CSCC01

Project Deliverable #5

Team Java Bean (Team 28)

Table of Contents

Note: personas have not changed, so they are not included

Sprint 05 Task Breakdown:

Sprint 05 consisted of us going back and updating existing user stories with one or two tasks, as well as implementing our sixth user story.

User Story 6: As Ellie (an Agency Analyst), I would like for the application to automatically merge postal codes and phone numbers if the only difference is their format, so that I don't need to worry which format the first uploaded document was in.

Task 29 – Support Client Info

- Handle duplicate clients with different Id during upload
- Handle file upload for client that does not exist
- Add popup to UI indicating if there are duplicate clients
- Add popup to IO indicating if no client exists

Task 30 – Improve Reports and implement graphs

- For the summary report, find an external package to generate visual graphs
- Implement this package for the existing summary report generator
- Export report as pdf instead of txt as it is currently designed
- Will need UI tests with screenshots for acceptance testing
- The third party library should be able to generate bar graphs, pie graphs, etc.
- A report should be generated for each column that a user has selected to filter by
- Ideally the API should be easy to use and extendible

Task 31 – Refactor Strings to Enums

- Map iCare sheet names to the respective service stream in the db (use an enum class)
- Add the other streams to the user table in the db (currently we only have Employment Service Stream as a column of user)
- Replace hard-coded strings with enums wherever possible

Task 32 – Create a regex formatter for phone numbers

- Given a string, return a formatted phone number
- Throw an error if phone number is invalid

Task 33 – Format Postal Code

- create a Formatter util class for all formatting methods
- find a regex that can validate Canadian postal codes and format it when necessary
- create and throw invalidValueException if the input is not valid to be formatted

Task 34 – Add Date Regex

- add date formatting method to Formatter class
- support the following cases:
  - yyyy-mm-dd || mm-yyyy-dd || mm-dd-yyyy
  - yyyy-dd-mm || dd-yyyy-mm || dd-mm-yyyy
  - support with separator as: / , - or " "

Task 35 – Create a regex to check if emails are of valid format

- Should return a Boolean – true iff email is formatted correctly
- Connect this to the register form

Task 36 – Email Formatter

- integrate the uploader and formatter classes
- check keys in parsed hashmap and pass values to formatter if the key field is supported
- throw exception to ui
- catch invalidValueException thrown by uploader in ui and display a error warning

Sprint 05 Planning and Execution:

Sprint 05 Plan									
User Stories	Tasks	Dependency	Story Points		Tuesday	Wednesday	Thursday	Friday	Saturday
3	29		3					MV:3	
4	30		8		A:2	A:2	A:2		A:2
1	31		2			B:2			
6	32		1			B:1			
6	33		1			MZ: 1			
6	34		1				MV:1		
6	35		2				B: 2		
6	35	T32, T32, T34, T35	5					MZ: 2	MZ:3

Sprint 05 Execution										
User Stories	Tasks	Dependency	Story Points		Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
3	29		3					MV:3		
4	30		8		A:2	A:1	A:1			
1	31		2			B:2	B:1	B:1		
6	32		1				B:1			
6	33		1			MZ: 1				
6	34		1				MV:1			
6	35		2				B: 2			
6	36	T32, T32, T34, T35	5					MZ: 1	MZ:3	MZ:1

Progress Reports:

Tuesday:

- sprint progressing as expected

Wednesday

- Not as much time needed for Task 30 as expected
- Task 31 taking longer than expected
- Task 32 got pushed back to Thursday

Thursday

- sprint progressing as expected

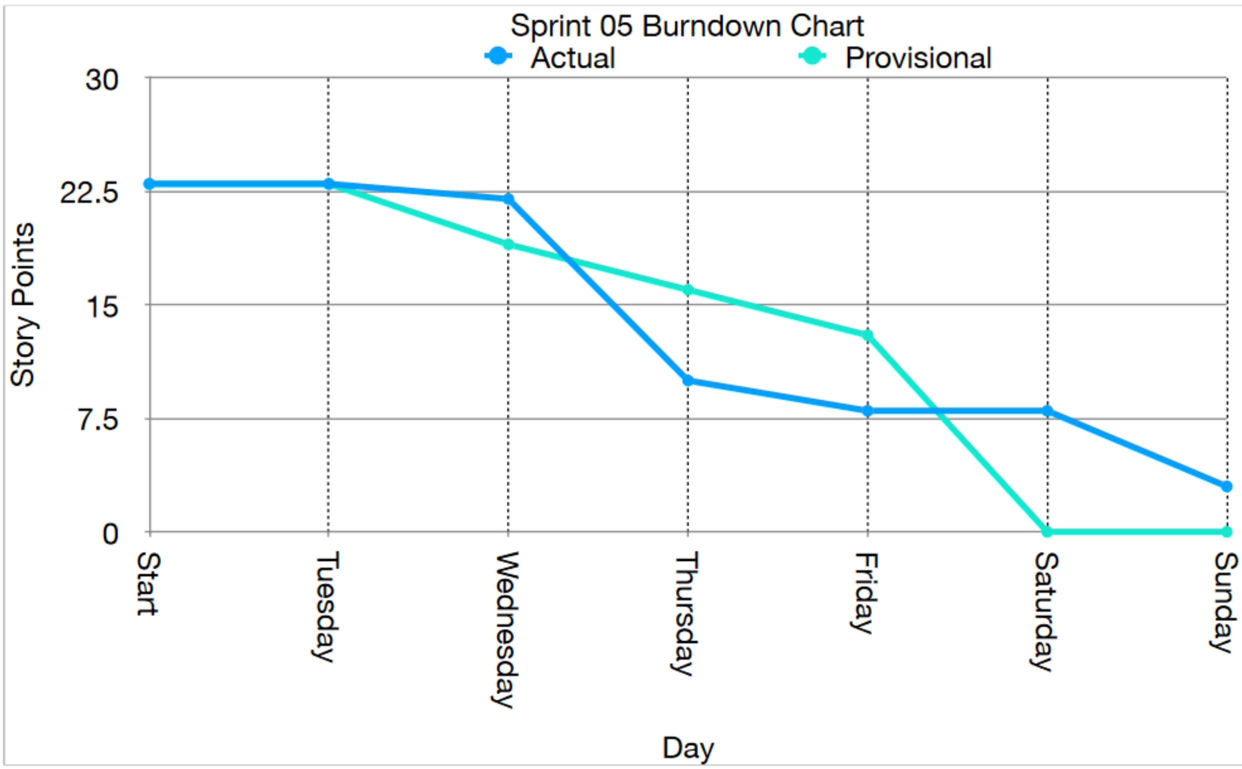
Friday

- Task 29 started taking longer than expected, and pushed back to next sprint due to low priority and big assignment for another class

Sprint 05 Burndown:

Sprint 5

	Actual	Provisional		
Start	23	23		
Tuesday	23	23		
Wednesday	22	19		
Thursday	10	16		
Friday	8	13		
Saturday	8	0		
Sunday	3	0		



Sprint 06 Task Breakdown:

Sprint 06 consisted of only of tasks which would improve existing features, rather than implementing a new feature. We also decided to have it be a shorter sprint, and finish on Thursday. This allowed for us to film our final product for the deadline, and work on bug fixes for the remainder of the week.

Task 29 – Support Client Info

- Handle duplicate clients with different Id during upload

- Handle file upload for client that does not exist
- Add popup to UI indicating if there are duplicate clients
- Add popup to IO indicating if no client exists

Task 37 – Logout

- add a logout tab in ui
- add logout logic in tabFactory while creating the tab

Task 38 – Add Support for Info & Orientation Service Stream

- Implement the Info & Orientation Service Stream
- Needs to be enabled in the UI
- - Need to create an enum class to select which columns can generate graphs for this stream
- Should be able to upload files to this stream as well

Task 39 – Add Support for Community Connections Service Stream

- Check that user can upload info into Community Connections table
- Add support for report generation for this stream

Task 40 – Add Support for Need Assessment service stream

- add NeedsAssessmentsColumnQueries enum class to match selected columns' ui names to db names
- enable NeedsAssessment service stream in report generator
- enable report generation for NeedsAssessment stream in ui
- No branch / commit exists for this as it was previously completed in a previous refactor by Brian

Sprint 06 Planning and Execution:

Sprint 06 Plan						
User Stories	Tasks	Dependency	Story Points		Tuesday	Wednesday
3	29		5		MV:2	MV:3
1	37		2		MZ:2	
2	38		2			A:2
2	39		2			B:2
2	40		2			MZ:2

Sprint 06 Execution							
User Stories	Tasks	Dependency	Story Points		Tuesday	Wednesday	Thursday
3	29		5		MV:2	MV:2	MV:2
1	37		2		MZ:2		
2	38		2			A:1	A:1
2	39		2			B:2	
2	40		2			MZ:2	

Progress Reports:

Tuesday:

- sprint progressing as expected

Wednesday

- Task 38 got pushed back Thursday
- 1 Hour of task 29 pushed back to Thursday

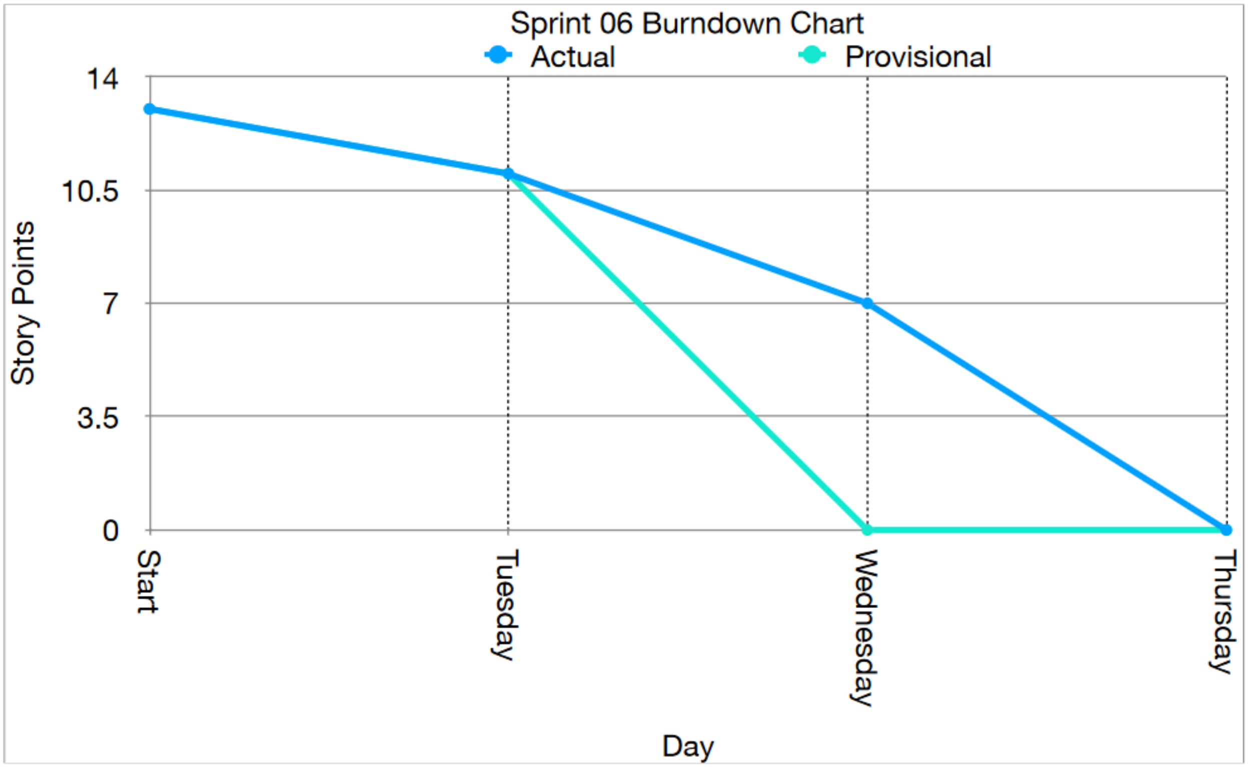
Thursday

- Sprint completed

Sprint 06 Burndown:

Sprint 6

	Actual	Provisional		
Start	13	13		
Tuesday	11	11		
Wednesday	7	0		
Thursday	0	0		



User Stories (Version 3):

The changes to our user stories for this sprint was the removal of the last two user stories (7 and 8). Other than that, they remained the same. These were removed as they were not necessary for a proof of concept, and could always be implemented at a later time if necessary.

