Things to work on from March 2nd to March 17th

Install updated project

You need the latest version of the project. Delete completely the old version of your ng-star project that you have installed (this includes all directories and all files). You might want to create a tarball of the ng-star project that you currently have as a backup.

Change directory into your home folder:

cd ~

On GitLab, go to the NG-STAR project and click on Project. Copy the SSH link which should look like this:

```
git@gitlab.corefacility.ca:irish.medina/ng-star.git
```

In your terminal, run the following command:

```
git clone git@gitlab.corefacility.ca:irish.medina/ng-star.git
```

Change directory to ng-star/MySQLWorkbench/PopulateDatabaseStatements

Log into mysql:

```
mysql -u root -p
use NGSTAR;
```

Clear the contents of the NGSTAR database by running:

```
source clear_db.sql
```

Next, run the following commands to ensure that the tables are populated:

```
select * from tbl_IsolateClassification;
```

If this table does not contain the values specified in insert_classification_codes.sql then run the following command:

```
source insert_classification_codes.sql
```

Next, run the command:

```
select * from tbl_Loci;
```

If this table does not contain the values specified in insert_loci.sql then run the following command:

```
source insert_loci.sql
```

Run the command:

```
select * from tbl_MIC;
```

If this table does not contain the values specified in insert_mic.sql then run the following command:

```
source insert_mic.sql
```

You will have to revisit the installation documentation in Ubuntu_Desktop_Installation_Guide.md in the Documentation folder and complete the appropriate steps to finish the installation (there are some steps that you might not need to do, such as installing software which you already have). This will probably include running generate_db_objects to generate database table classes, creating a directory called 'data' in ng-star/BusinessLogic if it doesn't already exist, changing any hardcoded path lines as specified in the installation documentation, running `perl Makefile.PL`, `make` and `sudo make install` for both BusinessLogic and DAL modules, starting up the development server to see if everything works and being able to login with the username 'test01' and password 'mypass'.

After you have installed the updated version of the project you should try running the Selenium IDE tests for our project.

The Selenium IDE tests (located in ng-star/SeleniumIDETests) are the only tests that we have right now for the project. Selenium IDE tests are not “real” tests and they were only written so I could get familiar with Selenium. I am planning on re-writing all tests for Selenium WebDriver in the future, but for now we will be running the TestSuite.html (located in ng-star/SeleniumIDETests) using the Selenium IDE plugin in Firefox to run our integration tests.

If you don't have the Selenium IDE plugin in Firefox then you need to install that.

On Selenium IDE, go to **File**, **Open** and then select TestSuite.html. Run the entire test suite by double clicking on the first test case and clicking on **Play Entire Test Suite**.

If the installation is successful then all tests should have passed.

Setup git development model

You need to setup your ng-star project (that you git cloned from GitLab) to follow our git development model.

a) Make sure that you are on the development branch:

```
git checkout development
```

b) Once you are in the development branch, create a branch called 'experimental' that branches off of development:

```
git branch experimental
```

c) Ensure that you have created a branch called experimental when viewing branches:

```
git branch
```

d) Switch to the experimental branch if you are not on the experimental branch already:

```
git checkout experimental
```

e) All the functionality that you implement for now will be pushed into the experimental branch (and not master or development). When pushing your code, always make sure that you are pushing only to the experimental branch. To push code to experimental, you would run the following command after at git add and commit (and after all tests have passed):

```
git push origin experimental
```

f) Push your experimental branch remotely (this branch will now show up on GitLab):

```
git push origin experimental
```

Your code will eventually be merged by your team member after they have reviewed your code from a merge request that you will submit much later on.

Procedure

You will be adding features locally on the ng-star project that you git cloned from GitLab. During development you run the development server on your local machine by running the command `script/ngstar_server.pl -r`

You will be able to see the progress of your development by starting off at: http://localhost:3000/allele/form

If you make any changes to the BusinessLogic or DAL code, you have to `make` and `sudo make install` these and then restart the development server on your local machine.

If you are adding a new file in the ng-star/BusinessLogic/lib/BusinessLogic folder, then you need to remember to include it in the MANIFEST in the BusinessLogic top directory and run `perl Makefile.PL`, `make` and `sudo make install`. You also need to remember to create an Adaptor in Catalyst MVC to connect the Catalyst Model layer to the BusinessLogic layer code. If you are unsure, there is an example in the AlleleDatabase tutorial.

