CSCC01

Project Deliverable #4

Java Bean (Team28)

Brian Chim

Abithan Kumarasamy

Matthew Varga

Maggie Zhou

Table of Contents

Note: Personas were not updated and so have not been included

Sprint 03 Task Breakdown:

<u>User Story 2:</u>

As Alice (an Agency Representative), I would like to upload an iCare excel file, so that the user's information is stored securely and I do not have to worry about losing my local copy.

Task 13 – insert parsed info into the database

- Story Points: 2; dependency: Task 8, Task 11, Task 12
- Integrate the excel parser with the database and the UI
- In UI, you should be able to choose the file to parse
- The excel parser parses each row and stores it into a hashmap
- Using the list of hashmaps, call the insert method in dbhandler to push the database

<u>User Story 3:</u>

As Ellie (an Agency Analyst), I would like for the application to let me know when it was unable to merge a field, so that I can fix the issue and reupload.

Task 15 - Add upload button with error checking

- Story Points: 5; Dependencies: Task 13
- add update tab to user section
- in the tab, include everything the upload tab has except add a popup when there are merge conflicts
- also add a popup when user tries to submit when no file selected

Task 16 – Add backend upload logic with error checking logic

- Story Points: 7; Dependencies: Task 14
- Add a safeUploader class to util
- Add a selectRow method to dbhandler for 'WHERE' clause query + unit tests
- Check client id after parsing the uploaded file(call parser)
- Record conflicting rows and insert the other records to database

Task 17 – Connect upload logic to the UI

- Story Points: 1; Dependencies: Task 16
- Note: this was descoped in our execution as the team decided it was a duplicate task of 13.

User Story 4:

As David (CCS organization manager), I would like to generate reports (summary), so that I can use the report to improve future newcomer service.

Task 18 – Create a generic select method in the database handler

- Story Points: 1; Dependencies: None
- create a generic select method in db handler
- should be able to pass in a list of cols you would like to select and a table name to select them from
- passing * selects all columns

Task 19 – Create a Generate Summary Reports Tab

- Story Points: 5; Dependencies: None
- Should support Employment Service Stream and Needs Assessment and Referral Services stream
- All other streams should be an option but user is not allowed to generate report for those
- Create an enum of approx. 10 query params for the reports
- Queries should be displayed as checkboxes (if user wants that field in their report then select that box)

Task 20 – Call DBHandler Select Method to generate client report and output results in .txt file

- Story Points: 2; Dependencies: Task 18
- Generate the first type of report that we support (a summary report)
- This report will simply be a frequency count of how many clients share the same column field value in the data
- A list of column names will be passed in
- Will need to search through the respective columns and count how many unique values there are and their frequency
- Then a report will be formatted as a string with new lines and formatting, and will be returned

Sprint 03 Planning and Execution

Sprint 03 Plan									
User Stories	Tasks	Dependency	Story Points		Tuesday	Wednesday	Thursday	Friday	Saturday
2	13	T8,T11, T12	2		A:2				
3	15	T13	5			MV:3	MV:2		
3	16	T14	7		MZ:3	MZ:4			
3	17	T16	1				MV:1		
4	18		1				MV:1		
4	19		5			B:2		B:3	
4	20	T18	2						A:2

Sprint 03 Execution										
User Stories	Tasks	Dependency	Story Points		Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
2	13	T8,T11, T12	2		A:2					
3	15	T13	5			MV:1				
3	16	T14	7		MZ:2			MZ:5		
3	17- Ended up being duplicate task	T16	1							
4	18		1			MV:3				
4	19		5			B:2		B:4		
4	20	T18	2							A:4

A: Abithan Kumarasamy

B: Brian Chim

MV: Matthew Varga

MZ: Maggie Zhou

Progress Reports:

Tuesday:

- Sprint progressing as expected.

Wednesday:

- Task 15 completed ahead of schedule.
- Task 16 delayed due to unforeseen circumstances. Re-scheduled for completion on Friday.
- Task 17 was removed as team decided it was a duplicate of Task 13
- Task 18 completed ahead of schedule as a result of Task 15's early completion

Thursday:

- Sprint progressing as expected.
- Note: Empty because of Task 15 and Task 18 completion on Wednesday, as well as Task 17 descoping

Friday:

- Sprint progressing as expected.
- Note: Task 16 was re-scheduled for Friday completion after Wednesday's delay.

Saturday:

- Task 20 delayed as there required more work and it was taking longer than expected.

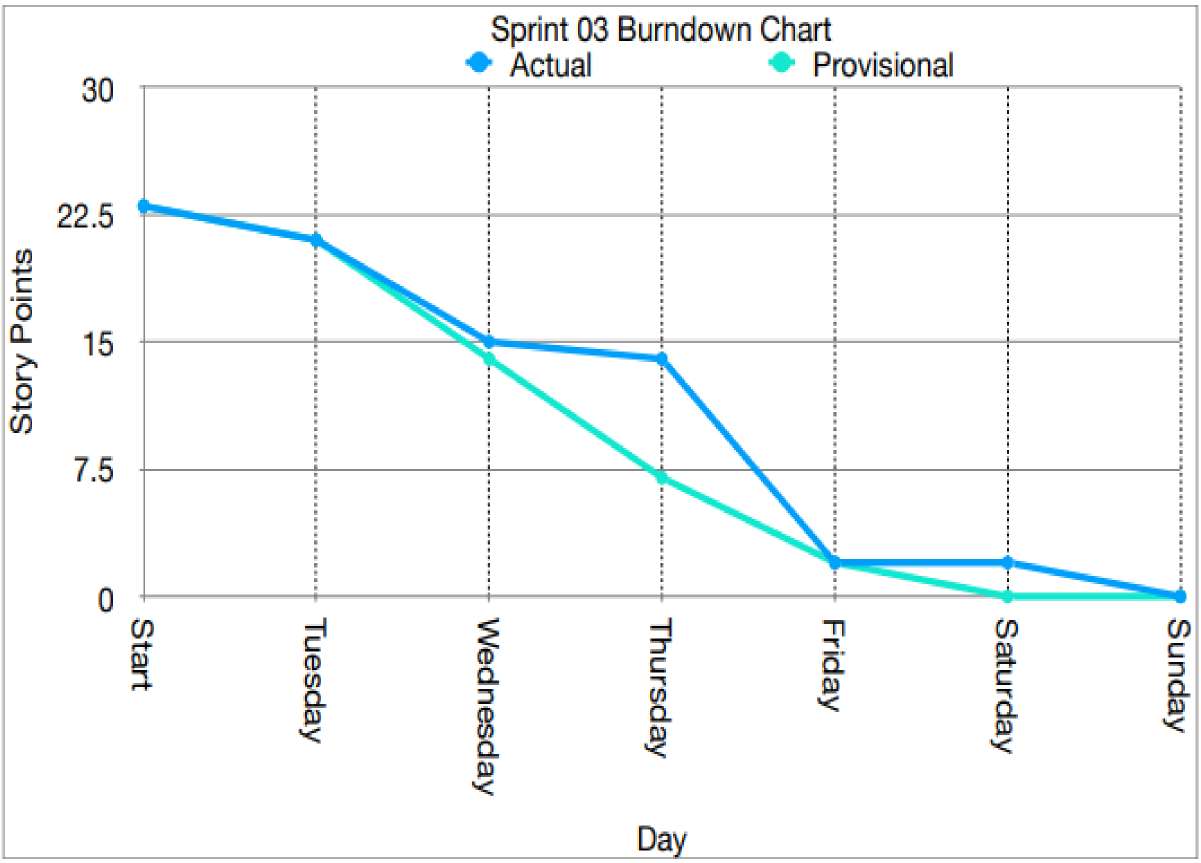
Sunday:

- All tasks for sprint completed as expected

Sprint 03 Burndown

Sprint 3

	Actual	Provisional		
Start	23	23		
Tuesday	21	21		
Wednesday	15	14		
Thursday	14	7		
Friday	2	2		
Saturday	2	0		
Sunday	0	0		



Sprint 04 Task Breakdown:

User Story 4:

As David (CCS organization manager), I would like to generate reports (summary), so that I can use the report to improve future newcomer service.

Task 21 – Connect generate report button with outputting the result

- Story Points: 3; Dependencies: None
- call the db handler with the selected columns from the ui to gather a list of records filtered by those fields.
- format the result nicely and output in a text file
- add testing strategy documentation

User Story 5:

As Charlie (an Admin) I want to be able to remove a certain user from the application or add a new agency to the application, as the users of the applications may have changed.

Task 22 – Add generic delete method to the dbhandler

- Story Points: 2; Dependencies: None
- Need to create a delete method for the dbhandler
- This method needs to be generic enough to be able to delete users, organizations, or client info in a table
- The method will be passed in a table name, and the data for a WHERE clause
- Parameters - String tableName, String primaryKey, String Value

Task 23 – Add admin tab to admin users

- Story Points: 1; Dependencies: None
- Add a tab for deleting user
- Add a tab for adding organization
- Modify admin access permissions corresponding to new tabs

Task 24 – Add a table of users to the set user access (admin) tab

- Story Points: 4; Dependencies: Task 23
- Get all users from database and filter all non-admin users
- Create a table based on the user lists to display id, username, email, user type
- Update table view after a user is successfully deleted

Task 25 – Add text input field to set user access (admin) tab and (non-functional) delete button

- Story Points: 2; Dependencies: Task 23
- Field should have some prompt text as well as a label for instructions
- Should support deleting multiple user ids (if the user specifies in the input box)

Task 26 – Connect the backend logic for delete user with the front end

- Story Points: 2; Dependencies: Task 22, Task 24
- Connect the delete method in the dbhandler with the frontend for deleting users
- The user should be able to pass in a list of user IDs that appear in the table, and delete them from the system
- Error checking is needed to see if the user ID is in the db
- Integration between front end and backend requires manual documentation of testing

Task 27 – Create the UI for the set organization access tab for admins

- Story Points: 2; Dependencies: Task 23
- Add a text input field
- Create an "add organization" button which connects with the value in the input field (does not have to be functional)

Task 28 – Connect the frontend for add organization with insert method in the backend

- Story Points: 1; Dependencies: Task 27
- Use DB handler's insert method to add the agency
- User should not be able to add an already existing organization

Sprint 04 Planning and Execution

Sprint 04 Plan

User Stories	Tasks	Dependency	Story Points		Tuesday	Wednesday	Thursday	Friday	Saturday
4	21		3			MV:3			
5	22		2		A:2				
5	23		1		MZ:1				
5	24	T23	4			MZ:1		MZ:3	
5	25	T23	2			B:2			
5	26	T24, T22	2						A:2
5	27	T25	2				MV:2		
5	28	T27	1					B:1	

Sprint 04 Execution

User Stories	Tasks	Dependency	Story Points		Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
4	21		3						MV:2	MV:1
5	22		2		A:1	A:1				
5	23		1		MZ:1					
5	24	T23	4					MZ:4		
5	25	T23	2			B:2				
5	26	T24, T22	2					MZ:1		
5	27	T25	2						MV:1	MV:1
5	28	T27	1							B:1

Progress Reports:

Tuesday:

- Slight delay on Task 22, pushed to Wednesday completion.