Code Review Comments and Summaries:

Brian:

In completing the code review for sprints 5 and 6 (and our team's code review over all our source code), a few things I noticed were that while generally our team continued to fix our issues that we identified from sprints 3 and 4 (for example, documentation and hard-coded strings were places of particular improvement), there were still issues like unnecessary print statements that continued into the next sprint(s). However, after the team dedicated sprint 6 to refactoring our code base, I feel like most of our code is now clean. Some remaining issues that I've noticed were the existence of some code smells within our code. An example of this would be the huge if..else… statements that reside within our reports related code. I think that this would be the team's best example of the benefit of code review (despite not being able to be fixed) – had the team not reviewed my code and pointed it out, I would not have recognized the problem. Unfortunately, it could not be fixed easily with our code design. Overall, the code review for sprints 5 and 6 was beneficial to the team, and the majority of code which has had poor design have been redesigned.

Maggie:

For the latest two sprints, we proceeded with code refactors to address the issues we discussed in the previous review. I reviewed the pull request for task 31 which created support in util package to translate the ui names for service streams to their database names to avoid hard-coding strings scattered in different files. The methods getDbName() and getUiName() are neat for use in ui and util classes. If in case we need to change some string literals, we no longer need to look for all the strings in every project file, but instead we can change the mappings in this utility Enum class. Another comment about this PR was that I noticed when dealing with the unit test for database, we saved a copy of the entire database file in @BeforeAll and reset it in @AfterEach, which did solve the consistency issue of unit test result, but copying the entire db file may be a costly operation with the increasing database size, so maybe we can consider resetting specific test tables instead of the entire db file. What's more, right now we are relying on the acceptance tests for the uploader which integrates multiple util and database classes, if we could instead to use mock objects to test the functioning of that class, it would be a better testing practice in case that we need to pinpoint bugs.

Abithan

For this deliverable's code review we kept along with our regular strategy of reviewing PRs from user stories to master, but we also did a grand code review as a team together at the end of sprint 6. We all got together face to face, and went through files where team members expressed their concerns about code smells, unnecessary methods, dead code, and in general how to improve readability. As each member brought out their concern, we wrote them down in a text file and later assigned a member to each refactor. This was effective as we discussed each refactor as a team already, so it is easy to pick up the task and refactor. Personally, some of the code smells I encountered were long if else statements, but some of it was unavoidable as they had to do with enum classes, so polymorphism couldn't save the day. Furthermore, there were some unnecessary getter and setter methods as they did not contain any logic, and we as a team decided that it would just be better to make the fields public. Overall, the team worked well together in identifying code smells, and it was a great learning experience.

Matthew

For the final deliverable, during our task implementation we each resolved any code review comments we had from the previous ones, and continued to carry forward the pieces of advice received. We also sat together as a team on Thursday, the end of our sprint, and went through the entire project as a team and each made comments on things needing to be fixed for the final release. Some things I noticed were Enums still not being used everywhere they could be, if statements that could be replaced with ternary statements, and an outdated delete method within the dbHandler that could be improved by utilizing other existing helper methods, reducing duplicate code. We then spent the next three days implementing these changes ensuring our final product is complete. I was responsible for adding new Enums to cleanup the duplicate client verification, as there were lots of hard-coded strings in many files. In doing so, I also found a bug within the date formatter and fixed that aswell. Overall, the changes found during the review were all very minor, and the team came together to find them and solution our approach to resolving each issue.

How To Build and Run the Application:

To compile the application, execute mvn compile within the NewcomerReportServices directory.

To run the application, run the Main class located in NewcomerReportServices/src/main/java/ui. Currently, this has been tested to work when running through Eclipse, IntelliJ, and the command line.

System Requirements: To run the application, it is required that you have a JDK version of 10 of less.

How to Test the Application:

To run the unit/integration tests, execute mvn test within the NewcomerReportServices directory.

To go through the acceptance/production tests, follow the steps below in the acceptance test appendix for the individual use cases.

Final Product Demo:

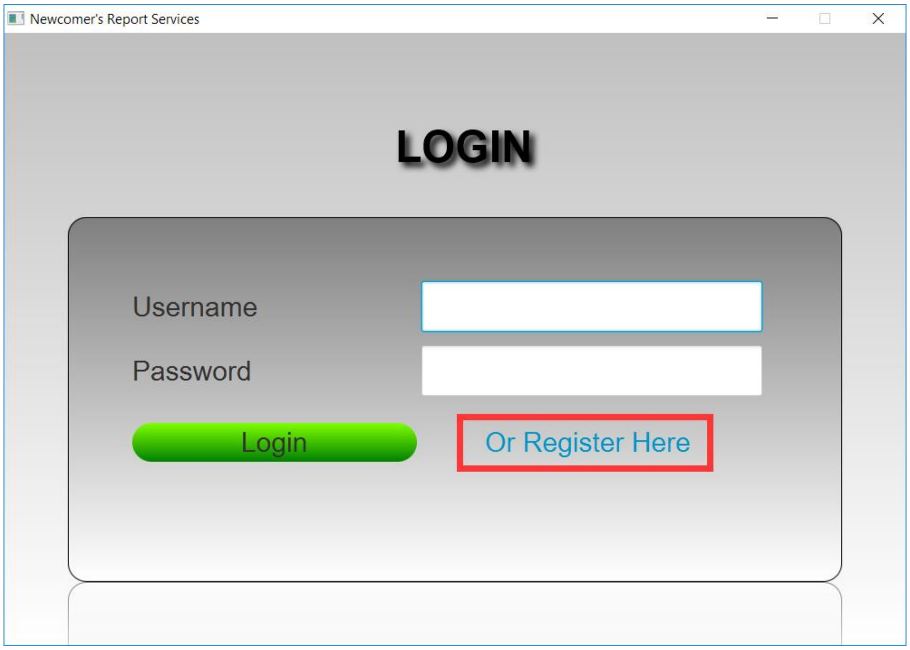
https://www.youtube.com/watch?v=_rxiDKTua7A&fbclid=IwAR3gA2tSFujsB6XKKVpIa6KDRJtxgsF4z_Fkf5UA5OkWzD7HmLzBTip8ZAo

Acceptance Test Appendix:

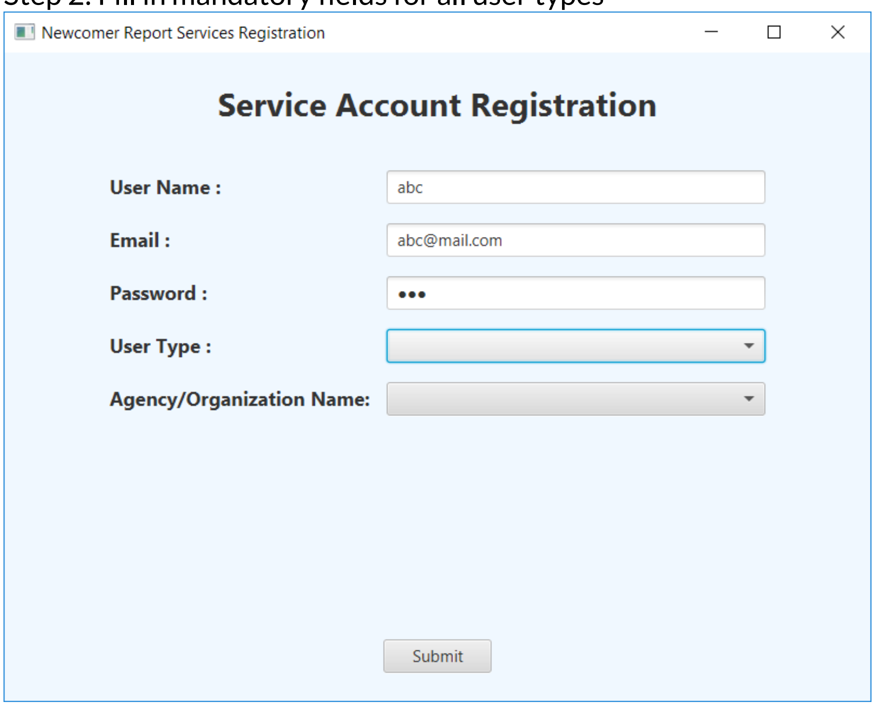
Validating User Registration

1. Register as an AGENCY user

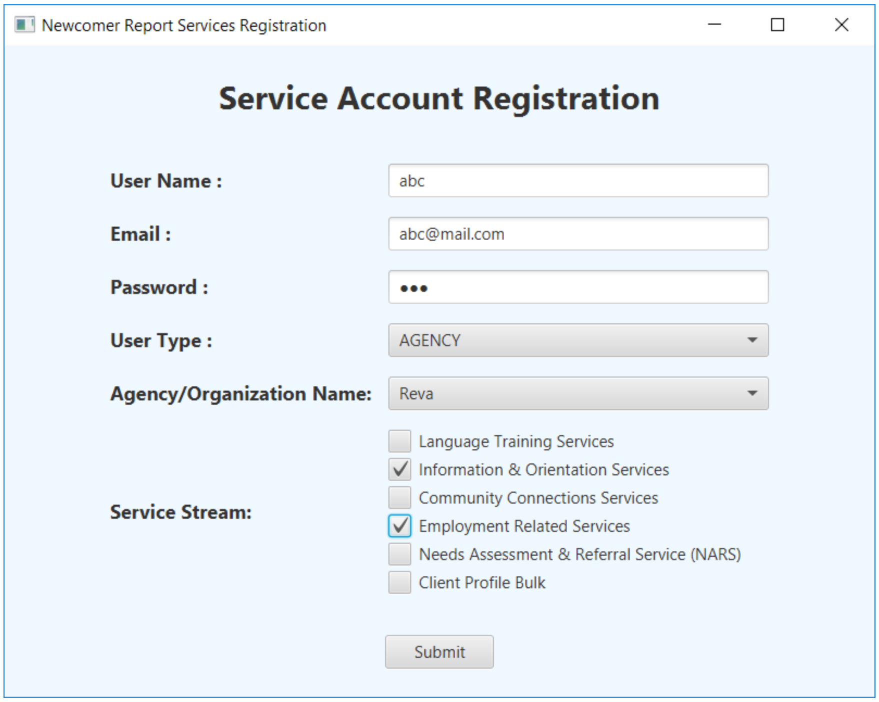
Step 1: Click “Or Register Here” link to bring up registration form page



