	5	✓	CA US
CA	14	5	✓	CA US

Application Flow Diagram - State Stage (Figure 2c)



Review Address Placement

We have applied a general algorithm to group some of the remaining addresses by similarity.
Please sort the remaining addresses individually, or in batches

3 groups remaining

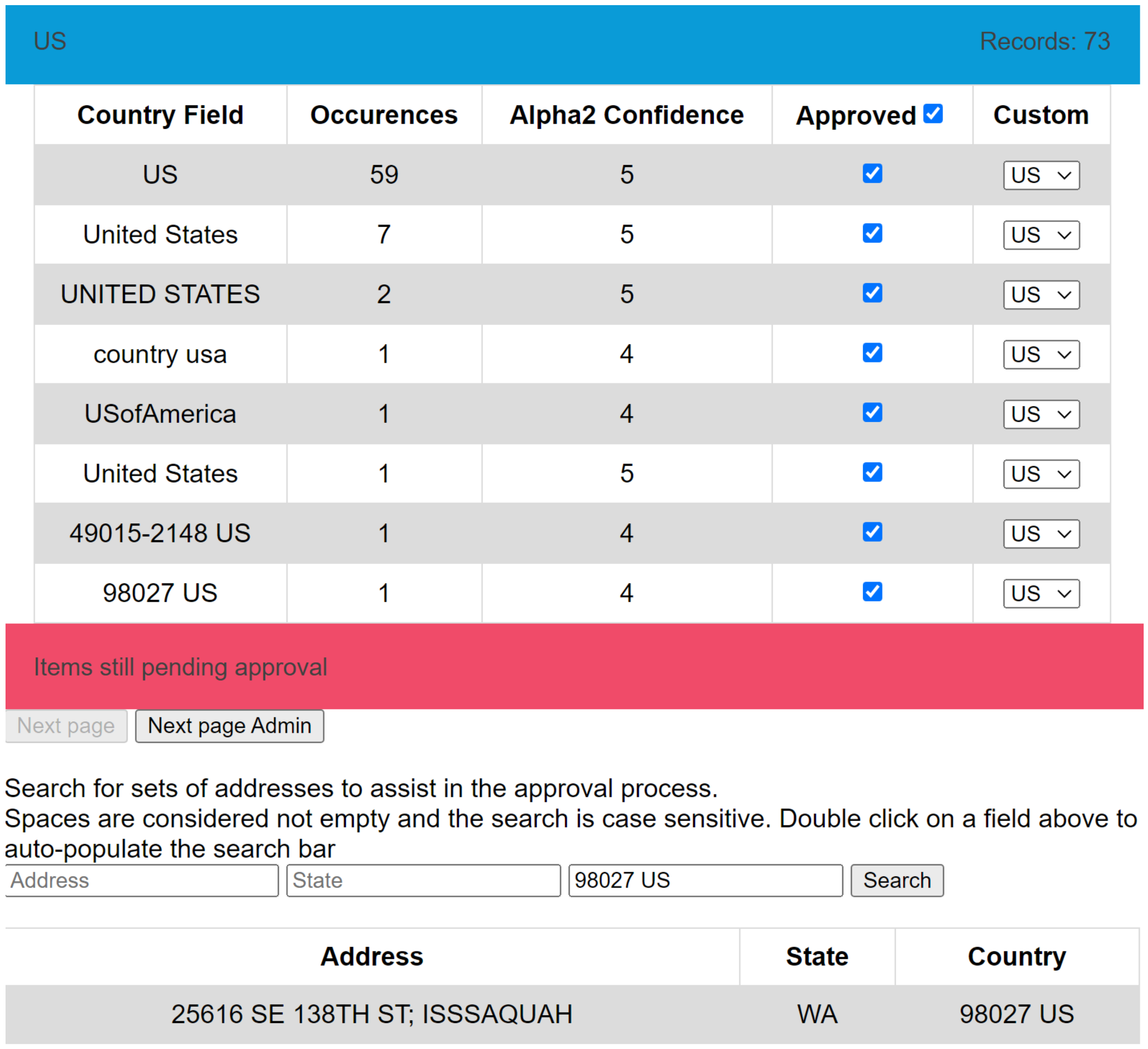
Export Settings

Unsorted

Address Field	Custom
Affect all remaining	FL US
Brandberget 125 Stromsund	AF
Lansmansgatan 25; Stromsund	AF
Flasjokajen 160; Stromsund; Hoting	AF
Brandberget 135, Stromsund	AF
Sikasvagen 30, Stromsund Hammerdal	AF
Jormvattnet 880, Stromsund Gaddede	AF