You should try committing early and often to the experimental branch. Once you have completed a single feature that is not too large, you can try running all Selenium IDE tests and making sure that all tests pass. If a test fails because you changed the behavior of the application somehow, then you need to update the tests. Once all tests pass you can add and save to the staging area and push all code to experimental:

```
git add -A
git commit -m 'My message'
git push origin experimental
```

If you stop running the Selenium IDE tests in the middle of the test suite or if some tests fail then you may still have values in your database that have not been cleaned up. When you try to run the test suite again from the beginning this may fail since the tests assume that you are starting with an empty database (of alleles and profiles). To clear the database, change directory to ng-star/MySQLWorkbench/PopulateDatabaseStatements and then log into mysql:

```
mysql -u root -p
use NGSTAR;
```

To clear the database, run the following command:

```
source clear_db.sql
```

When more than one person is pushing code to the project, you need to make sure that you `git pull` periodically (at the start of the day to the end of the day) and that you `git pull` after a merge request has been accepted. This is very important.

Functionality to Implement

(you can select which task you want to work on in whatever order you want)

You can use the NGSTAR administrator account with the following credentials to log in:
Username: test01
Password: mypass

1. Add FormHandler form for batch adding alleles

Create a FormHandler form for batch adding alleles so that it checks in the client side if there are any invalid characters in the textbox. Right now we are only validating characters on the server side. Invalid characters would be considered special characters such as '$', '%', '@', '^', '(', ')', etc (but not '>' and some commonly used special characters such as '_' and '-'). All the letters of the alphabet and numbers should be considered valid characters. There are multiple FormHandler forms that you can take a look at in ng-star/NGSTAR/lib/NGSTAR/Form for your reference.

For this task you only need a partial FormHandler form (since we only need to client side validate the single textarea). BatchAlleleForm.tt2 should mostly stay the same except for the lines:

```
<div class="form-group">
    <label class="col-sm-2 control-label">Enter sequences in FASTA format:</label>
    <textarea name="fasta_sequences" class="form-control" rows="30"
cols="95"></textarea>
</div>
```

Here, you would just render the textarea you specified in your FormHandler form similar to this:

```
<div class="form-group">
    <label class="col-sm-2 control-label">Enter sequences in FASTA format:</label>
    [% form.field('fasta_sequences').render %]
</div>
```

There may be a few other things you need to change in BatchAlleleForm.tt2 (but most of BatchAlleleForm.tt2 should not have to change).

Once you are done, run all tests and make sure that they pass. You may have to update some of the tests (due to a change in the functionality and not due to the application not behaving as we expect it to) in order to make the tests pass. If you enter multiple sequences in FASTA format that contains invalid special characters (except for '>' and some commonly used special characters such as '-' and '_') then you should get an error message immediately without calling the server side when submitting the form. If you have included duplicate allele types or duplicate sequences (and all client side validation passes), then these can only be checked in the server side only and you should get an error message after validating in the server side.

Once all tests pass, make sure all your code is committed to your feature branch called 'experimental'. Typically you would submit a merge request after the functionality in your feature branch is complete (to be merged into the development branch) but since your team member is not here then you do not submit a merge request right now. You will submit a merge request (to merge experimental into development) after your team member returns (in which they will review all the code you have checked into experimental).

2. Add FormHandler form for batch adding profiles

This is similar to the previous task. You need to create a FormHandler form for batch adding profiles so that it checks in the client side if there are any invalid characters in the textbox (right now it only does server side validation). In this case, the only valid characters are numbers, commas and newlines (everything else is invalid).

Similar to the previous task, most of BatchAddProfileForm.tt2 will stay the same except the textarea that you are rendering from the FormHandler form that you created.

Make sure that all tests pass before pushing your code to experimental. If tests fail because behaviour changes as we expect, then you will need to update the tests until they pass and then push your code when all tests pass.

3. Fix the “Additional Epidemiological Data” field in the <u>Add Allele</u> functionality to consider commonly used special characters such as a comma, as a valid character. In order to mitigate SQL Injection or Cross Site Scripting, you should still consider characters such as '{', '}', '@', '$', '=', '<', '>', '/', '\', ''', '"', '(', ')' and ';' as invalid characters (and any other characters used in code).

To fix this field you will have to update client side validation in the FormHandler and potentially server side validation located in the Controller and in the Business Logic Layer.

When a user enters invalid characters, a client side message that says "Please enter valid epidemiological data" as specified in the FormHandler form is displayed. Change this message to something more descriptive such as "Please enter valid epidemiological data that does not include any of these characters: { } @ $ = < > / \ ' " ( ) ;".

Push code early and often after all tests have passed.