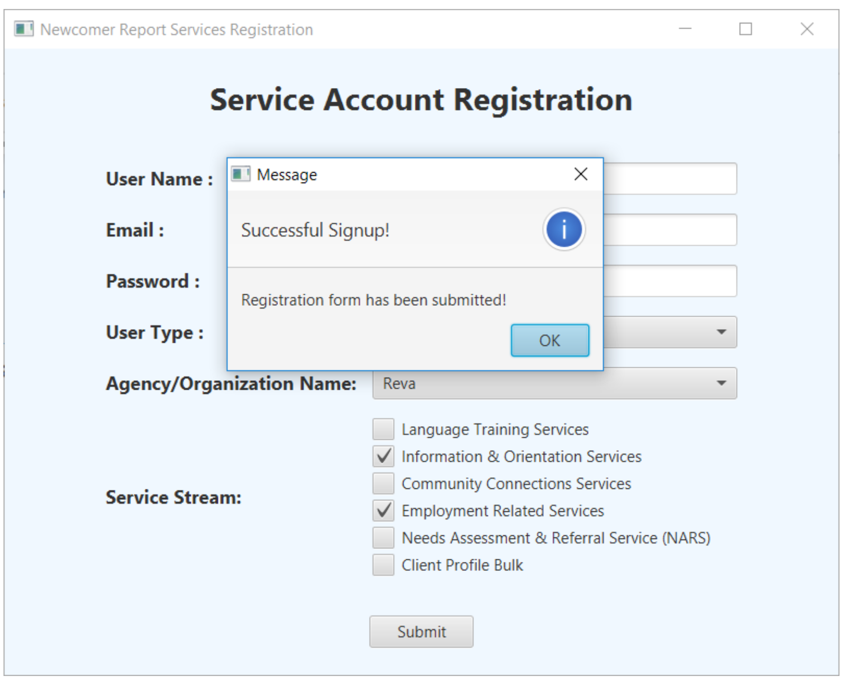
Step 2: Fill in mandatory fields for all user types



Step 3: Select AGENCY as user type to get access to service stream selection fields, choose agency name and single/multiple service streams



Step 4: Submit and confirm that the registration is successful



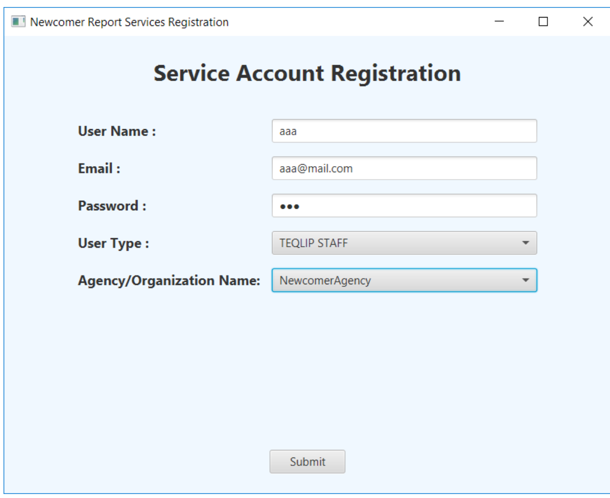
Step 5: Click “OK” to return to the Login page

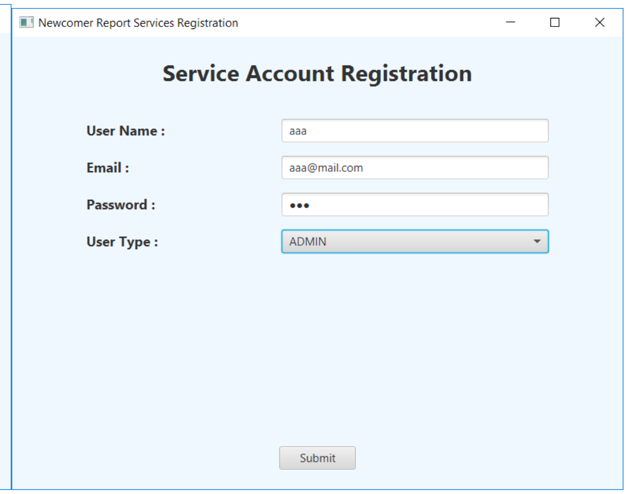
Step 6: Go into database to confirm that the user has been inserted

ID	UserType	Username	Password	OrganizationID	Email	EmploymentServ...
13	AGENCY	abc	abc	0	abc@mail.com	TRUE

2. Register as an ADMIN/TEQLIP STAFF

Step 1: Click “Or Register Here” link to bring up registration form page (same as above)
Step 2: Fill in mandatory fields for all user types (same as above)
Step 3: Select TEQLIP STAFF as user type and choose organization name from dropdown, or select ADMIN as user type and therefore no selection for agency/organization name

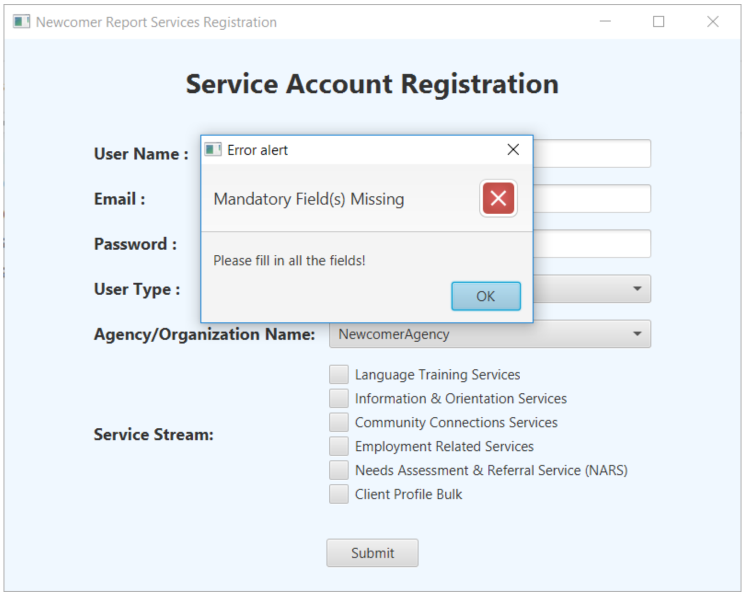




Step 4&5: Submit and confirm successful registration (same as above)

3. Missing mandatory fields when registering

Step 1: Click “Or Register Here” link to bring up registration form page (same as above)
Step 2: Fill in the registration form while leaving some mandatory fields empty(note that all the displaying fields are mandatory)
Step 3: Submit and confirm that registration fails



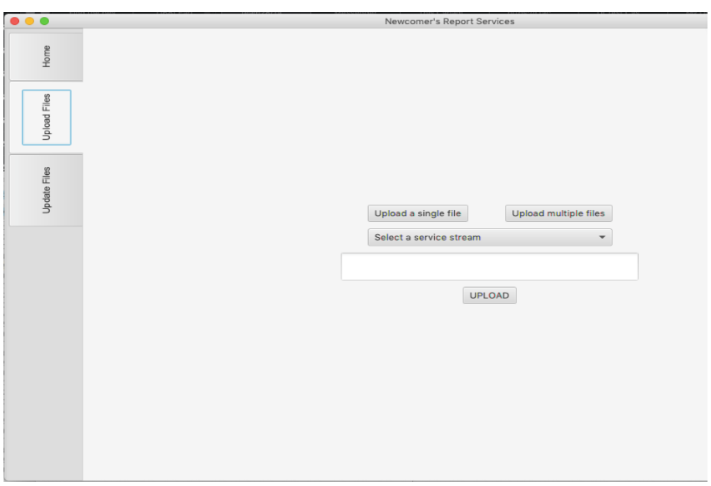
Step 4: Go into database to confirm that there is no user inserted with incomplete information (no user)

ID	UserType	Username	Password	OrganizationID	Email	EmploymentServ...
13	AGENCY	abc	abc	0	abc@mail.com	TRUE

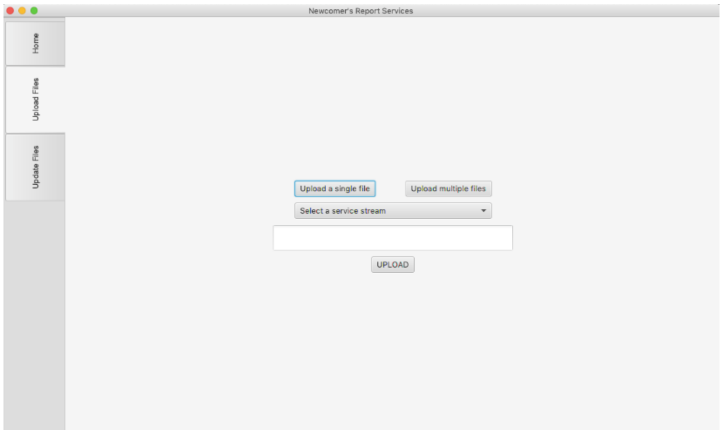
Validation of Uploading Content

1. Uploading a valid Single File

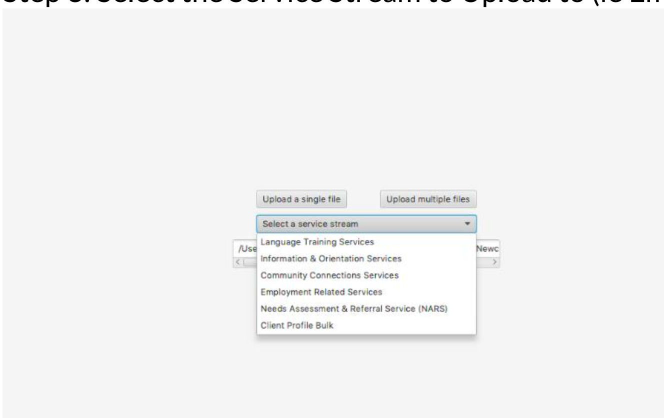
Step 1: Select the Upload Files Tab



Step 2: Open the File Chooser via Upload File and select a valid .xlsx file

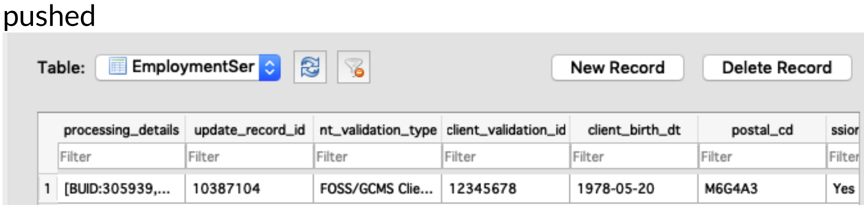


Step 3: Select the Service Stream to Upload to (ie Employment Service Stream)