Application Flow Diagram - Address Stage (Figure 2d)

4



US Records: 73

Country Field	Occurences	Alpha2 Confidence	Approved ✓	Custom
US	59	5	✓	US
United States	7	5	✓	US
UNITED STATES	2	5	✓	US
country usa	1	4	✓	US
USofAmerica	1	4	✓	US
United States	1	5	✓	US
49015-2148 US	1	4	✓	US
98027 US	1	4	✓	US

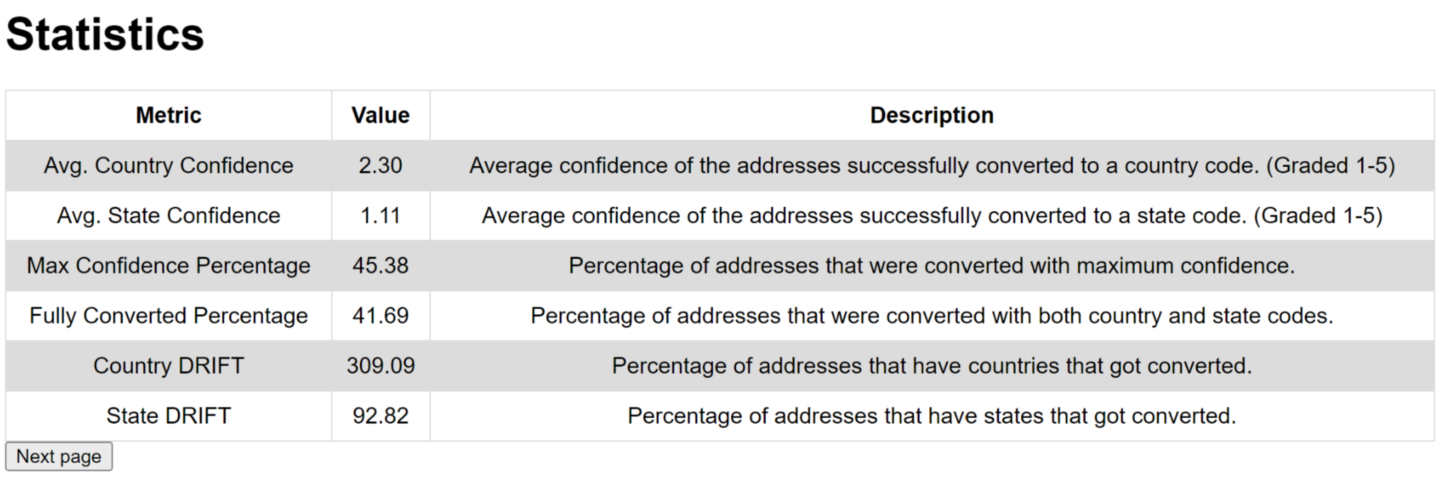
Items still pending approval

Search for sets of addresses to assist in the approval process.
Spaces are considered not empty and the search is case sensitive. Double click on a field above to auto-populate the search bar