Wednesday:

- Task 21 pushed back due to other course's midterm. Rescheduled for Saturday.
- Task 24 pushed back to Friday because of other course work.

Thursday:

- Task 27 pushed back due to other course's midterm. Rescheduled for Sunday.

Friday:

- Task 28 pushed back to Sunday due to Task 27 delay.
- Task 26 completed earlier than expected and was taken over by another member as it was found necessary to test Task 24.

Saturday:

- Proceeded as expected after re-scheduling.

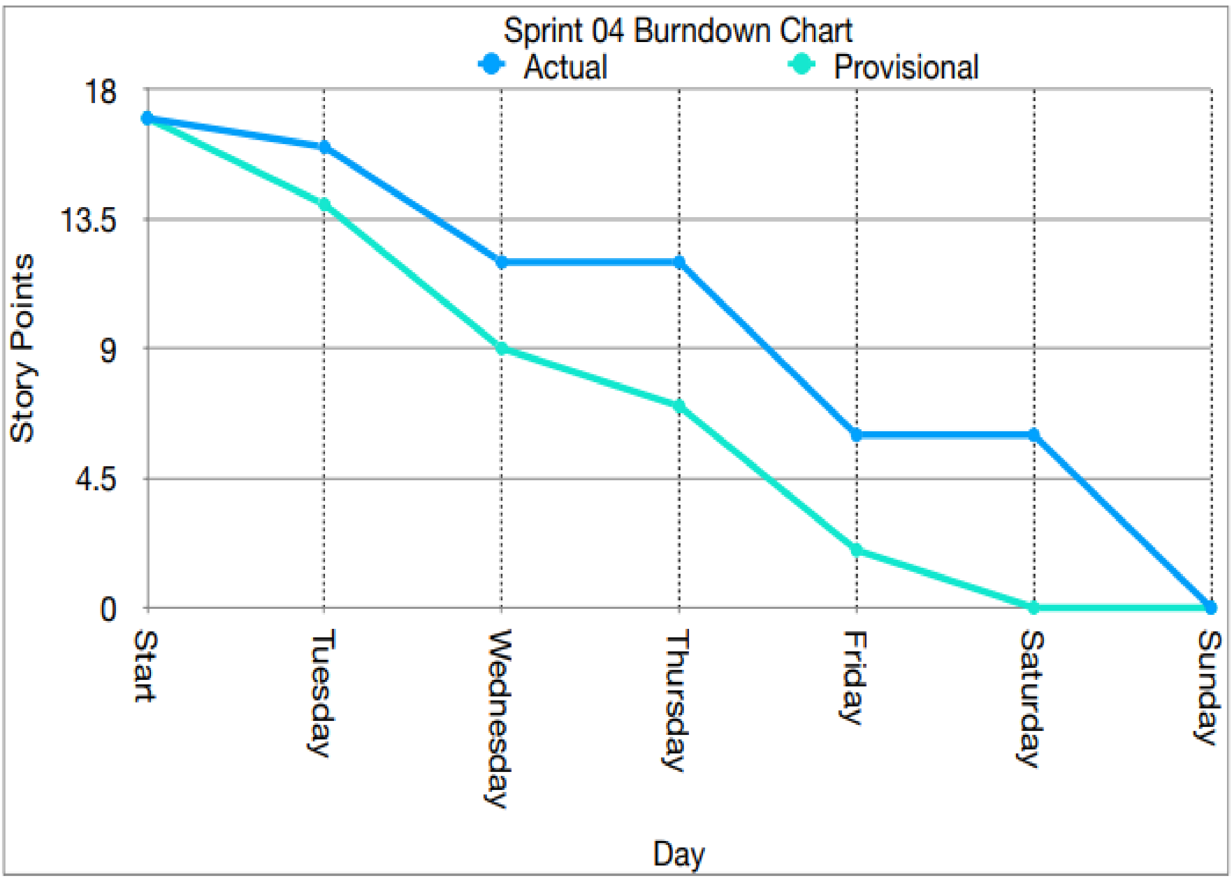
Sunday:

- All tasks for sprint completed as expected.

Sprint 04 Burndown

Sprint 4

	Actual	Provisional		
Start	17	17		
Tuesday	16	14		
Wednesday	12	9		
Thursday	12	7		
Friday	6	2		
Saturday	6	0		
Sunday	0	0		



Testing Strategy

In testing our application, our team has decided to use the approach of automating all unit and integration tests regarding the backend features. Each task will include 2-3 test cases to sufficiently declare the task as being done and working. These test cases should include at least one happy path (correct input leading to correct output) and at least one unhappy path (incorrect input leading to correct output) to satisfy that the program is correct.

Relative to the front end tasks, those will be tested alongside the acceptance tests. In this regard, our team will create at least one acceptance test encompassing the happy path of the user story to verify that the application is working correctly. This acceptance test will validate that our frontend is working cohesively with the backend.

Code Review Strategy

In our team, we have come to the decision that we will execute code review whenever we are ready to make a user story merge to the master branch.

In our code reviews, each member is required to have a look through what is being pushed to master and provide their constructive comments on what can be improved.

In general, the following is what each member will look out for:

- Dead Code
- Unused Imports
- Hard-Coded String literals
- Debugging messages (print statements)
- Proper JavaDocs
- Reasonable Unit Tests
  (approximately 2-3 tests per method, otherwise a written manual test case)
- Design Patterns
- Implementation
- Efficiency

Within the comments resulting from the code review, team members are to inform whether they still approve of the pull request or not. Given that there are improvements, the team member should inform the team on the priority of the improvement as part of the code review process.

The team's code review debriefing meeting can be viewed at the following link:

https://drive.google.com/file/d/1Jd-fTKkmWaIIy2dzFWUThaXaBniCGtPZ/view

Code Review Comments and Summaries

Brian

In our team, we have been conducting code review since the first sprint. As a result of that, I feel that generally our team has been following good practices. One of the things I've noticed in my code reviews is that more often that not, there are print statements littered throughout different methods which shouldn't be there. While not particularly devastating to our code, I feel like our team should be more vigilant in removing them prior to merge. Additionally, a point of note is the inconsistency between using hard-coded Strings and Enums. Though we had began using hard-coded Strings only at the beginning of the project, it has followed us up to user stories 4 and 5. We have been getting better at using Enums but at some point, we should go back and replace those early Strings with Enums to have consistency and avoid any small typing mistakes. Another place of general improvement that I noticed would be beneficial to our group is the way we handle documentation. For tests and even the README, while not directly related to the code, it helps to keep clarity and understanding within our team's work. As well, if we document better with comments or Javadoc for example, it would benefit our team in that we can follow the thought process and suggest better methods of coding.

Abithan

Each member of the team is assigned to code review the Pull Request (PR) from each user story branch to master branch. I will explain some of the issues that I found when reviewing the PR from User Story 4 and User Story 5 to master. Overall the code quality that we as a team have produced appears to be top notch in quality, and a lot of good practices and principles have been followed. However, there have been minor typos that have been called out. Also, I noticed that we could have potentially used a Factory pattern to generate our UI components, as currently we are using a huge switch statement. The Factory pattern would be a great benefit to implement, as we could move the switch statement to a class of its own, and whenever we have a new UI component, we just have to add a single new case in the factory class. This works because of the fact that all the UI components extend the same parent class. Another minor comment that I had is to replace all the hardcoded strings with Enums if they are being used across multiple classes. This reduces the chances for typos and silly mistakes, and it brings a sense of consistency to all the developers. All in all, the code review process is beneficial to everyone in the team as we learn new shortcuts and ways to improve our code.

Matthew

Overall the quality of our code has been increasing, as we implemented our own code review process from the very beginning. Something I did notice from this sprint however was that we have two very similar methods in our db handler class which can easily be combined to a single method, reducing code duplication. It should also be renamed, as currently both names are quite vague and misleading. Currently they are named selectRows and selectCols, but in reality they are both selecting specified columns from every record, so a potentially better name might be: selectColsOfAllRecords. Another thing I noticed was the inconsistency between strings and Enums. We should be using Enums wherever possible, and not hard-coding strings if we don't have to.

Maggie

According to our team agreement on code review policy, each of the team members need to review the user story PR. After reviewing the PR for user stories 3 and 4, I noticed that we are still missing unit tests for the file parser and uploader. Since they are responsible for two main functionalities of the application, it is necessary to write test cases for those classes to improve the test coverage. Also, while reviewing the unit tests for DBHandler, I realized that multiple test cases, for example tests for selectCols and selectRows, share a Test table in the database, but some latter test cases add in more data into the table that may potentially affect the assertions of previous one. As a better practice, we should make each test case independent from test table changes, i.e. having before and after methods to add in rows required for specific test case and remove them after execution. By this approach, we should be able to keep the test table empty and always available instead of maintaining unknown data added for different tests. Generally speaking, we try to provide clean code with sufficient Javadoc and separate different operations into suitable structures so that it is easy to follow even though we work on different tasks.