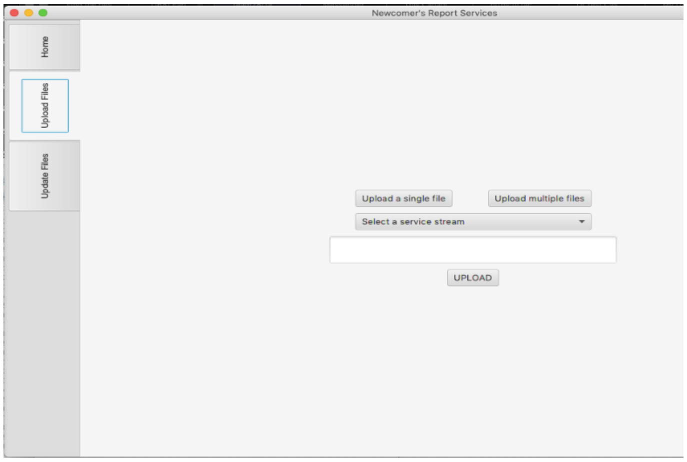
Step 4: Click Upload

Step 5: Check the EmploymentServiceStream table in the db to see if your file has been parsed and pushed

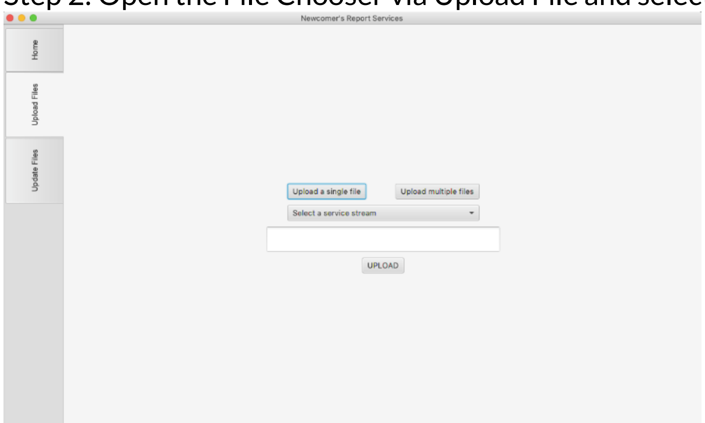


2. Uploading an invalid Single File

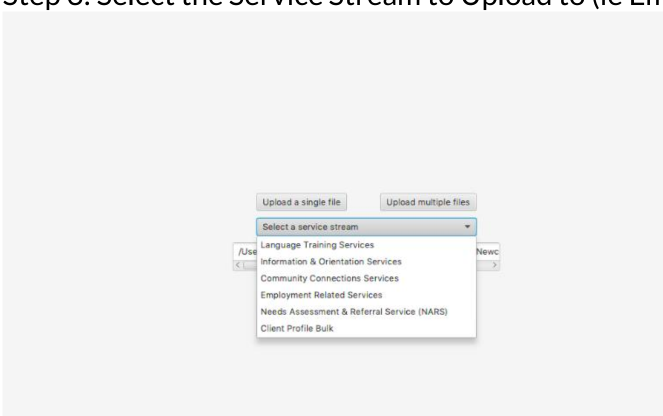
Step 1: Select the Upload Files Tab



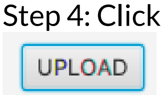
Step 2: Open the File Chooser via Upload File and select a valid .xlsx file



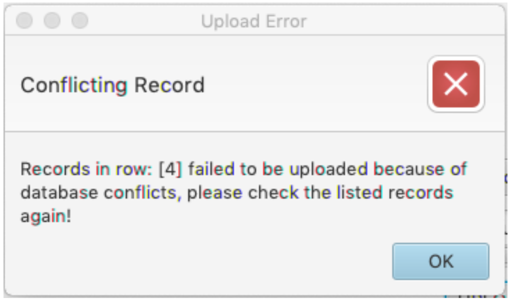
Step 3: Select the Service Stream to Upload to (ie Employment Service Stream)



Step 4: Click Upload



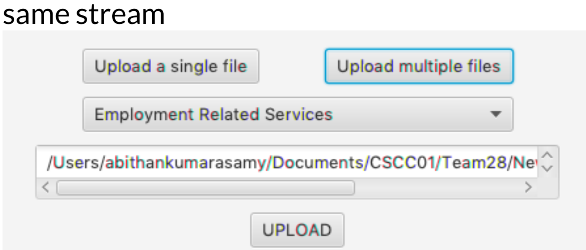
Step 5: You will be prompted with a dialog box informing you of your error, and you will have to reupload



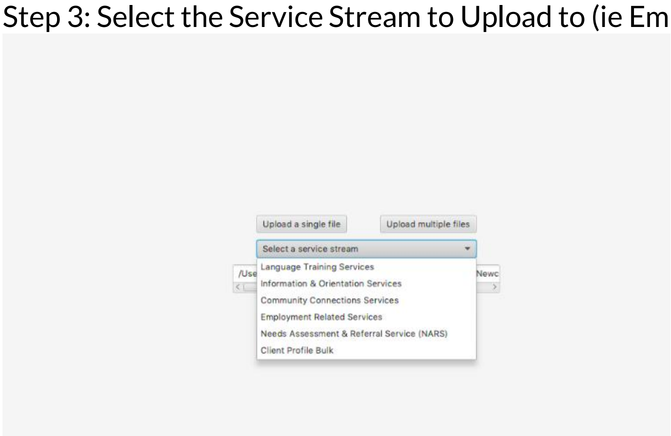
3. Uploading multiple files

Step 1: Select the Upload Files Tab

Step 2: Open the File Chooser via Upload Files and select multiple valid .xlsx file that belong to the same stream

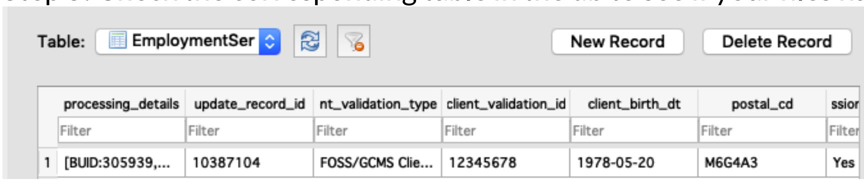


Step 3: Select the Service Stream to Upload to (ie Employment Service Stream)



Step 4: Click Upload

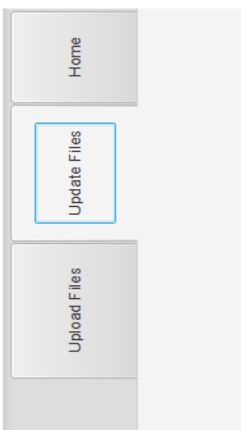
Step 5: Check the corresponding table in the db to see if your files have been parsed and pushed