Address: (empty) | State: (empty) | 98027 US

Address	State	Country
25616 SE 138TH ST; ISSSAQUAH	WA	98027 US

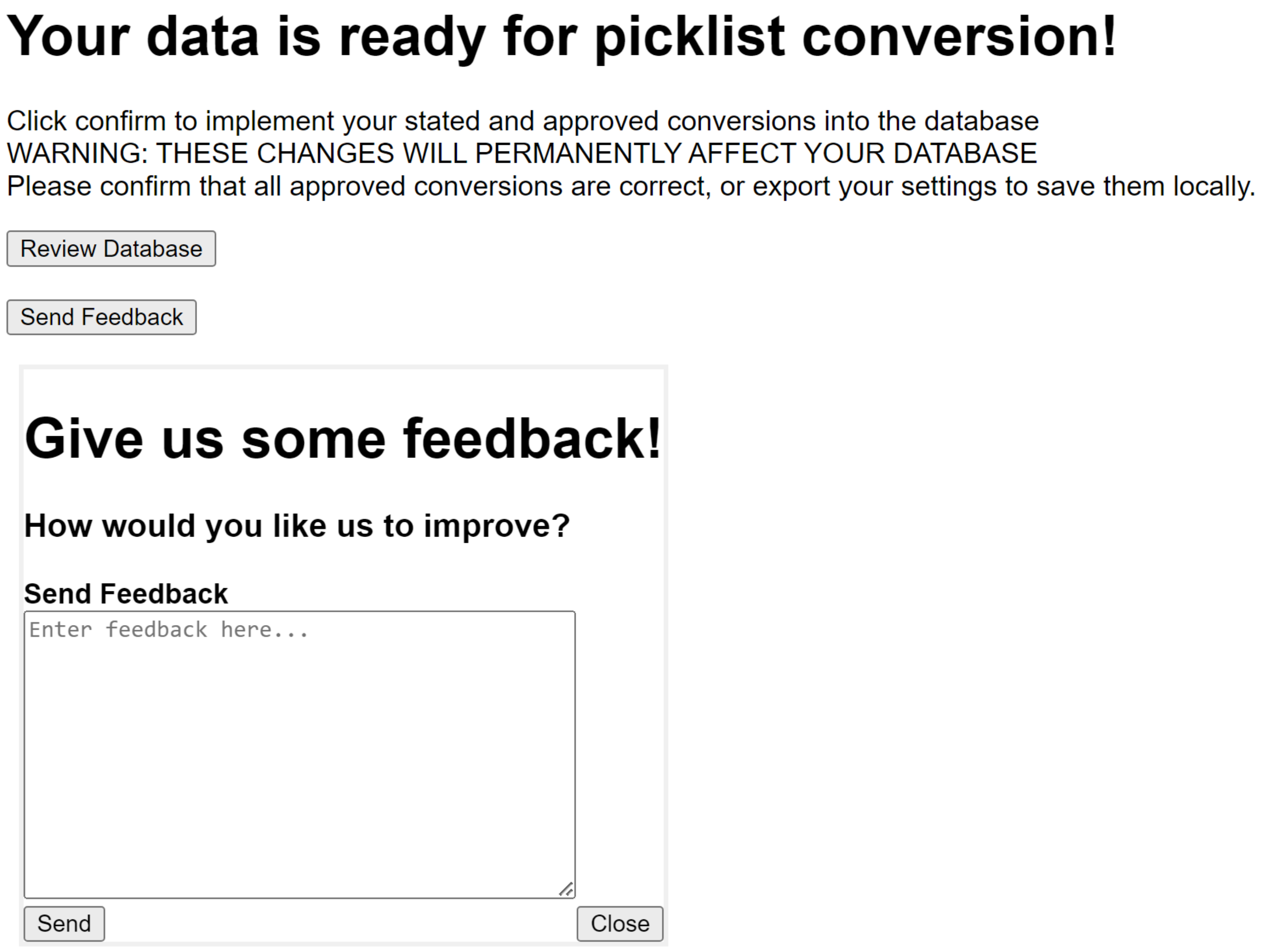
Application Flow Diagram - Search Function (Figure 2e)



Statistics

Metric	Value	Description
Avg. Country Confidence	2.30	Average confidence of the addresses successfully converted to a country code. (Graded 1-5)
Avg. State Confidence	1.11	Average confidence of the addresses successfully converted to a state code. (Graded 1-5)
Max Confidence Percentage	45.38	Percentage of addresses that were converted with maximum confidence.
Fully Converted Percentage	41.69	Percentage of addresses that were converted with both country and state codes.
Country DRIFT	309.09	Percentage of addresses that have countries that got converted.
State DRIFT	92.82	Percentage of addresses that have states that got converted.

Application Flow Diagram - Statistics Page (Figure 2f)



Your data is ready for picklist conversion!

Click confirm to implement your stated and approved conversions into the database
WARNING: THESE CHANGES WILL PERMANENTLY AFFECT YOUR DATABASE
Please confirm that all approved conversions are correct, or export your settings to save them locally.

Give us some feedback!

How would you like us to improve?

Send Feedback

Application Flow Diagram - Results and Confirmation Page (Figure 2g)

VII. Software Test and Quality

The first part of our software testing and quality plan are our timing tests. We created a suite of timing tests for each of our database queries and each of our filters. The timing constraints are operation and sample database dependent, but we sought to complete most operations within a few seconds. Our deliverable was expected to run overnight (or longer), thus our system does not have to be very quick, but the sample database we worked with is 10,000 times smaller than the true databases. Thus, if we were not strict with the timing constraints on the smaller data, it may not have scaled to the larger databases. We also needed to make sure we were not using an excessive amount of computing resources as these are not free.