User Stories (Version 2)

1. As Charlie (an Admin), I would like to create an account, so various users can use the application.

2. As Alice (an Agency Representative), I would like to upload an iCare excel file, so that the user's information is stored securely and I do not have to worry about losing my local copy.

3. As Ellie (an Agency Analyst), I would like for the application to let me know when it was unable to merge a field, so that I can fix the issue and reupload.

4. As David (CCS organization manager), I would like to generate reports (summary) so that I can use the report to improve future newcomer service.

5. As Charlie (an Admin), I want to be able to remove a certain user from the application or add a new agency to the application, as the users of the applications may have changed.

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.

7. As David (CCS organization manager), I would like to generate reports (trend) so that I can use the report to improve future newcomer service.

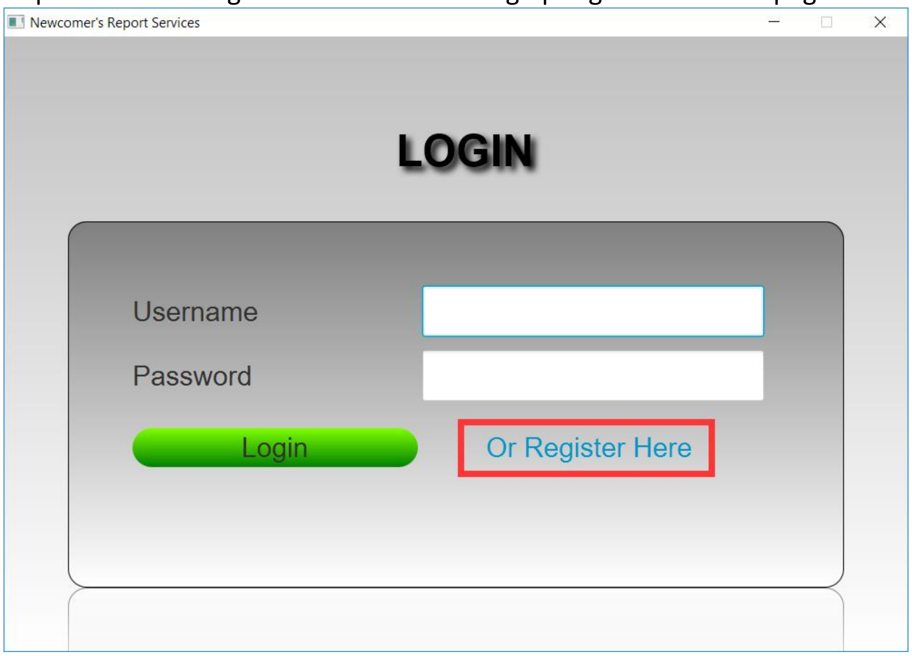
8. As Bob (an Agency Supervisor), I would like to generate my employees history, so I can monitor what changes were made and which employee made them in order to identify issues quickly if they arise.

Acceptance Test Appendix

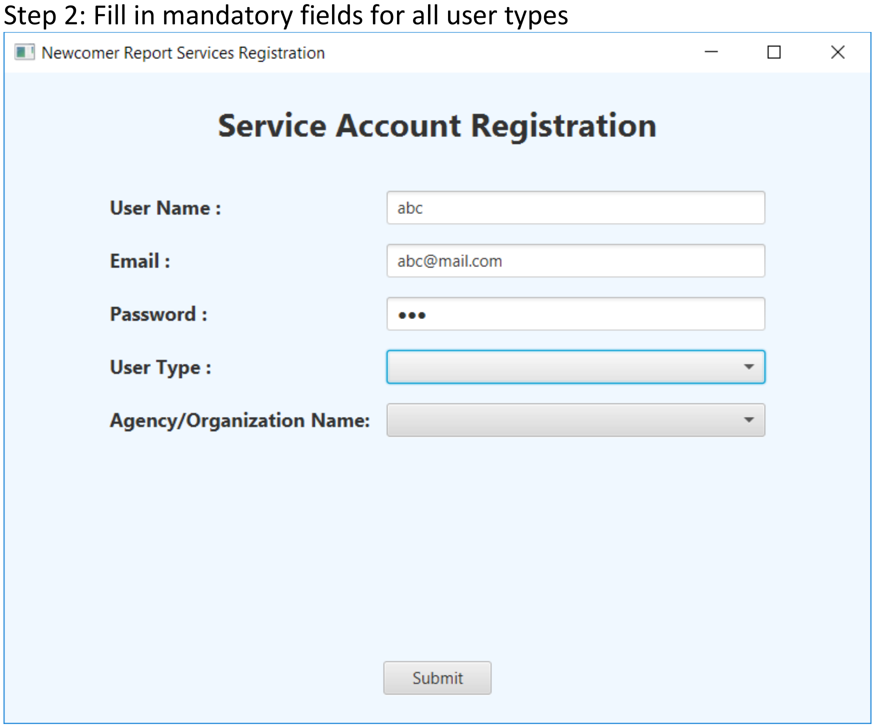
Validation of User Registration

1. Register as an AGENCY user

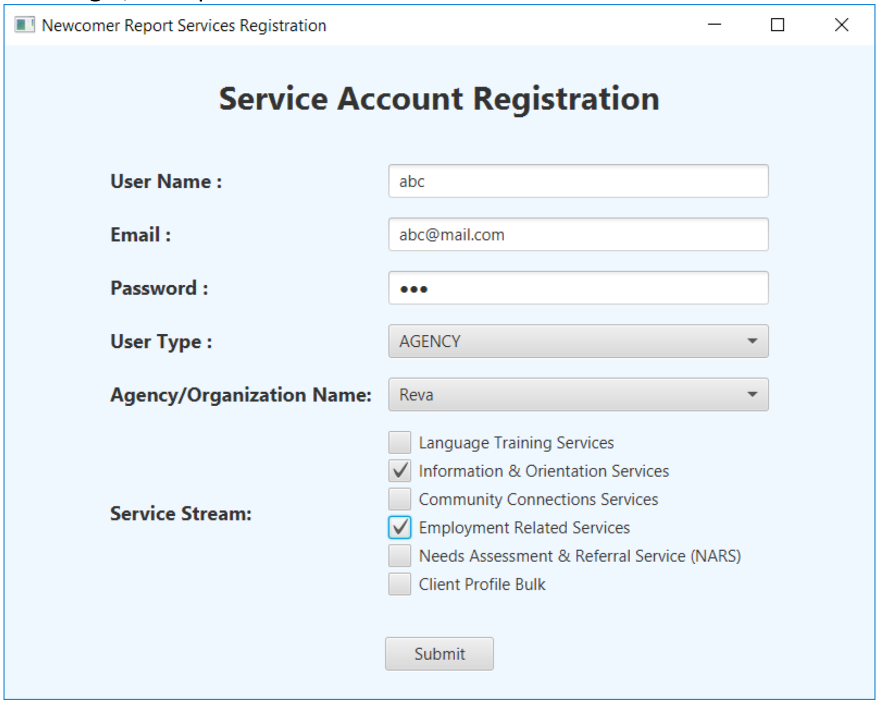
Step 1: Click “Or Register Here” link to bring up registration form page



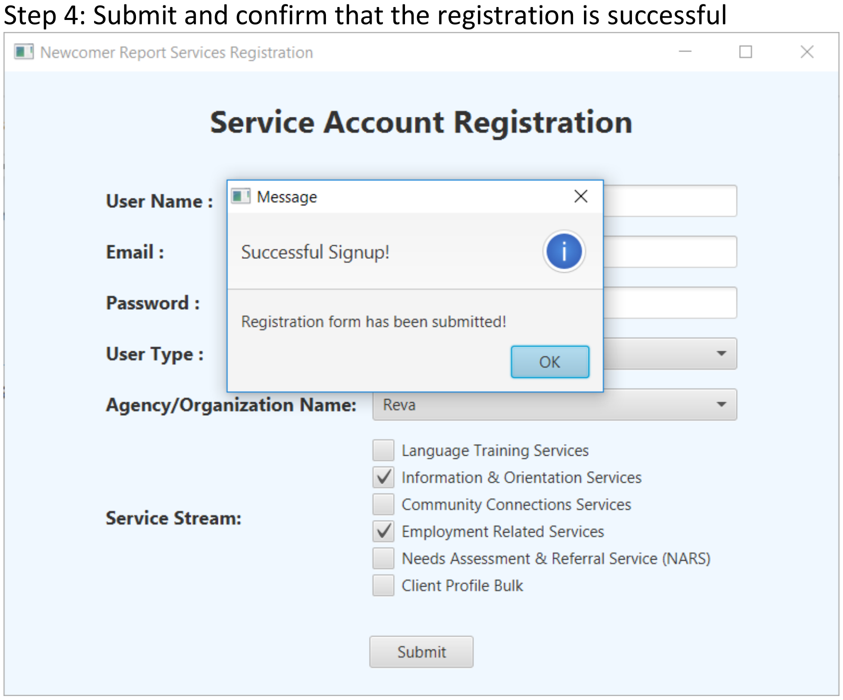
Step 2: Fill in mandatory fields for all user types



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

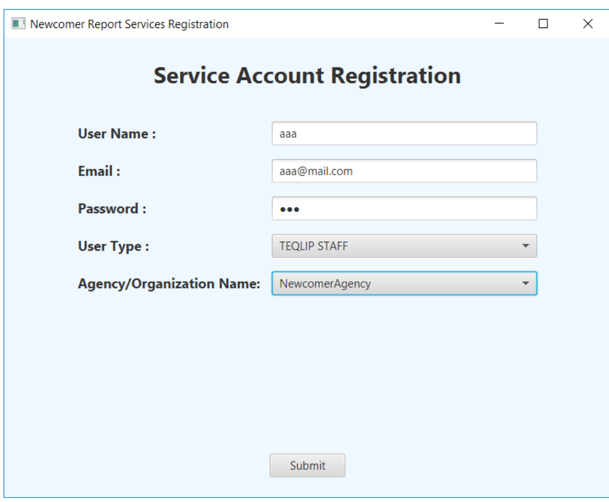
ID	UserType	Username	Password	OrganizationID	Email	EmploymentServ...
13	AGENCY	abc	abc	0	abc@mail.com	TRUE