Validating Update File Tab

Step 1: Login as an agency to the system

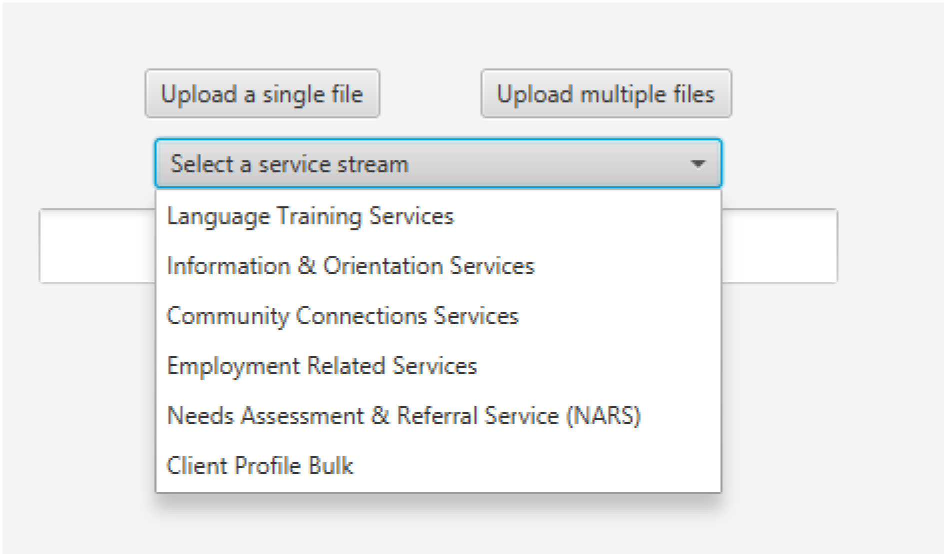
Step 2: Click on the Update Files tab



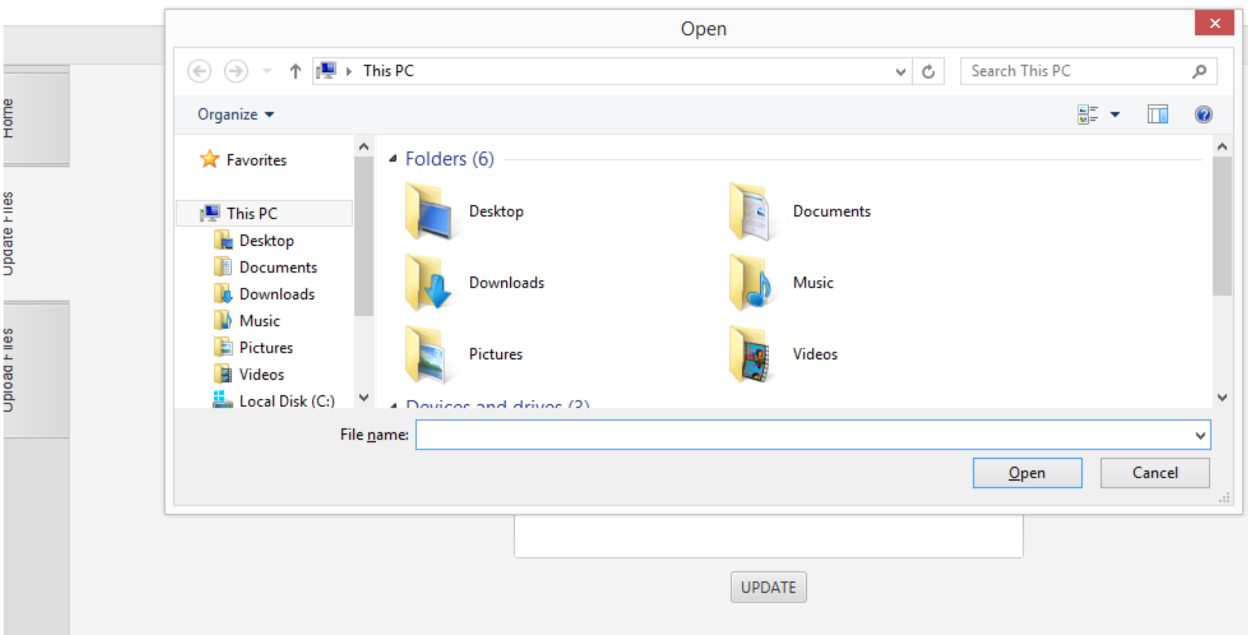
Step 3: Click on the Update Files tab

1. Updating a single file

Step 4: Select a Service Stream

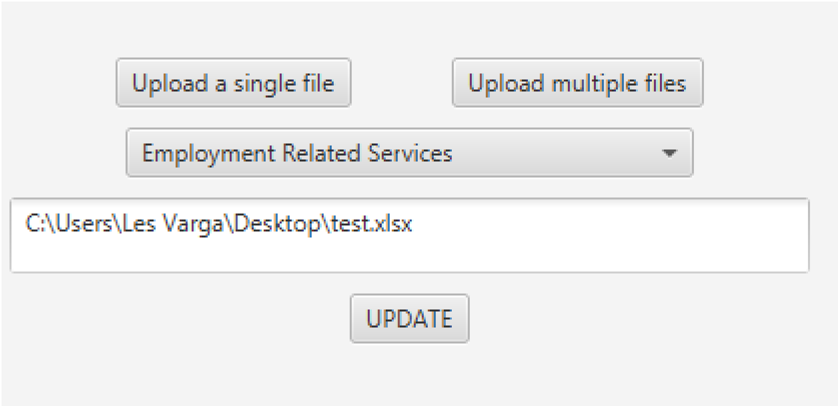


Step 4: Select Upload a single file
Step 5: Verify file selector pops-up

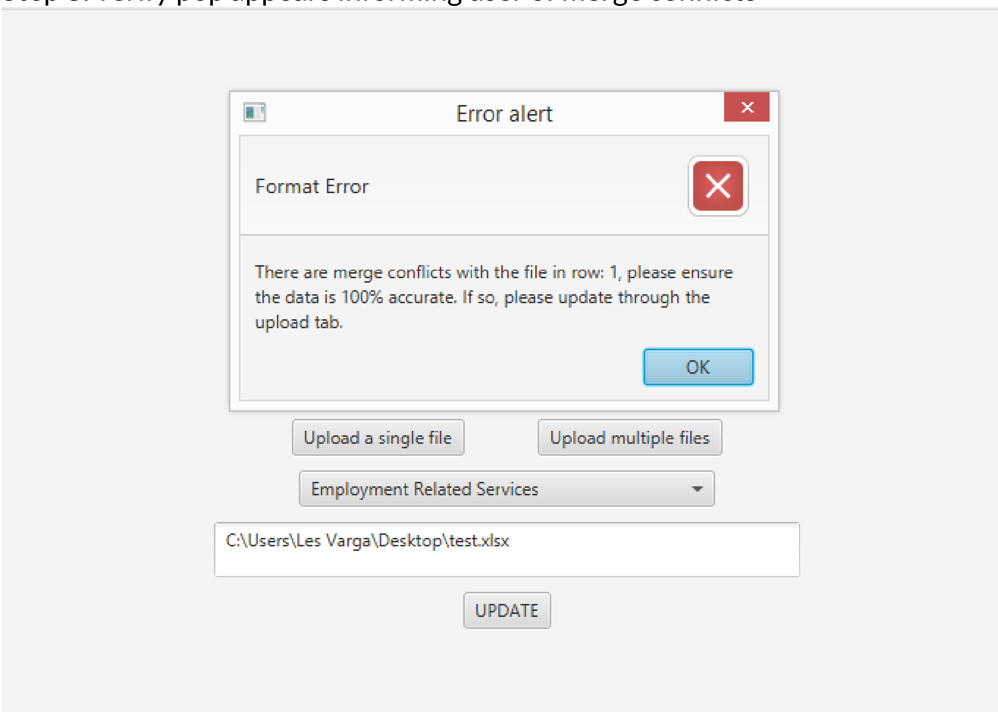


2. Selecting a valid file with merge conflicts

Step 6: select a valid file

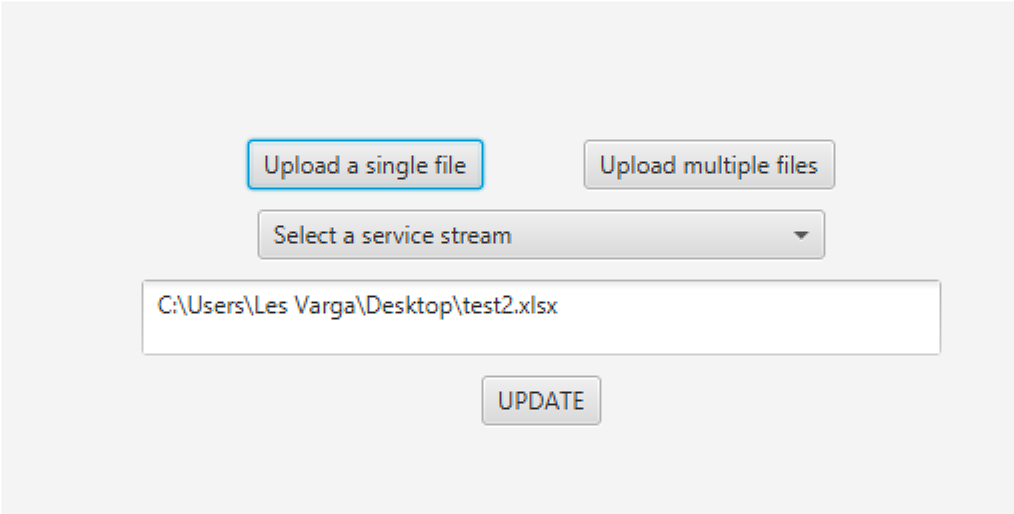


Step 7: select Update
Step 8: verify pop appears informing user of merge conflicts



3. Selecting a valid file with no merge conflicts

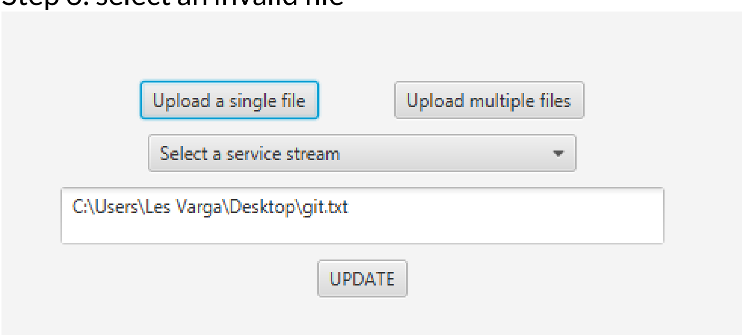
Step 6: select a valid file with no merge conflicts



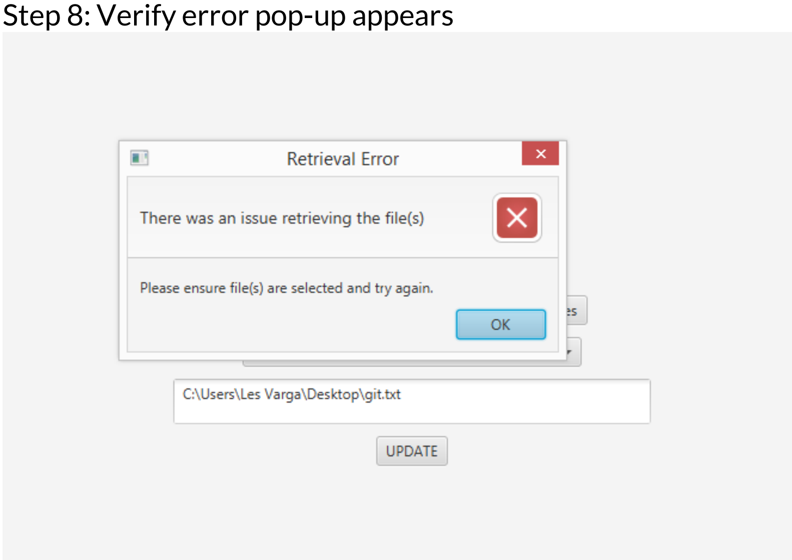
Step 7: select Update
Step 8: verify in DB that the file contents were properly uploaded

4. Selecting an invalid file

Step 6: select an invalid file



Step 7: select Update
Step 8: Verify error pop-up appears

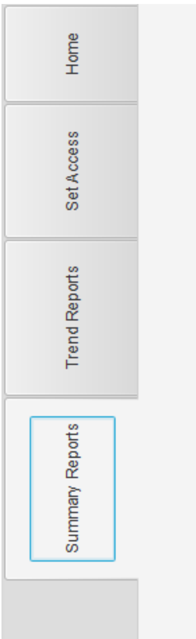


5. Updating multiple files
Repeat the above steps, but using the upload multiple files instead of upload single file

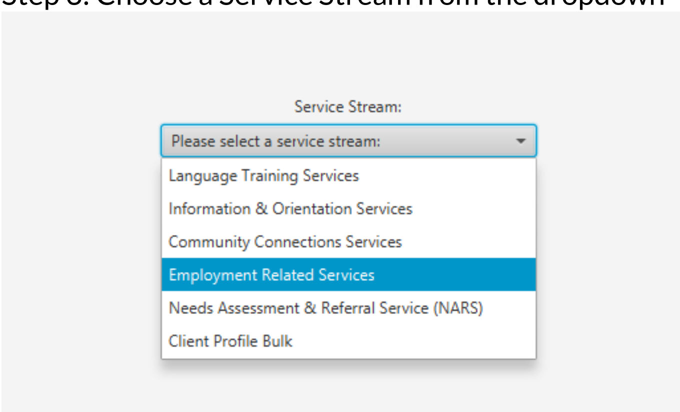
Validating Generating Summary Report

Step 1: Login as an admin to the system

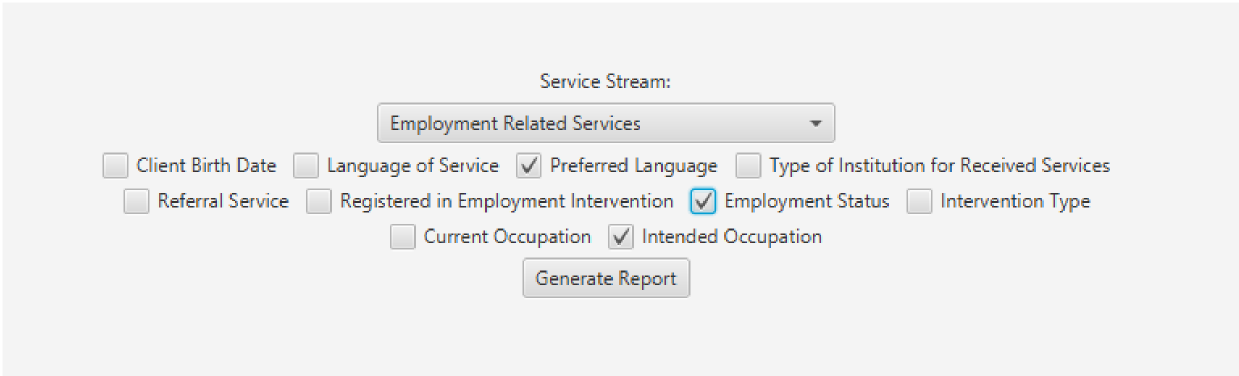
Step 2: Click on the Summary Reports tab