The next part of our software testing plan involved unit testing. We created unit tests for each of our filters, the database connection, each of our database queries, and each of our statistics functions. We needed to ensure that each of these functions return accurate information to the other functions and users.

The final part and most important part of our software testing and quality plan involved manual user testing. To test our web app interface, we needed to simulate a user's behavior. We did this by passing sample data to our app, making sure that it displayed the proper information to the user and that any "bad" user behaviors resulted in an exception being thrown rather than breaking the user experience. Below is a table with the names of each of the tests we created

Table 1: Tests

Name of Test	**Type of Test**
Country Filter Execution	Timing
State Filter Execution	Timing
Processing Filter Execution	Timing
Get String Frequency in DB Execution	Timing
Search Address in DB Execution	Timing
Get entire table from DB Execution	Timing
Country Filter Accuracy	Unit
State Filter Accuracy	Unit
Processing Filter Accuracy	Unit
Get String Frequency in DB Accuracy	Unit
Search Address in DB Accuracy	Unit
Get entire table from DB Accuracy	Unit
Connect to DB Accuracy	Unit

Statistics Accuracy	Unit
User display Accuracy	Manual user testing
User display Exceptions	Manual user testing

VIII. Project Ethical Considerations

One of our ethical concerns is ensuring the accuracy of our database changes. There is an inherent risk in any automatic data cleaning system making false conversions. These inaccurate conversions could lead to flawed analysis of the data, damaged customer data, and a worse customer experience. To mitigate these risks, we have implemented comprehensive testing as explained in the previous section. Our system is also designed as a sandbox environment with the user able to view each of these changes and approve them before they are pushed to production. By prioritizing user control and ensuring our system's accuracy, we know users can be confident in the safety and accuracy of our product.

Another critical ethical consideration is the potential for bias in the converted database. We recognize that the data we were provided with was biased towards Latin Alphabets and there was a risk that our system converted Latin Alphabet addresses with greater accuracy than other alphabets. This could have lead to a biased dataset if the user only utilized cleaned data for their analysis. To mitigate these risks, our system was designed to incorporate user conversion rules and future expansion of the default rules/filters. By allowing the user and Salesforce to easily expand the system and override existing convention we ensure that the conversions can be more comprehensive in the future.

Our final ethical consideration was ensuring the security of the application. The solution at Salesforce will be connecting to databases with potentially sensitive user information, which must not be leaked. Therefore, in the prototype, we needed to have a secure connection to the database to avoid consequences including identity theft, financial loss, and damage to Salesforce's reputation. Our team lacked the necessary expertise and time to create a fully secure application to the standards of the Salesforce team. We instead attempted to mitigate this risk by making the application as secure as we were able to and documenting which areas have vulnerabilities so that the team rebuilding our work can have an easy time filling in the holes that we were unable to.

IX. Project Completion Status

We accomplished most of what we set out to achieve for this project. The nature of the project as a "proof of concept" meant that our code will not be extended by the engineers at Salesforce, but to be very explicit about what we were able to achieve, here is a list of the features we were able to implement and the ones we were not.

Implemented

- Database connection established
- Modular filtering system design and infrastructure
- Exact match filter created
- Fuzzy match filter created
- Processing filter created
- Initial hash table of common names created

- Testing environment created
- Dynamically updating web pages for the steps in our user flow
- Force users to review changes in web app before moving on
- Search function within web app
- Auto select based on confidence in web app
- Statistics review upon approval completion

Unimplemented

- Import/export user rules from web app
- Better ML approach would need to be researched; DBSCAN isn't cutting it.

IX. Future Work

While we were able to finish a lot of what we set out to, there are still some areas that we would like to improve on. There were also a small number of bugs that can affect the user experience that we were unable to resolve by the end of the project. Our focus for the prototype was much more oriented on the general user flow and classifier system. We hope that we have given the Salesforce engineers a jumping off point to implement some of the goals that we were not able to get to, but in their own language/environment.