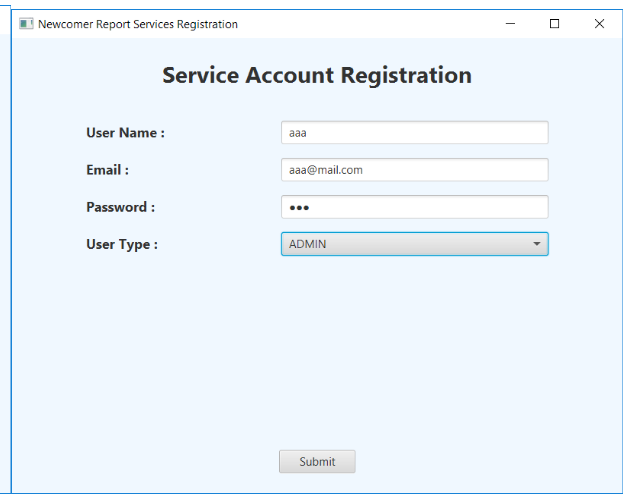
2. Register as an ADMIN/TEQLIP STAFF

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





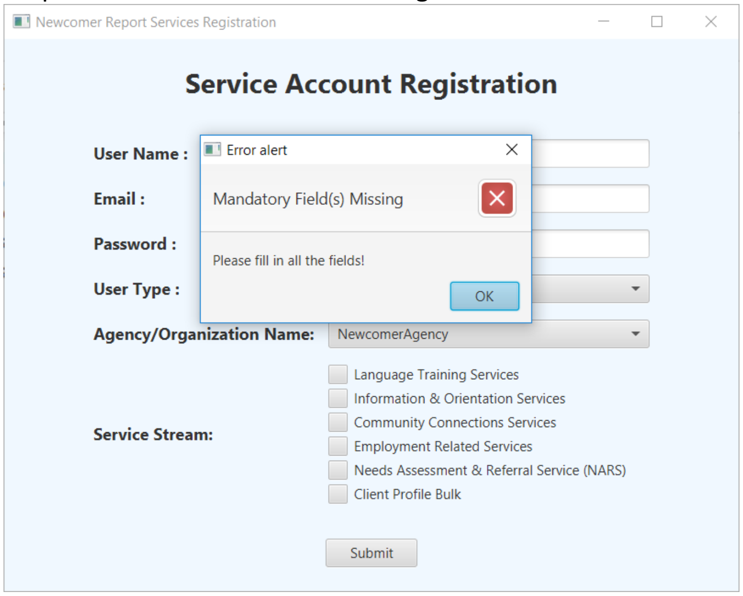
Step 4&5: Submit and confirm successful registration (same as above)

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



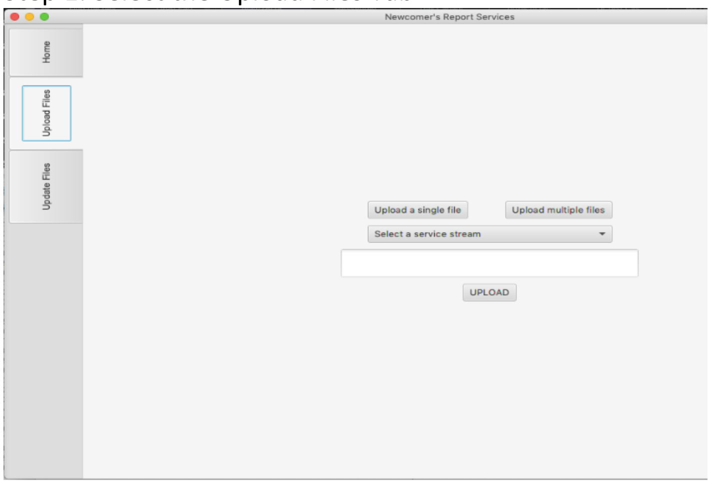
Step 4: Go into database to confirm that there is no user inserted with incomplete information (no user aaa)

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

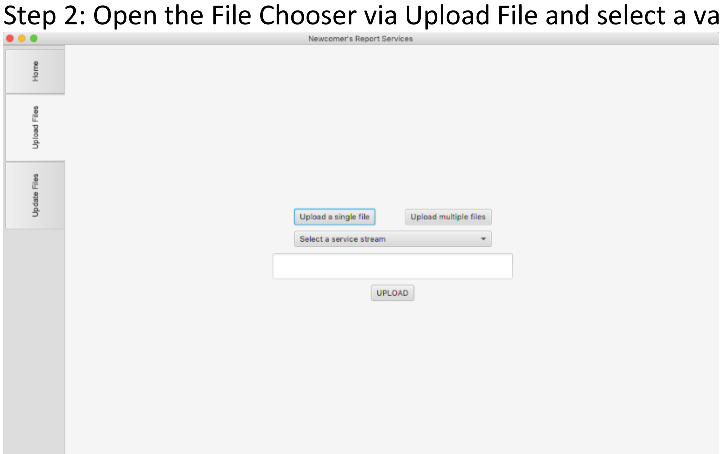
Validation of Uploading Content

1. Uploading a Valid Single File

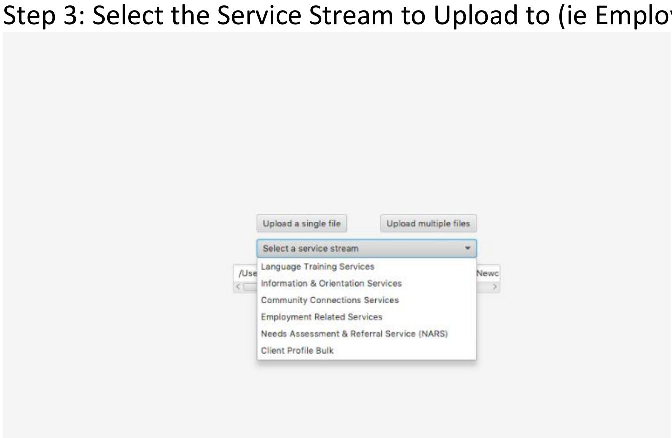
Step 1: Select the Upload Files Tab



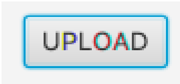
Step 2: Open the File Chooser via Upload File and select a valid .xlsx file



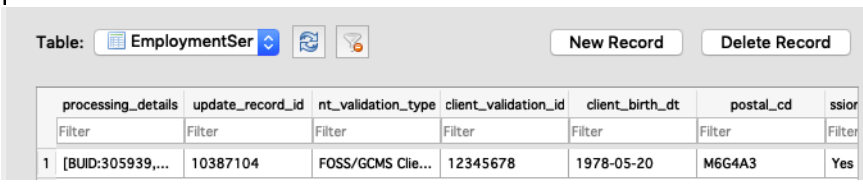
Step 3: Select the Service Stream to Upload to (ie Employment Service Stream)



Step 4: Click Upload

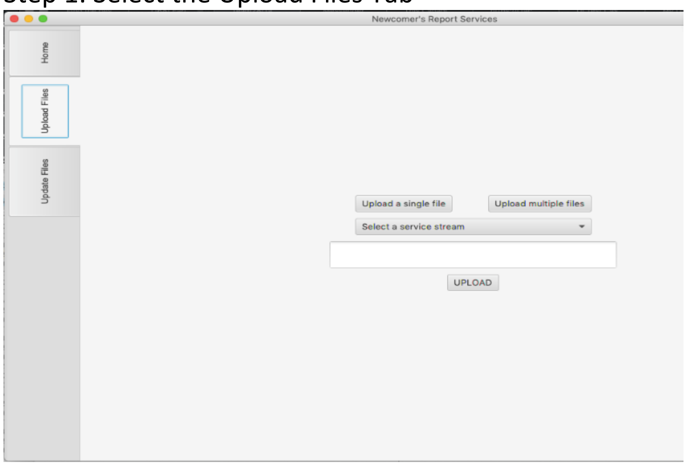


Step 5: Check the EmploymentServiceStream table in the db to see if your file has been parsed and pushed

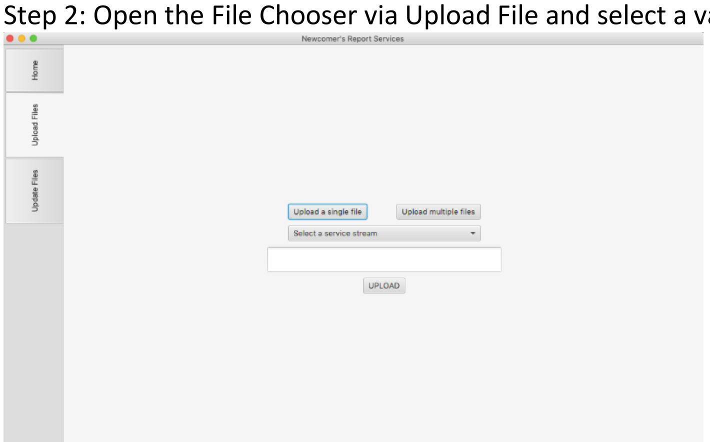


2. Uploading an invalid Single File

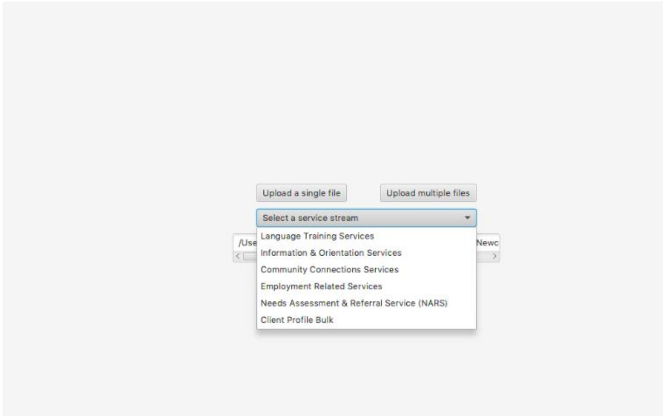
Step 1: Select the Upload Files Tab



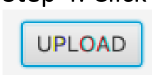
Step 2: Open the File Chooser via Upload File and select a valid .xlsx file