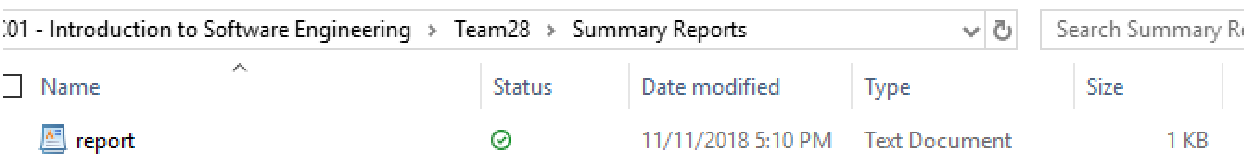
Step 3: Choose a Service Stream from the dropdown



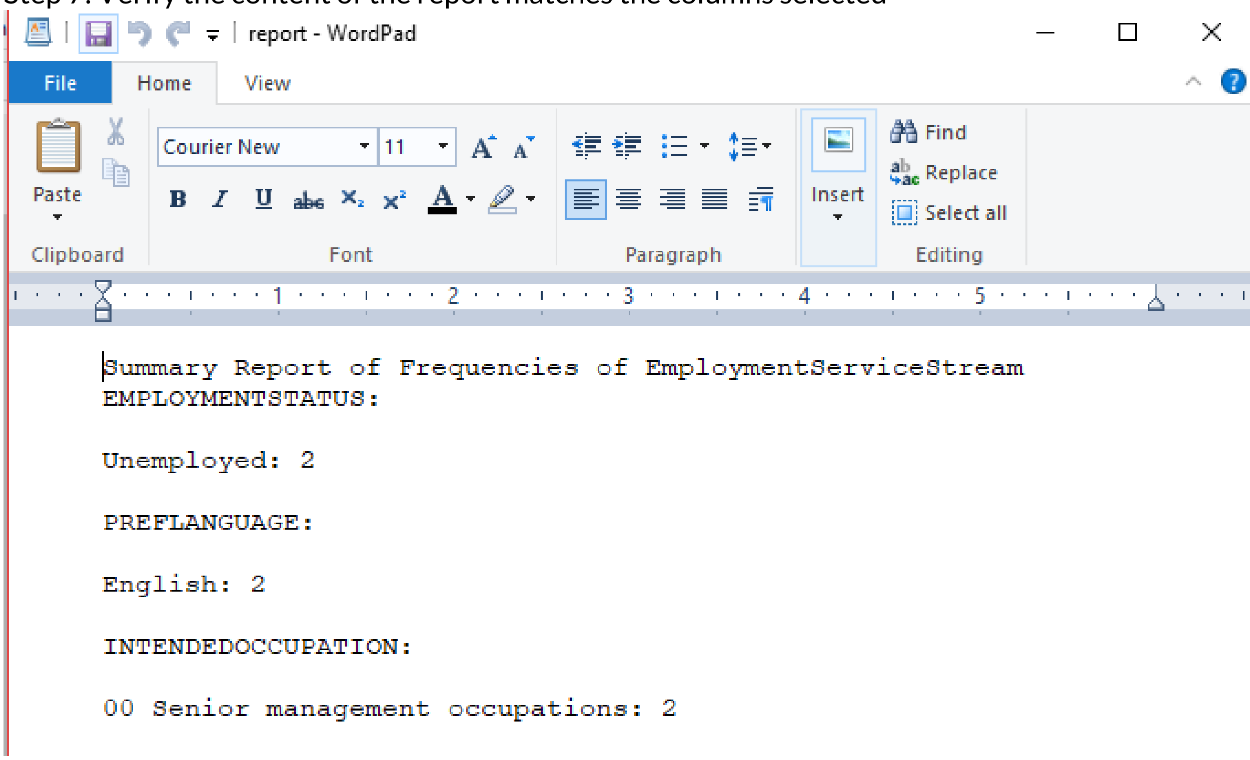
Step 4: Select the columns you would like to filter by



Step 5: Click the generate Report button
Step 6: Verify if report.txt file exists in Summary Reports folder in project root



Step 7: Verify the content of the report matches the columns selected



Step 8: Verify that the graphs were generated by going into ../Summary_Reports/ There should be a pdf file for all the bar charts and another pdf with all the pie charts.

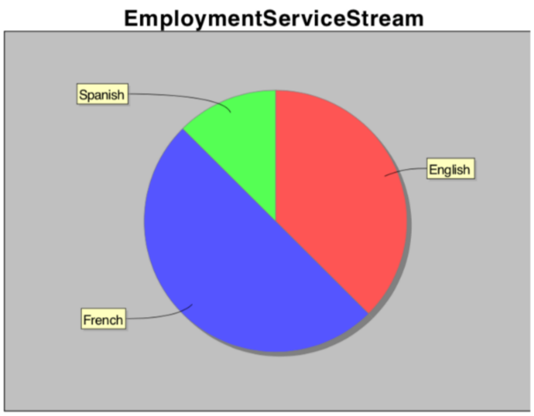
A sample pie chart looks as follows:

Summary Graph of SERVICELANGUAGE

English: 3

French: 4

Spanish: 1



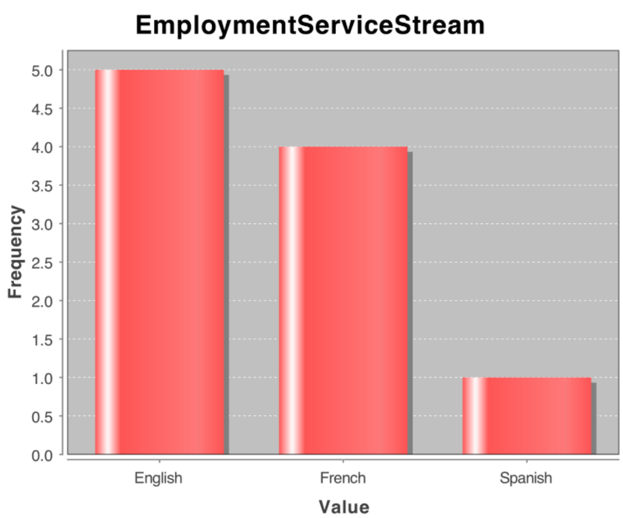
A sample bar chart looks as follows:

PREFLANGUAGE

English: 5

French: 4

Spanish: 1



The key observances are that the data in the db should coincide with the data in the graphs as well as the data in report.txt, which is the raw data without any graphs

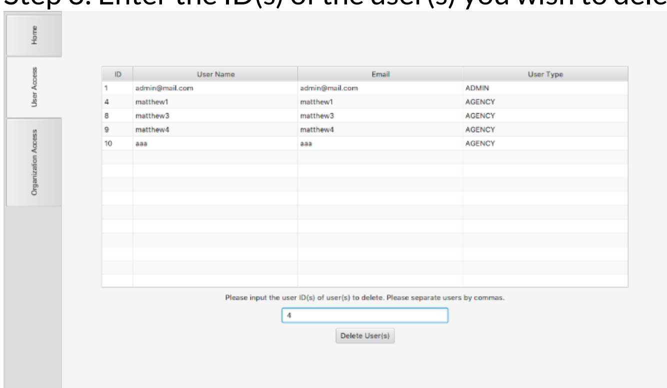
Validating Admin Functions

1. Deleting a User

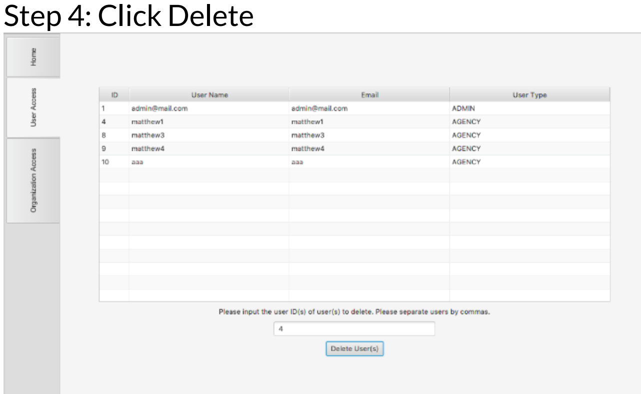
Step 1: Login as an admin to the system

Step 2: Click on the User access tab

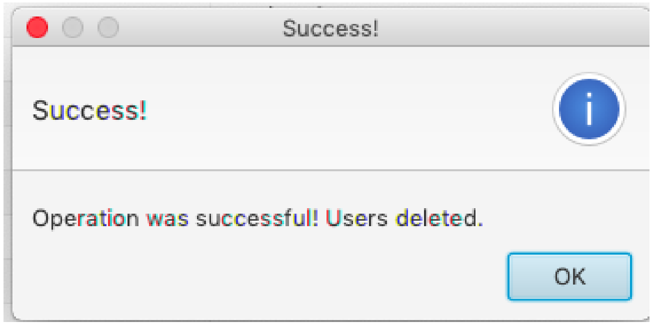
Step 3: Enter the ID(s) of the user(s) you wish to delete