General Environment Resource Requirements:

- Python 3.12 (w/ flask, numpy, pandas, pg8000, scikit-learn installed)
- Postgres installed on computer
- Test database created on computer (setup included in documentation)
- Quick start guide (included in project files)

Table 2: Future Work Information

Feature	**Knowledge**	**Time Estimate**
Enhance web app security	Cybersecurity, web apps	1 week
Analyze/enhance database security	Cybersecurity, SQL	1 day - 1 week
Import user conversions	Python	1-2 days
Export user conversions	Python	1-2 days
Expand default conversion database	ChatGPT	1 week
Improve algorithm speed	Python, SQL	1-2 weeks
Create a better final filter/filtering system	Python, SQL	2-3 weeks
Test on real user data (volunteer	Salesforce engineer	1 week

basis)		
Convert project to a faster language from python	Java, Rust, or C++	2-3 weeks
Bug Fixes: Accordions not opening. Country dropdowns not moving items	Javascript, css, HTML	2 days

X. Lessons Learned

We learned many lessons from this project. The key lesson we learned is that the “Dunning Kruger effect” also applies to software engineering. The closer we get to what we thought was the end of our project, the further we actually feel from a production ready product. There is so much more that would have to be done to our product to make it ready for production, such as adding security measures, testing on actual user data, and expanding the system’s robustness. We feel very lucky to be passing this project onto the engineers at Salesforce to complete these tasks. Another important lesson we learned is the value of doing a lot of research before diving into unfamiliar territory. We found it helpful to spend a large amount of time researching web app development before diving into it. We would have probably found it even more helpful if we were able to put off development for another week, but could not due to the time constraints of the project. The last lesson we would like to share is that client meetings are more necessary earlier on in the project life cycle. We found that at the beginning of the project we would have many questions for our client and we would change our direction a lot based on what they said. Once we were well into development of the project the client meetings became more about showing off our product and the meetings could be spaced out a bit more.

XI. Acknowledgments

We would like to thank our client, Jens Schutt, for working with us on this project. He has been an incredible client and guide through this project, and we appreciate the time that he has given to cooperate with us on meetings and giving feedback and advice. We would also like to thank our advisor, Kathleen Kelley, for providing guidance throughout the project for how to handle problems and helping us to get in contact with our client initially.

XII. Team Profile

Bryce Merritt (Bachelors in Computer Science: Robotics and Intelligent Systems)

Bryce is a part-time student in Computer Science, part-time supervisor at the LEGO Store where he has automated the printing of the sale and box tags based on the manifest each shipment and learns human-centered skills. At home, he plays a lot of resource management games like Factorio and Oxygen Not Included, occasionally modding them or writing scripts to help him play. Outside of coding, Bryce enjoys reading, classical music and enjoying the weather from his back porch.

Calvin Ko (Bachelors in Computer Science: Robotics and Intelligent Systems, Minor in Applied Mathematics)

Calvin is a full-time student at Colorado School of Mines. At home, he plays a variety of games, mostly competitive but also casual such as Minecraft. Outside of school work he likes to ski and snowboard, and is an avid car enthusiast.

William Shellberg (Bachelors in Computer Science: Robotics and Intelligent Systems, Minor in Computational Applied Mathematics)

William is a full-time student in Computer Science. In his free time, he is a casual cellist and game developer. He also enjoys baking and Argentinian tango.

Ian Marchbank (Bachelors in Computer Science: Data Science)

Ian is a full-time student in Computer Science. He has worked for a TA for 3 years in various classes and loves helping students learn. At home, Ian enjoys listening to post punk/indie rock, lifting weights, and language learning (programming and spoken).

References

"ISO 3166 - Country Codes." *ISO*, International Organization of Standardization, 10 June 2021, www.iso.org/iso-3166-country-codes.html. Accessed 17 May 2024.

R, Adwait Dathan. "A Beginner's Introduction to Ner (Named Entity Recognition)." *Analytics Vidhya*, 20 Mar. 2024, www.analyticsvidhya.com/blog/2021/11/a-beginners-introduction-to-ner-named-entity-recognition/. Accessed 17 May 2024.

Appendix A – Key Terms

Include descriptions of technical terms, abbreviations and acronyms

Term	**Definition**
Salesforce	*Cloud-compute Software as a Service Company, our client.*
SaaS	*Software as a Service. Applications are hosted by a company and made available to users over the cloud.*
CRM	*Customer Relationship Management. A system for governing interaction with current and potential customers to ensure a standardized and optimal experience.*
ISO 3166	*International Organization for Standardization. An internationally recognized standard for referring to countries and states.*
Confidence	*A numerical score for how certain the classifier is of it's classifications.*
Address	*A sentence used to identify a physical location, formats and completeness may vary between Addresses.*
Picklist	*A more formal name for a dropdown system with a discrete number of options.*
Web App, DatabaseManager, Classifier (capitalized)	*Names of the 3 python classes we used to structure our System Architecture. Capitalized to identify them as proper nouns.*