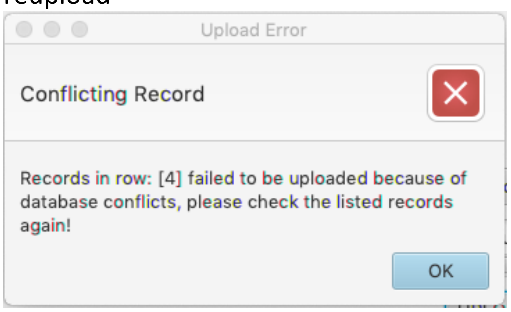
Step 3: Select the Service Stream to Upload to (ie Employment Service Stream)



Step 4: Click Upload

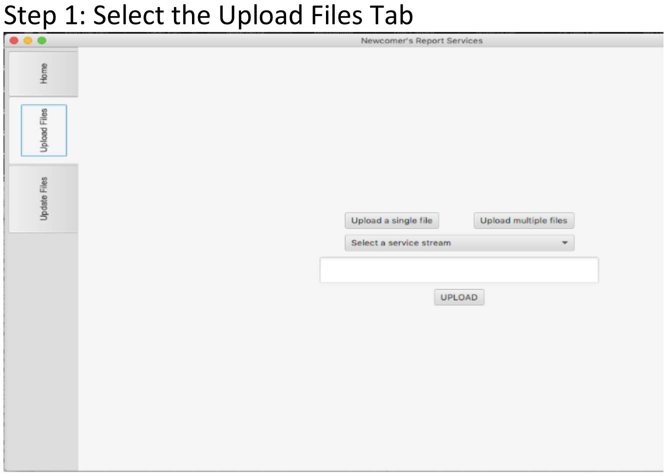


Step 5: You will be prompted with a dialog box informing you of your error, and you will have to reupload

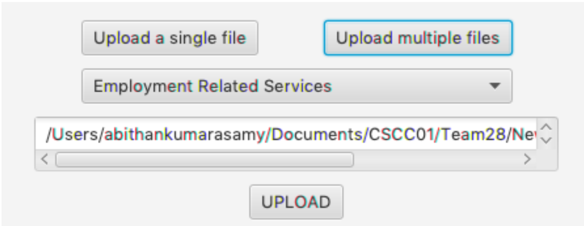


3. Uploading multiple files

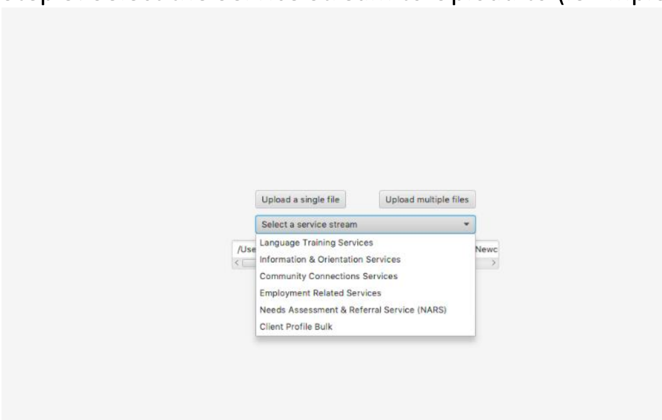
Step 1: Select the Upload Files Tab



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

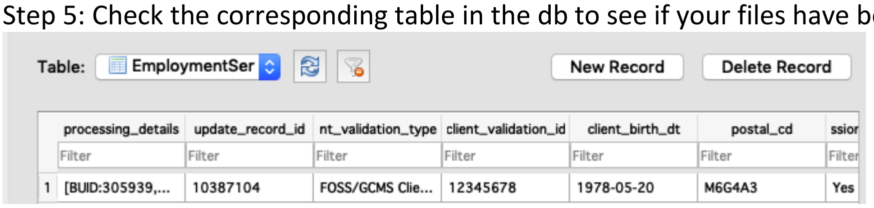


Step 3: Select the Service Stream to Upload to (ie Employment Service Stream)



Step 4: Click Upload

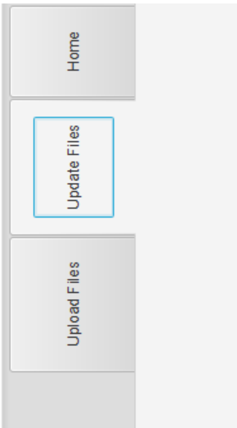
Step 5: Check the corresponding table in the db to see if your files have been parsed and pushed



Validation of Updating Content

Step 1: Login as an agency to the system

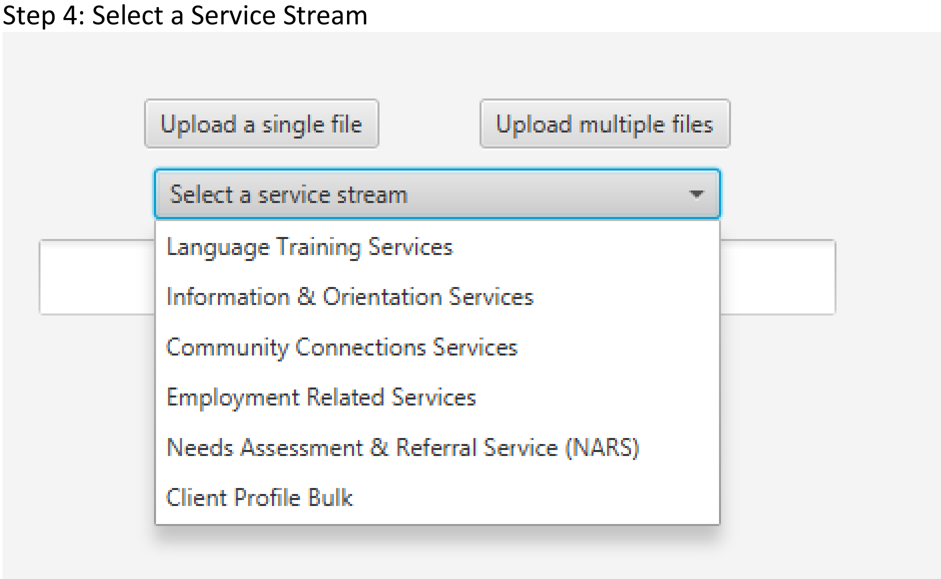
Step 2: Click on the Update Files tab



Step 3: Click on the Update Files tab

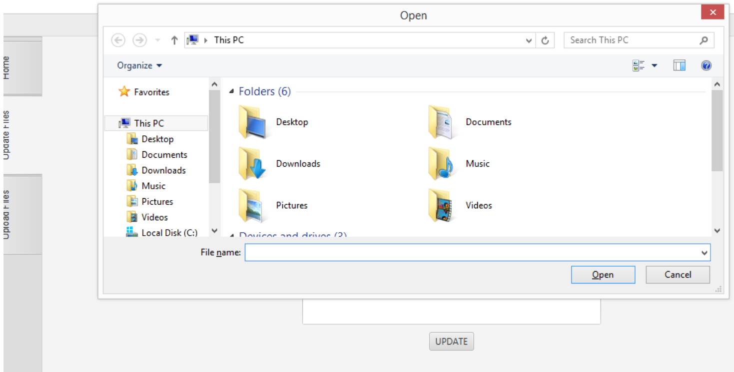
1. Updating a single file

Step 4: Select a Service Stream



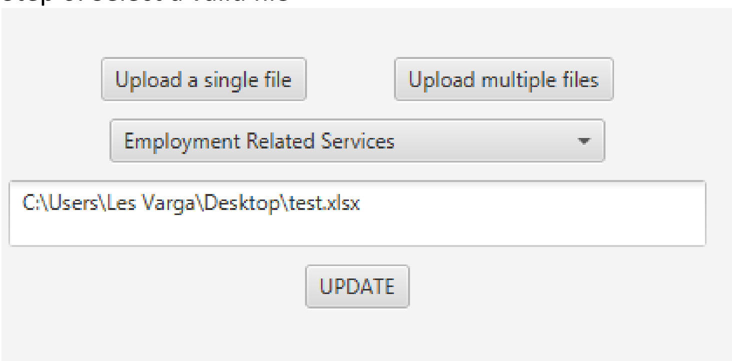
Step 4: Select Upload a single file

Step 5: Verify file selector pops-up



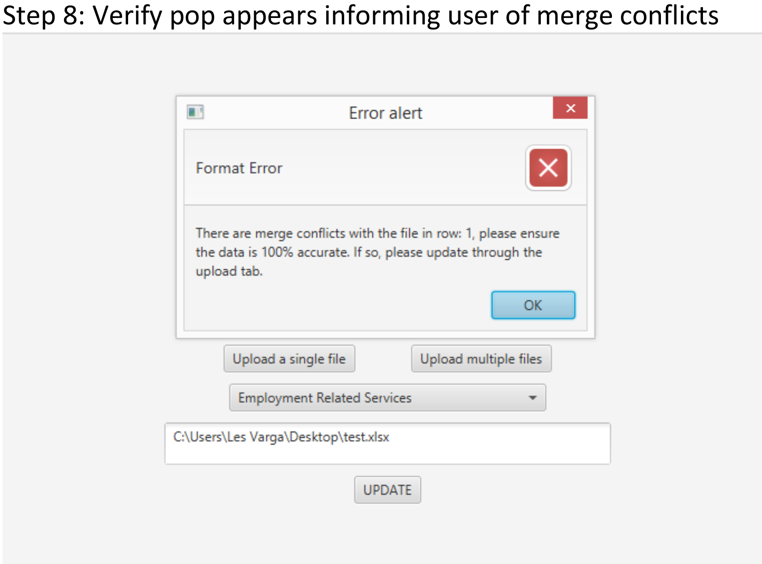
2. Selecting a valid file with merge conflicts