Step 4: Click Delete

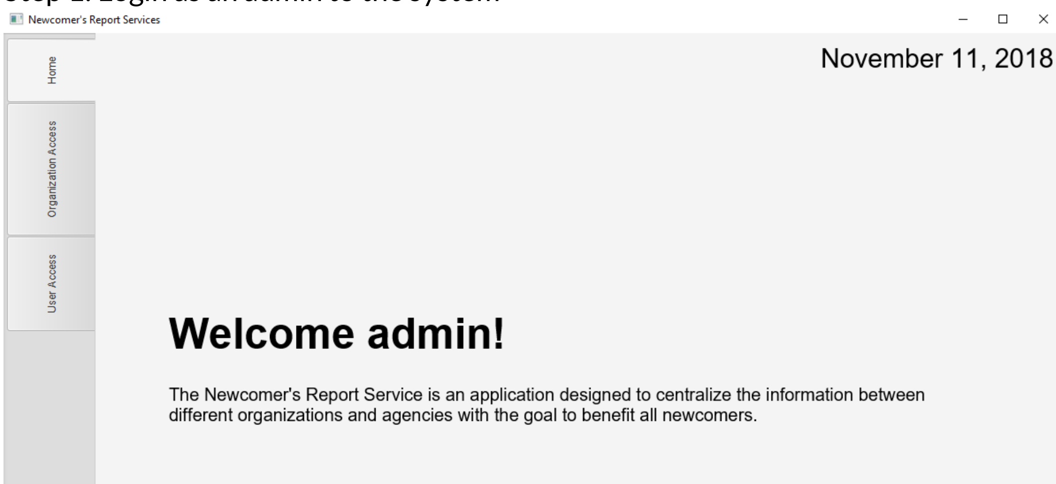


Step 5: Verify in the table above that the user has been deleted

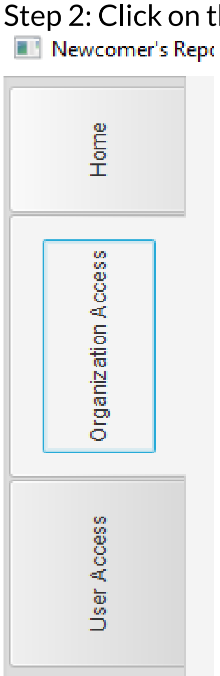


2. Inserting Organization

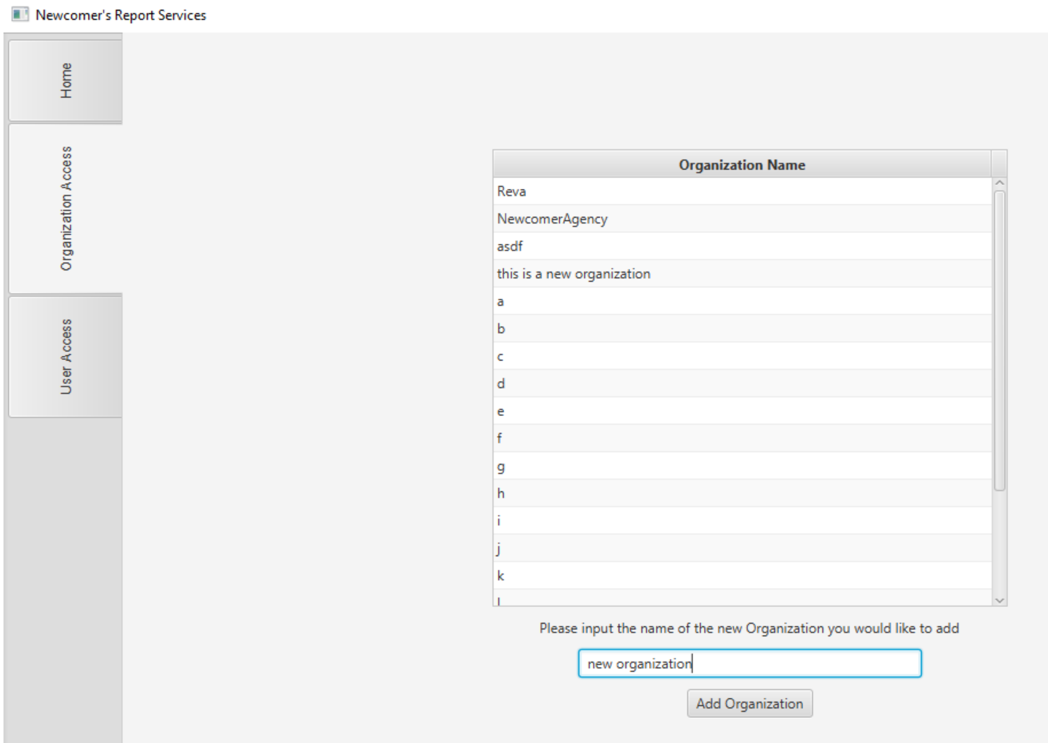
Step 1: Login as an admin to the system



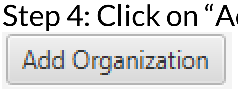
Step 2: Click on the Organization Access tab



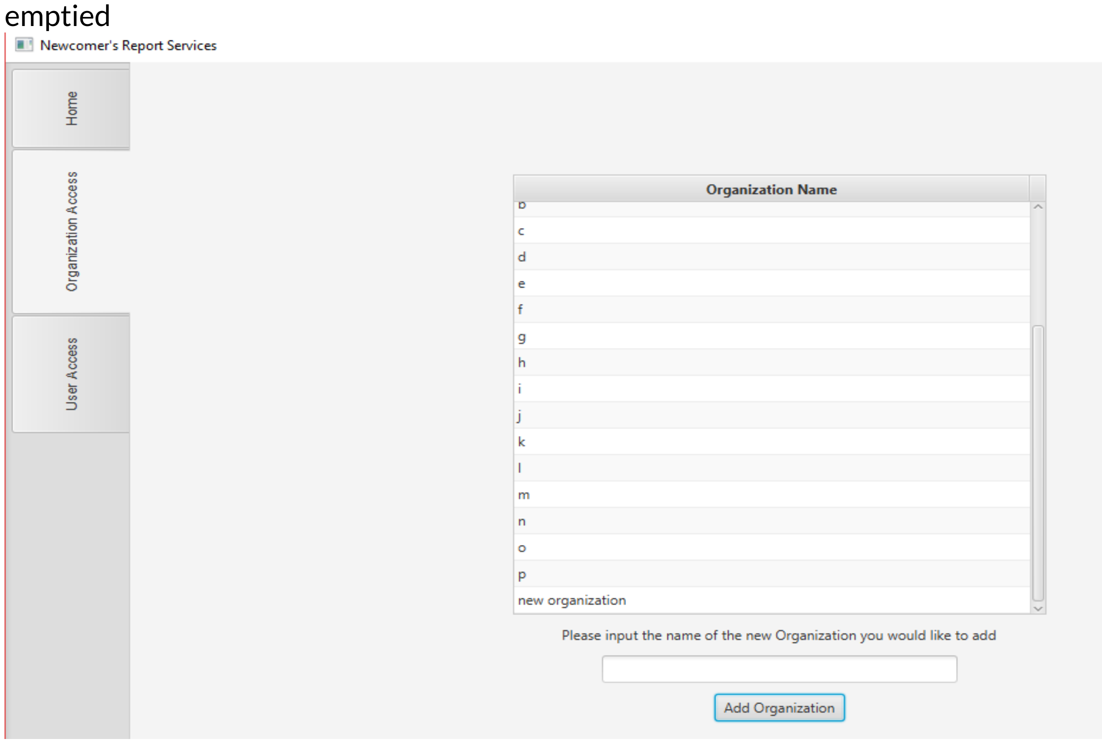
Step 3: Enter the name of an organization you would like to add in the text field



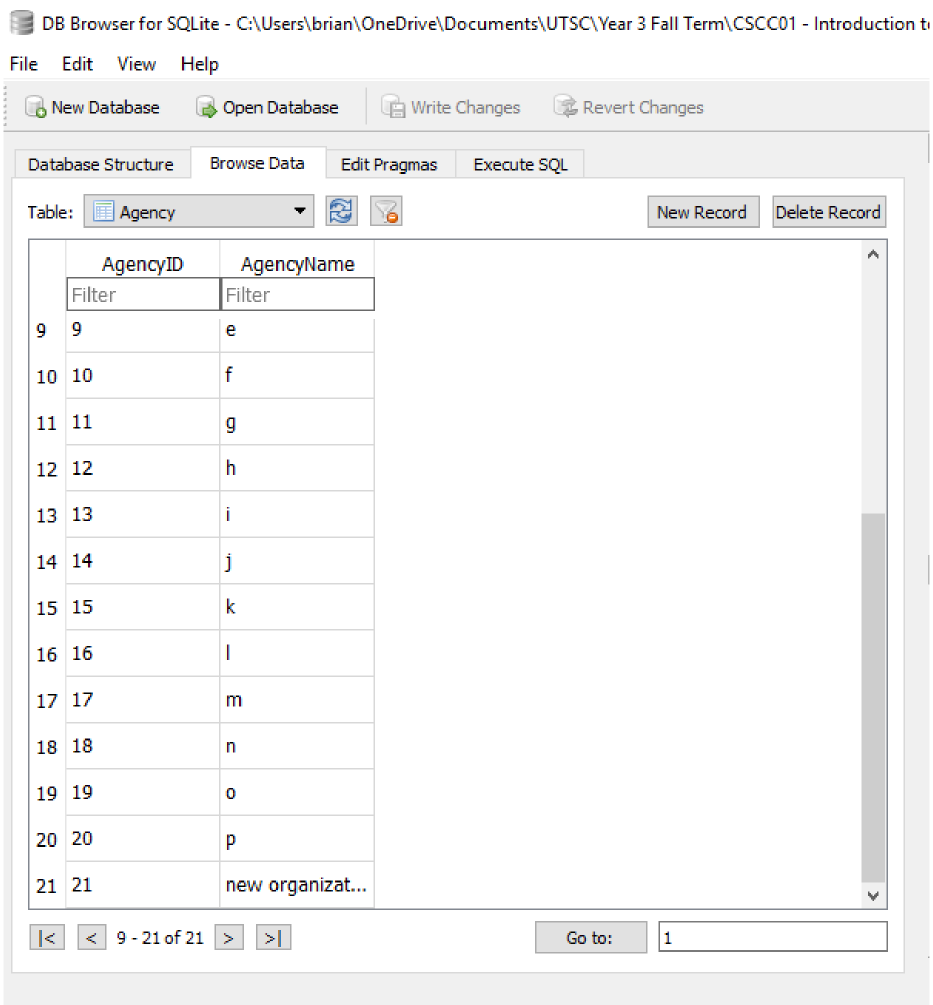
Step 4: Click on "Add Organization button"



Step 5: Verify that organization is added to the bottom of the table and that the text field is emptied