4. Fix the “Additional Epidemiological Data” field in the Edit Allele functionality to consider commonly used special characters such as a comma, as a valid character. In order to mitigate SQL Injection or Cross Site Scripting, you should still consider characters such as '{', '}', '@', '$', '=', '<', '>', '/', '\', ''', '"', '(', ')' and ';' as invalid characters (and any other characters used in code).

To fix this field you will have to update client side validation in the FormHandler and potentially server side validation located in the Controller and in the Business Logic Layer.

When a user enters invalid characters, a client side message that says "Please enter valid epidemiological data" as specified in the FormHandler form is displayed. Change this message to something more descriptive such as "Please enter valid epidemiological data that does not include any of these characters: { } @ $ = < > / \ ' " ( ) ;".

Push code early and often after all tests have passed.

5. Fix the password field (on user account creation) so that it accepts commonly used special characters. To mitigate SQL Injection or Cross Site Scripting we will still consider characters such as '{', '}', '@', '$', '=', '<', '>', '/', '\', ''', '"', '(', ')' and ';' as invalid characters.

To fix this field you will have to update client side validation in the FormHandler and potentially server side validation located in the Controller.

When a user enters invalid characters, a client side message that says "Please enter a valid password" as specified in the FormHandler form is displayed. Change this message to something more descriptive such as "Please enter a valid password that does not include any of these characters: { } @ $ = < > / \ ' " ( ) ;".

Push code early and often after all tests have passed.

6. This is an intermediate level difficulty task. You might want to only try out this task only after you have finished 2 to 6. Before starting this task you need to have a good understanding of the NGSTAR database schema. If you don't remember Relational Databases then you will have to review that. Add a metadata field called "Mutations". Walter wanted to add this feature previously as specified here:

https://drive.google.com/folderview?id=0B4Vx2hKNcWEaY3htWGVmQ2FvV1k&usp=sharing

First you need to contact Irene and Walter to ensure that they still want this feature and to gather any additional requirements.

Metadata is present in the Alleles (in add alleles, edit alleles and details) and in the ST profiles (in add profile, edit profile and details).
BUT, Walter specifically mentioned that he only wants the "Mutations" field to be in the allele metadata only (and not the NGSTAR profile metadata). You may want to double check with Walter again to make sure this is what he really wants.

If Irene and Walter still want this feature then you can try adding the field in the following order:

a) Add to add allele form (with proper client and server side validation). Run all tests to see if anything broke and then push to experimental branch (before starting step b).
b) Add to allele details. Run all tests to see if anything broke and then push to experimental branch (before starting step c).
c) Add to edit allele form (with proper client and server side validation). Run all tests to see if anything broke and then push to experimental branch.

7. This is an intermediate level difficulty task. You might want to only try out this task only after you have finished 2 to 6. Before starting this task you need to have a good understanding of the NGSTAR database schema. If you don't remember Relational Databases then you will have to review that. Add a metadata field called "Curator Comments". Walter wanted to add this feature previously as specified here:

https://drive.google.com/folderview?id=0B4Vx2hKNcWEaY3htWGVmQ2FvV1k&usp=sharing

First you need to contact Irene and Walter to ensure that they still want this feature and to gather any additional requirements.

Metadata is present in the Alleles (in add alleles, edit alleles and details) and in the ST profiles (in add profile, edit profile and details). Irene and Walter mentioned that they want this field to be part of both the metadata for alleles and NGSTAR profiles. You may want to contact them to see if any plans have changed.

If Irene and Walter still want this feature then you can try adding the field in the following order:

a) Add to add allele form (with proper client and server side validation). Run all tests to see if anything broke and then push to experimental branch.
b) Add to allele details. Run all tests to see if anything broke and then push to experimental branch.
c) Add to edit allele form (with proper client and server side validation). Run all tests to see if anything broke and then push to experimental branch.
d) Add to add NGSTAR profile form (with proper client and server side validation). Run

all tests to see if anything broke and then push to experimental branch.
e) Add to NGSTAR profile details. Run all tests to see if anything broke and then push to experimental branch.
f) Add to edit NGSTAR profile form (with proper client and server side validation). Run all tests to see if anything broke and then push to experimental branch.

I will be returning on March 18th. There is alot specified in this document so you are not expected (and you should not) complete everything on this list by March 18th. If you are going at a good and careful pace then you should not have completed all tasks on this list. You can submit a merge request (to merge experimental into development) when I return.

If Irene and Walter post issues or email about features that they want to add or change (from the test server), we can deal with those features after I get back. For now we will focus on dealing with higher priority issues such as the tasks in this list.