Step 6: Select a valid file



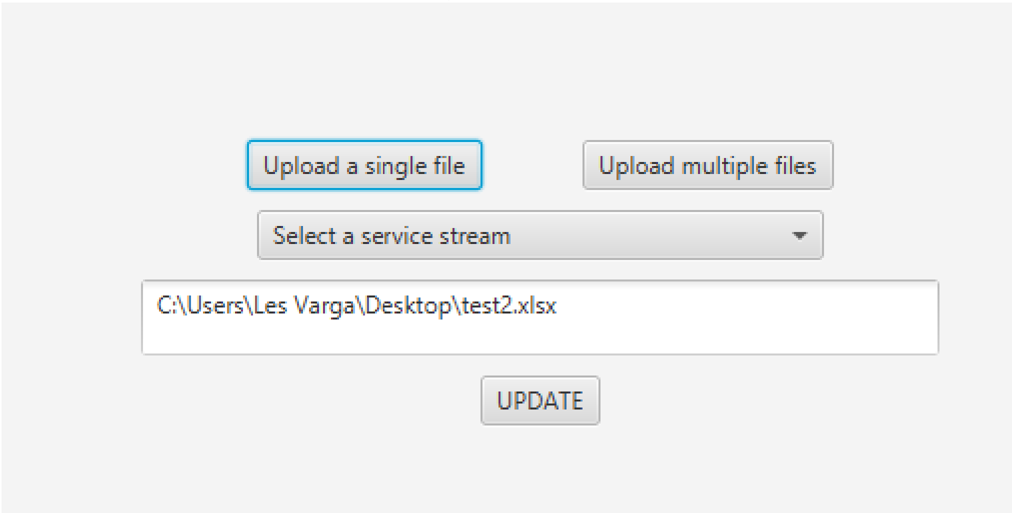
Step 7: Select Update

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

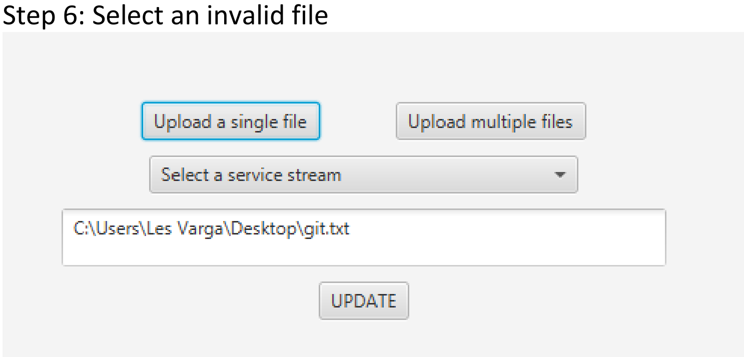


Step 7: Select Update

Step 8: Verify in DB that the file contents were properly uploaded

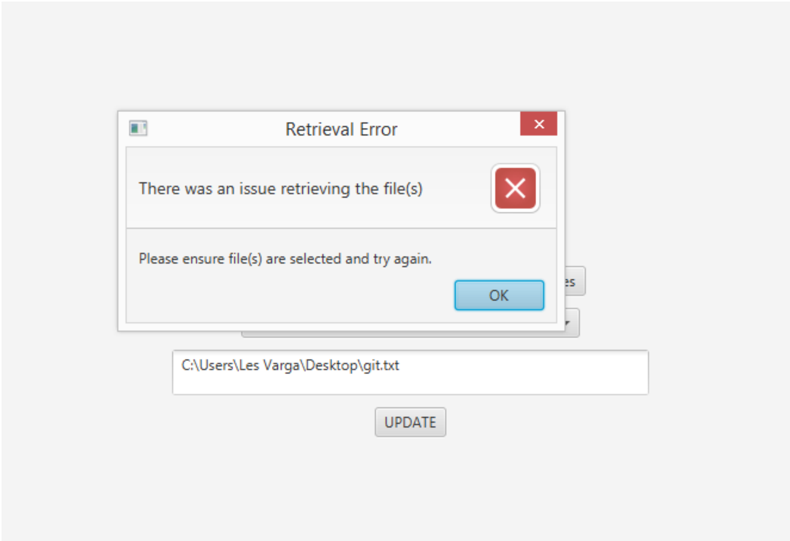
4. Selecting an invalid file

Step 6: Select an invalid file



Step 7: Select Update

Step 8: Verify error pop-up appears



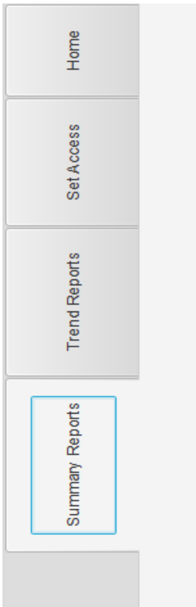
5. Updating multiple files

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

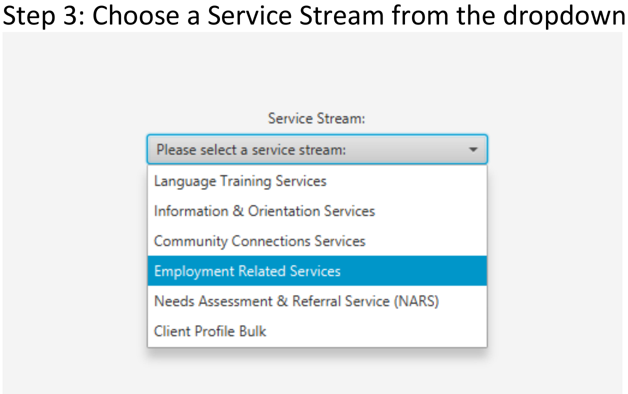
Validation of Summary Report Generation

Step 1: Login as an admin to the system

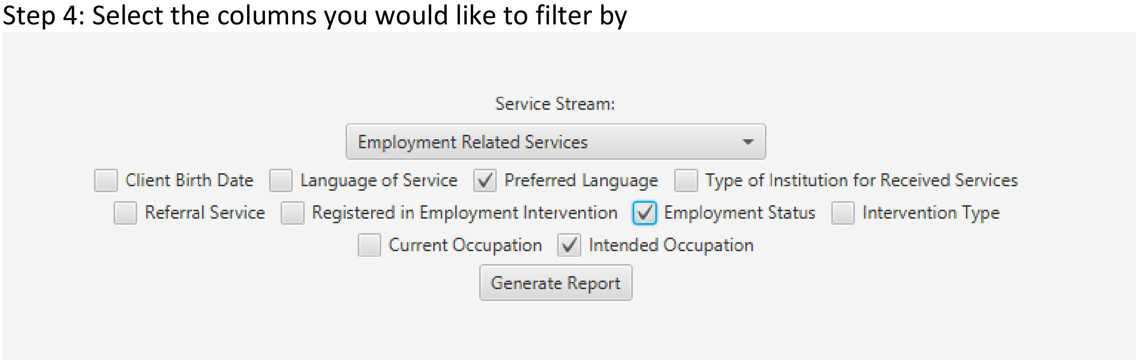
Step 2: Click on the Summary Reports tab