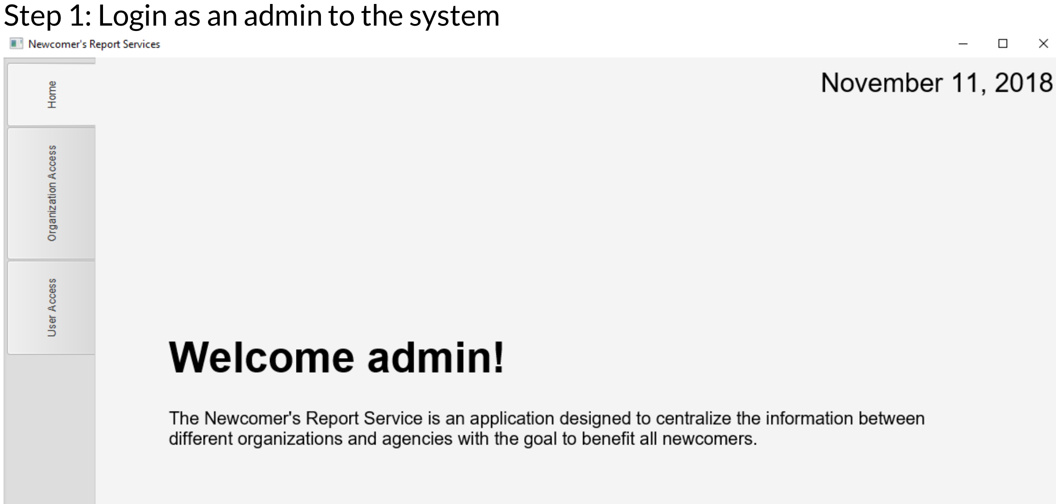


Step 6: Verify that the organization is added in the database

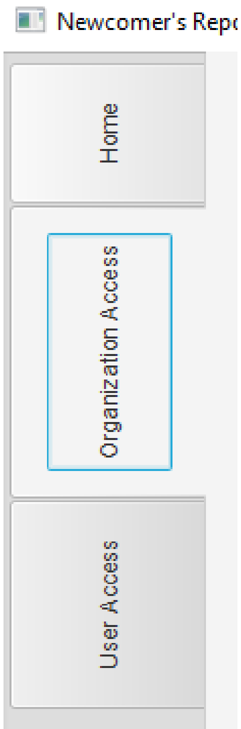


3. Inserting Organization that already exists

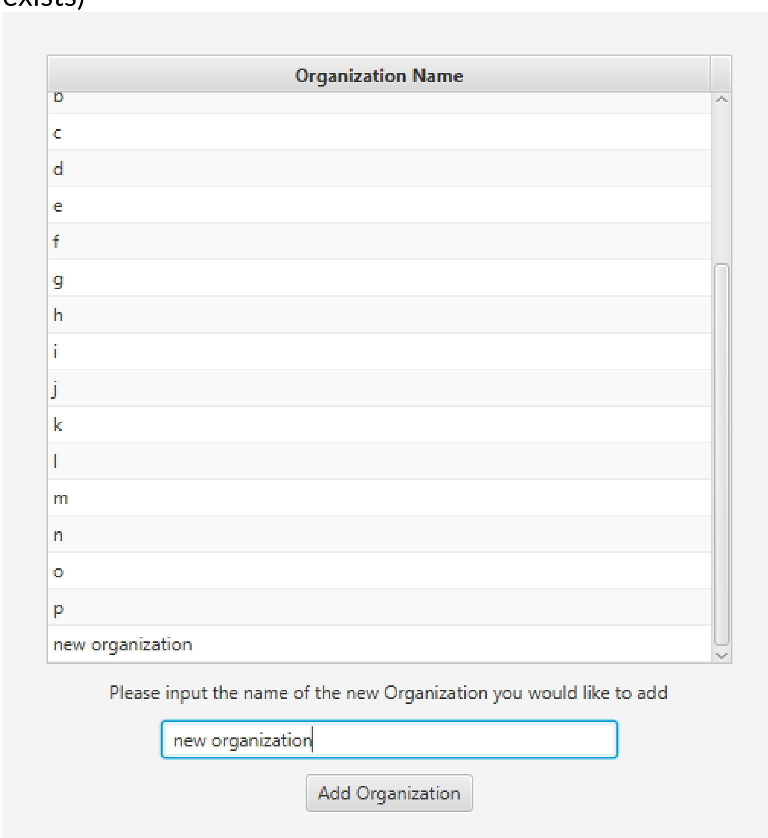
Step 1: Login as an admin to the system



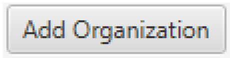
Step 2: Click on the Organization Access tab



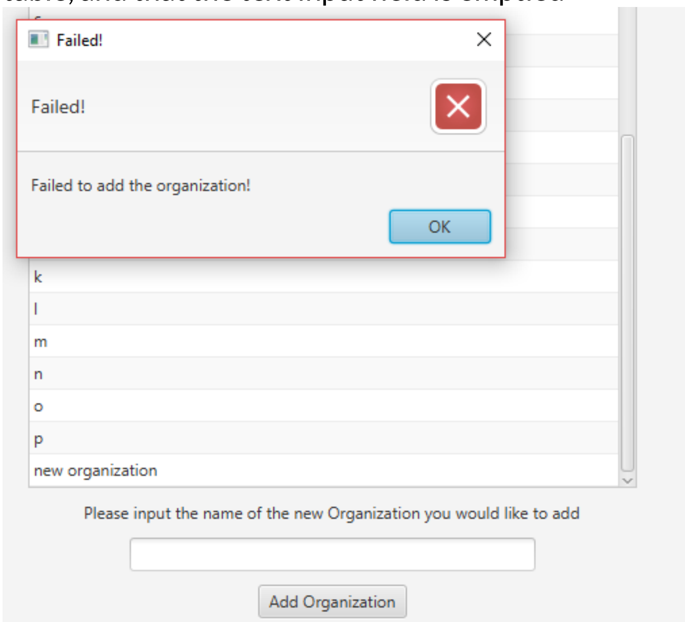
Step 3: Enter the name of an organization you would like to add in the text field (that already exists)



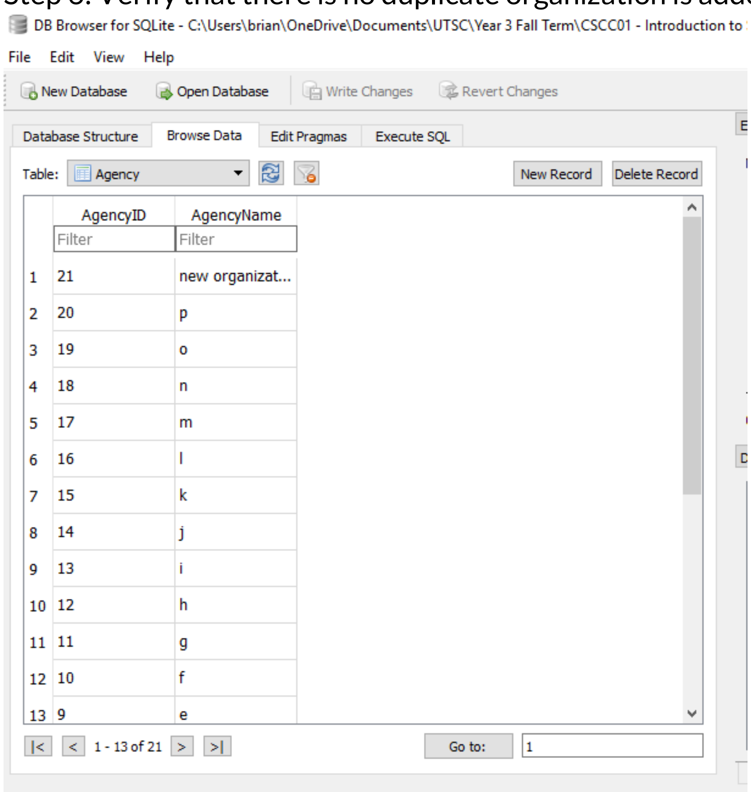
Step 4: Click on "Add Organization button"



Step 5: Verify that failed dialogue appears on screen, there is no duplicate organization in the table, and that the text input field is emptied



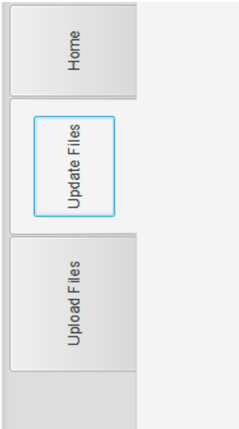
Step 6: Verify that there is no duplicate organization is added in the database



Validating Client Info Upload

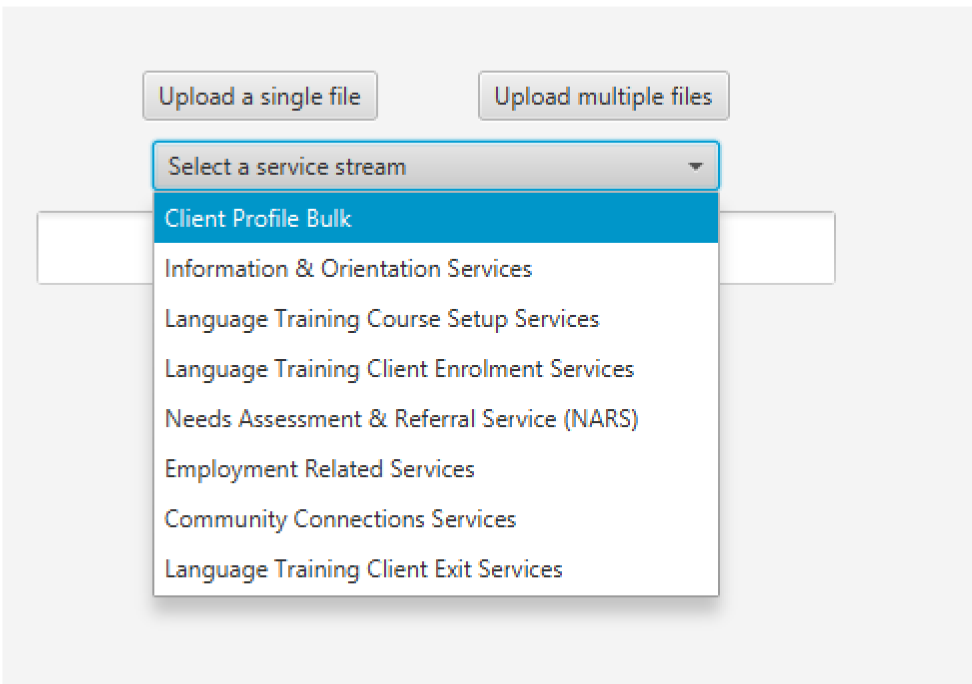
Step 1: Log in as Agency User

Step 2: Navigate to upload file tab



Step 3: choose a file with client profiles

Step 4: select client profile bulk from serve stream list



Step 4: press UPLOAD

If adding new clients for first time, check that no error message appears, but a success message pops up.

If adding same client with different ID, check that duplicate client message pops up

If duplicating a client, check that duplicate client message appears.

Validating File Upload Requires Clients to Exist

Step 1: log in as agency

Step 2: select Upload tab
Step 3: select file and service stream you wish to upload to
If trying to upload file for client that is not in system, check that popup occurs, letting you know no client exists with the given id and the record is not uploaded

If trying to upload for client that is in system, ensure that no error message pops up, and that the4 success message appears.