Step 3: Choose a Service Stream from the dropdown

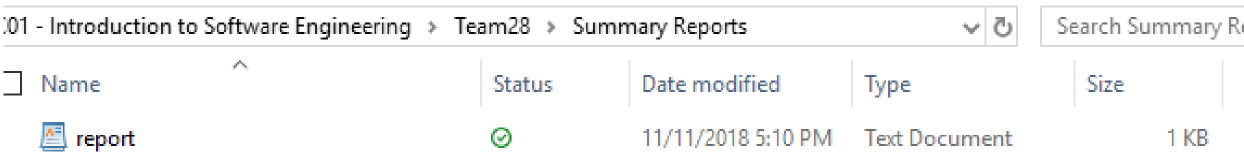


Step 4: Select the columns you would like to filter by

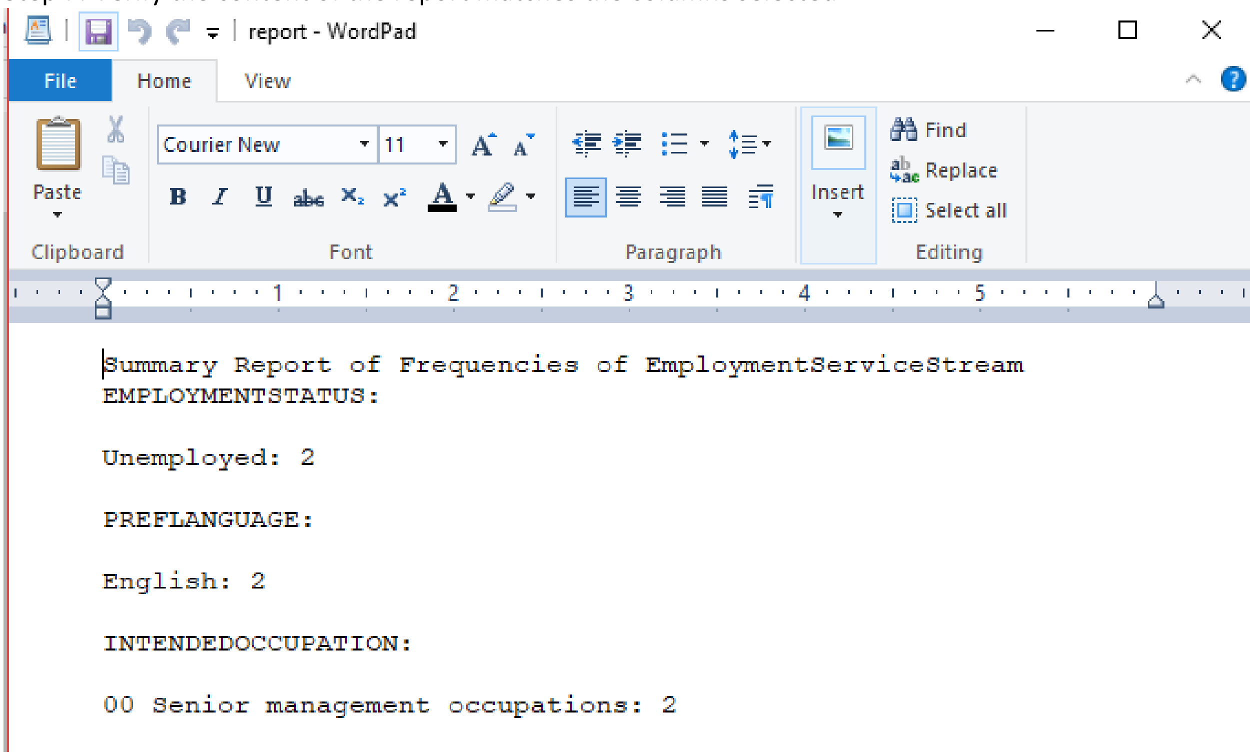


Step 5: Click the generate Report button

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



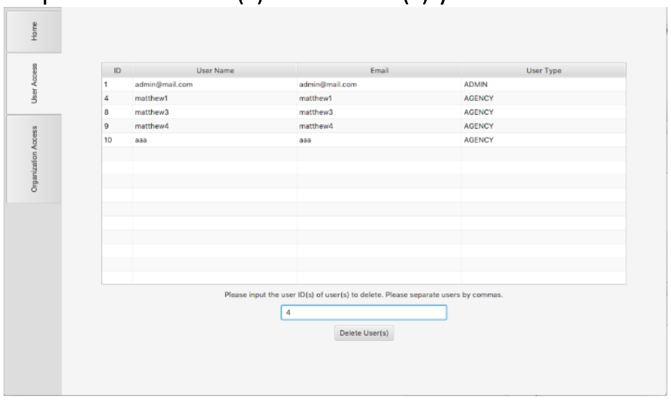
Validation of Admin Functions

1. Deleting a User

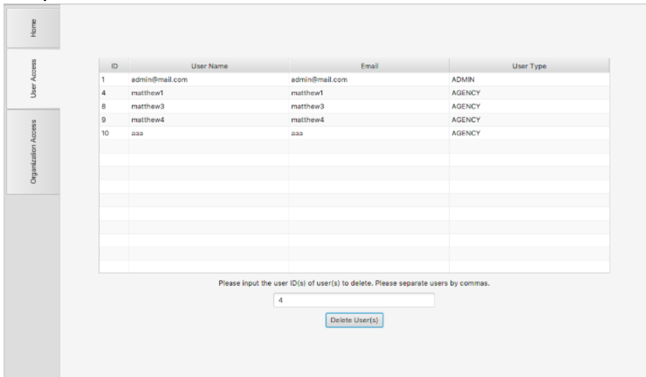
Step 1: Login as an admin to the system

Step 2: Click on the User access tab

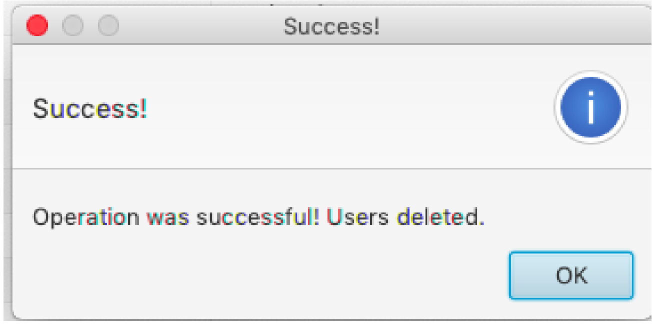
Step 3: Enter the ID(s) of the user(s) you wish to delete



Step 4: Click Delete

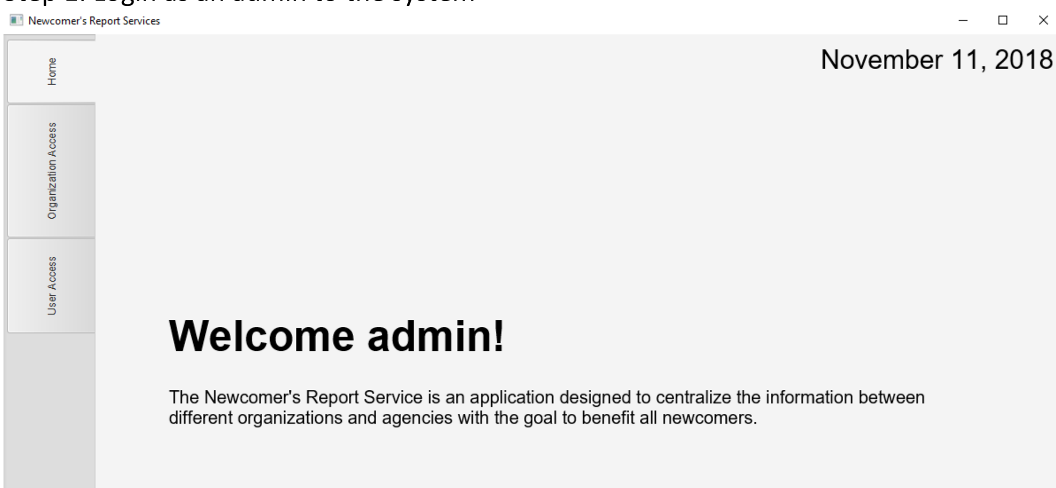


Step 5: Verify in the table above that the user has been deleted

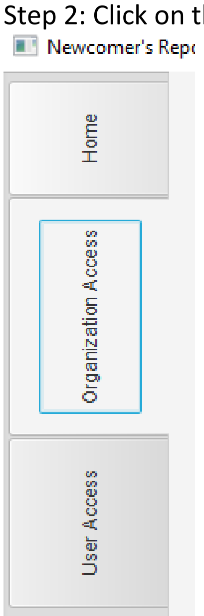


2. Inserting Organization

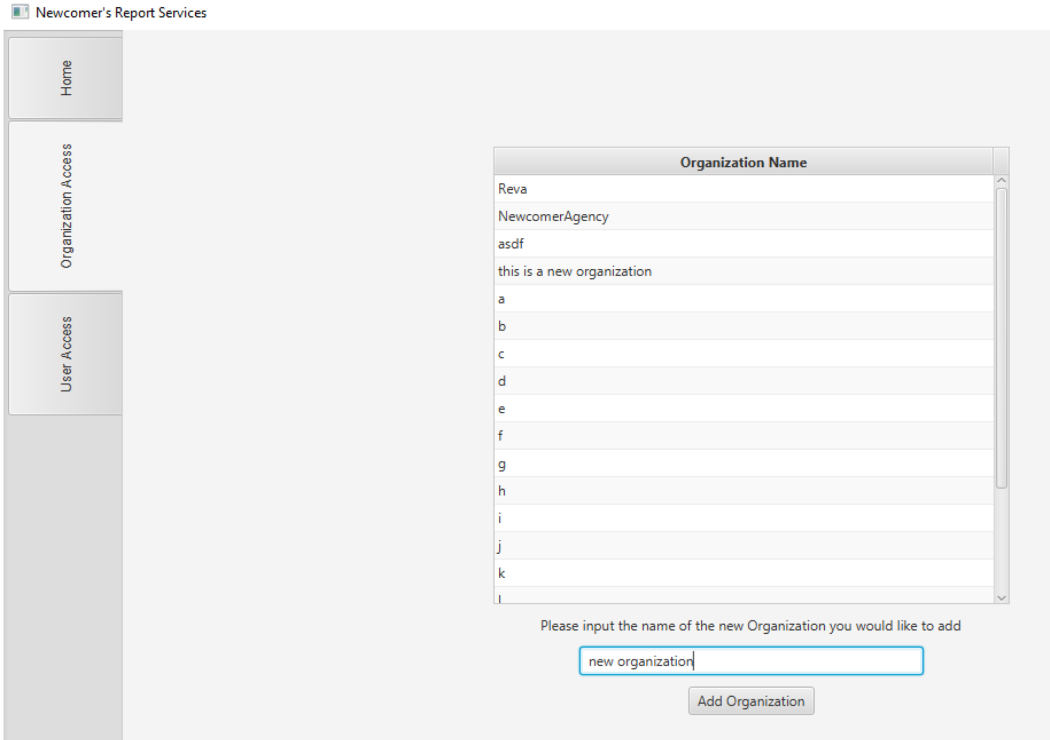
Step 1: Login as an admin to the system



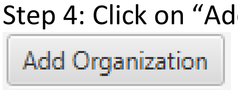
Step 2: Click on the Organization Access tab



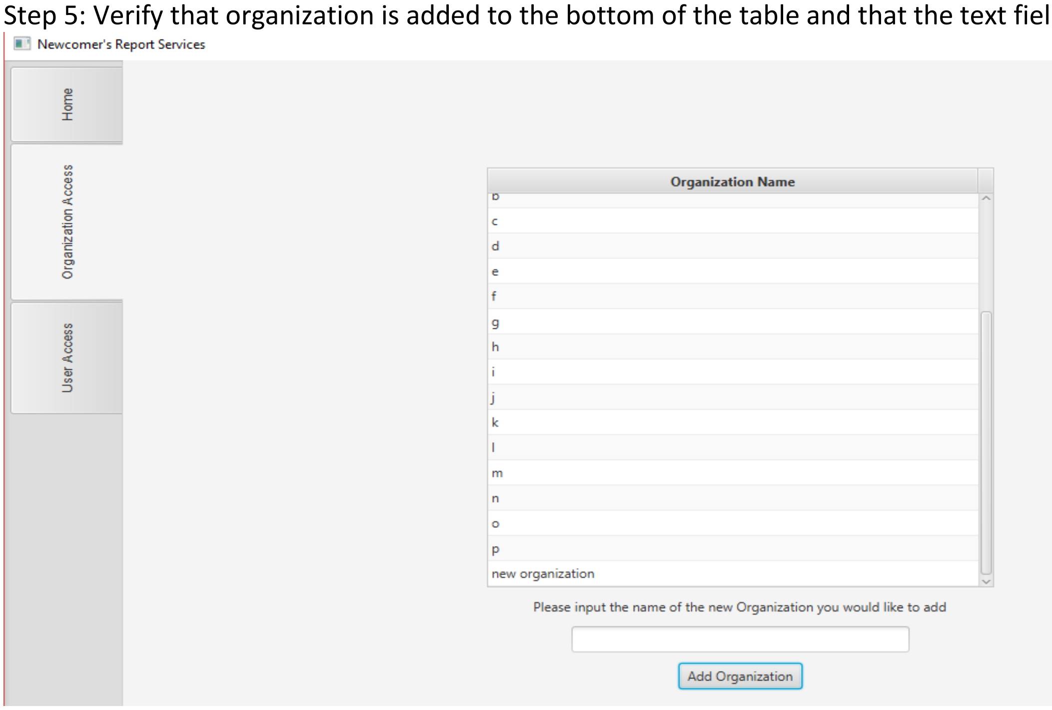
Step 3: Enter the name of an organization you would like to add in the text field



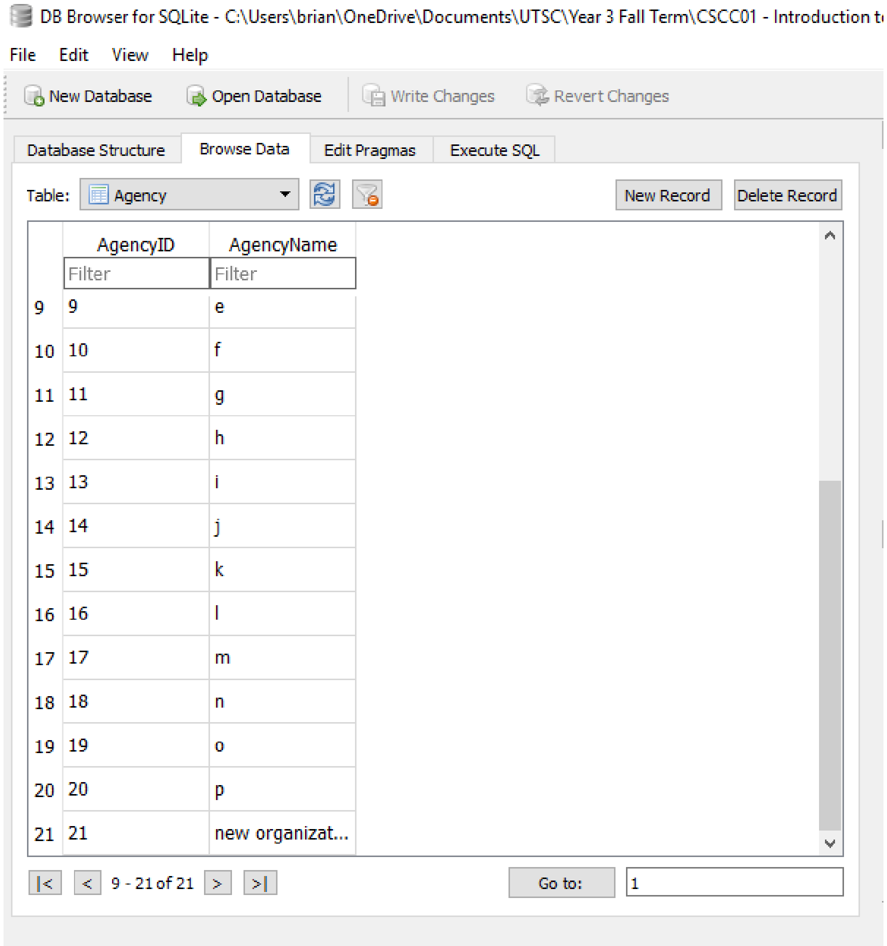
Step 4: Click on "Add Organization button"



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

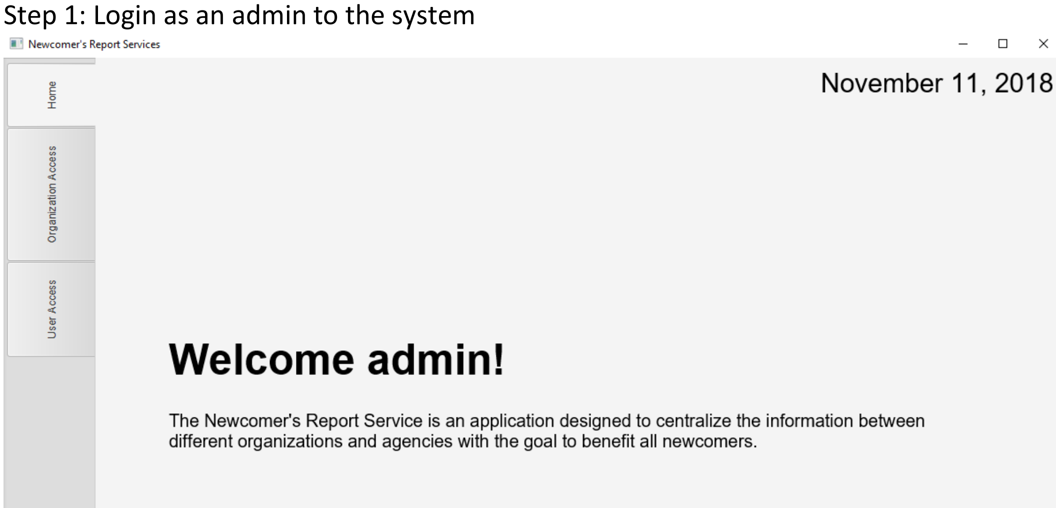


Step 6: Verify that the organization is added in the database

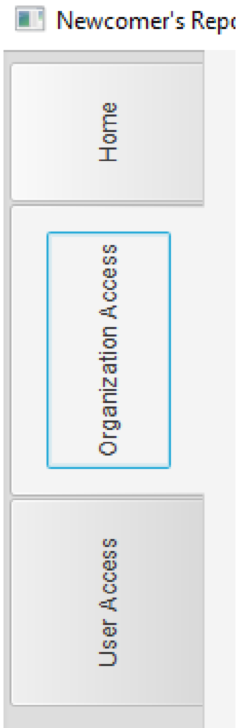


3. Inserting Organization that already exists

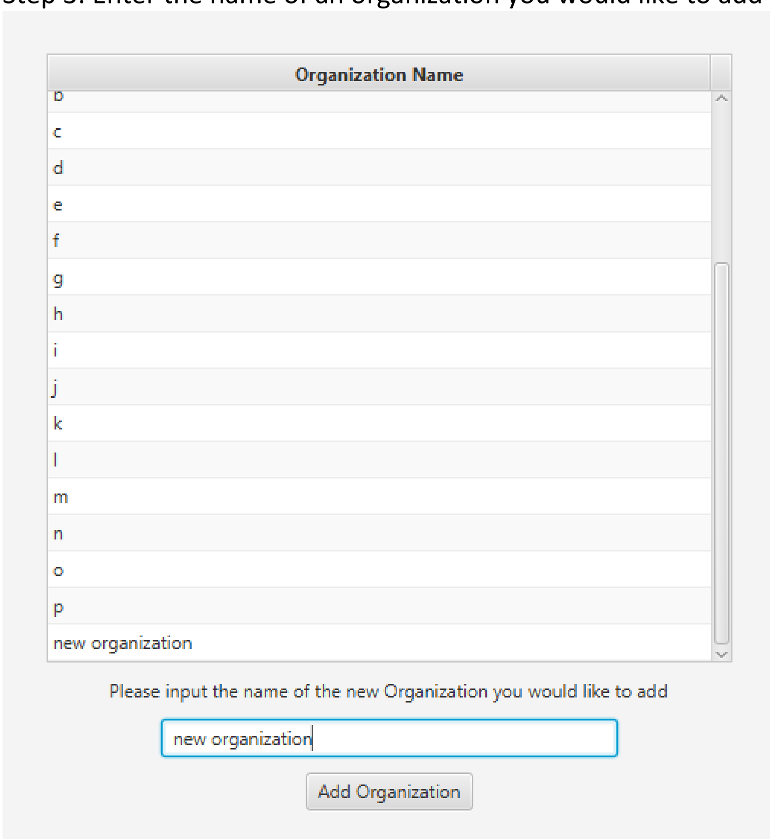
Step 1: Login as an admin to the system



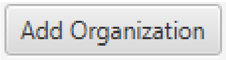
Step 2: Click on the Organization Access tab



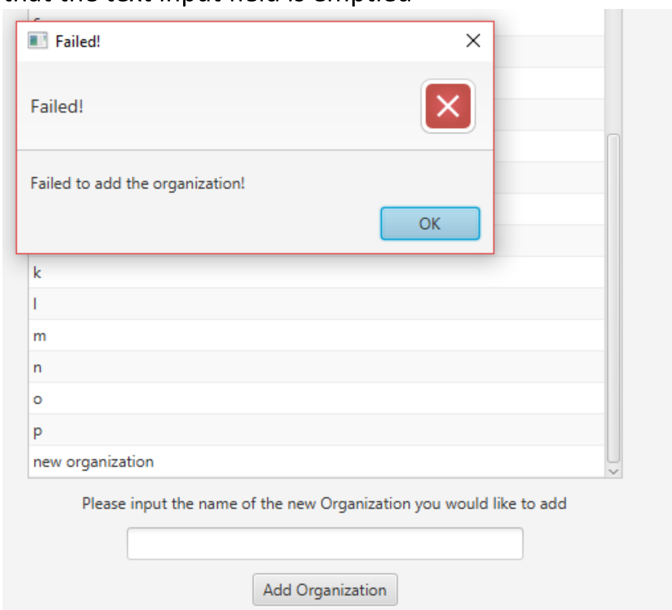
Step 3: Enter the name of an organization you would like to add in the text field (that already exists)



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