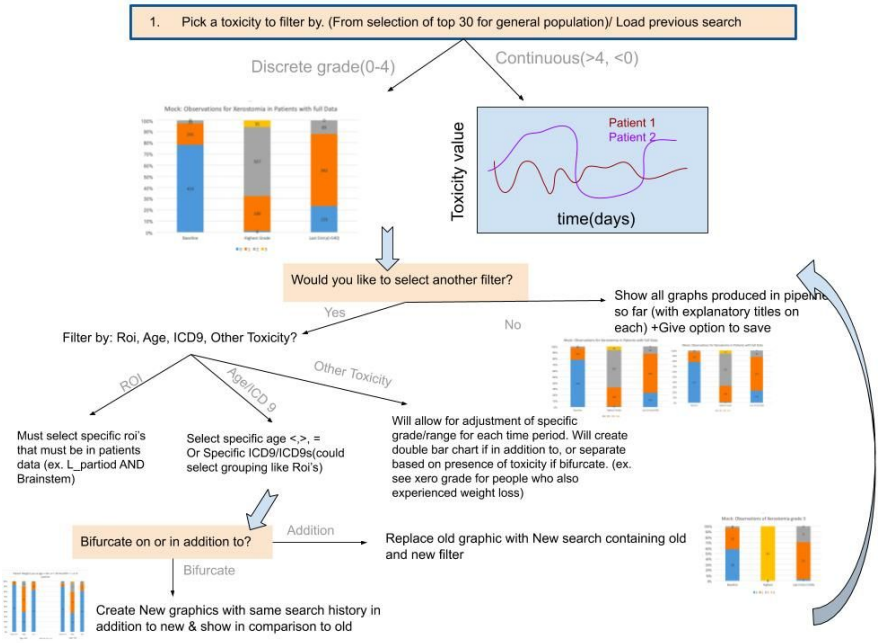
Application Documentation

User Experience/Pipeline Description:

The user will be prompted to select a toxicity to first filter by. This will produce the first visualization. The user will then be able to continue searching by another variable. They can choose between ROI, age, or ICD9. The patient ID's, obtained from the first toxicity filter, will be filtered again with the new selected variable. The resulting ID's from this new filter will then be displayed through visualization and can be further filtered. The user should be able to add on filters until satisfied with the results.



Major Filters:

Toxicity (Initial)
Age
IDC9
ROI

Toxicity:

User Option:

User is given a list of the 30 most common toxicities for the patient population in the pipeline at the moment. User can select one option.

Function Input:

If this filter is the start of the pipeline use Patient ID's of the entire database. If pipeline has already started, use Patient ID's in currently in the search pipeline. Toxicity selection from above.

Query details:

From Patient ID's input select the patients who have had an instance where toxicity is equal to toxicity selection input. In addition, dates of that documented toxicity must include a date <7 (a baseline), a date between 180-365, and a date >540. This ensures that the patient has completed full treatment and recovery status can be identified.

Function Output:

Output patient ID's with the given requirements above along with graphic described below

Graphics:

Dependent on whether toxicity has continuous grade or discrete grade data:

Discrete(grade <=4 And >=0 - a naive determination):

Type of Graph: Stacked Bar Chart

X axis: 3 distinct groupings of Baseline Score(earliest date), Highest Score Achieved, Last Score in time

Y axis: Can have two representations. One is as percentage of patients in current search pipeline. The second is the quantity of patients. (ability to toggle between percentage of patients and count)

Interactivity: When the toxicity filters have been added and bar graphs have been shown, there is a filtering tab labeled "Grade". The design of the of this filter should be similar to the age filter, and be divided into 3 parts. The first part will filter "First Entry" for a desired grade range. The second part will filter "Highest Entry" for a desired grade range. The third part will filter "Last Entry" for a desire grade range.

Continuous(any instance >4 or <0):

Type: Line Plot

X axis: Date

Y axis: Value of grade

Legend: each patient is represented by a line in the line graph

Interactivity: When the toxicity filters have been added and the line graphs have been shown, there is a further filtering option in the tab labeled "Grade/Date". The design of this filter should be similar to the age filter, but have two areas to input a range of values. The first area of input is for the range of the date. The left box is for inputting the lower boundary and the right box is for inputting the upper boundary. The second area of input is for the range of the grade. The left box is for inputting the lower boundary while the right box is for inputting the upper boundary. After clicking the button "Go", the

patient IDs will be filter to only contain patients that had a value in this range. “Restore” button will remove the application of the filters

ROI:

User Option:

User given a list of top 30 ROIs for population in current pipeline. Top 30 ROIs are chosen by frequency of occurrence in the data. The user should be able to select any number of the ROIs presented.

Function Input:

ROI Selection list and Patient ID’s in pipeline. If pipeline has not started, use all data in database.

Query details:

Given some input, the current patient IDs in the pipeline will be narrowed down by selecting the patient IDs that have an instance of all of the selected ROIs in their patient representations data table(multiple AND statements given multiple selections).

Function Output:

Patient ID’s of new cohort of patients given specifications. True/False output, indicating whether there was a change in number of patients in pipeline.

Graphics:

Recalculation of the graphics for toxicity with the new patient ID’s.

ICD9:

User Option:

User given a list of the ICD9 codes present in patient population, may select multiple options.

Function Input:

ICD9 Selection list and Patient ID’s in pipeline. If pipeline has not started, use all data in database.

Query details:

Given the list of input, the current patient IDs in the pipeline will be narrowed down by selecting the patient IDs that have an instance of any of the selected ICD9 codes in their patient information data table(multiple OR statements given multiple selections).

Function Output:

Patient ID’s of new cohort of patients given specifications. Output number of unique patients in current pipeline.

Graphics:

Recalculation of the graphics for toxicity with the new patient ID’s.

Age:

User Option:

User is given two boxes to enter the upper and lower bound of age. If nothing is entered in the box, then no bounds are assumed.

Function Input:

Age boundary and Patient ID's in pipeline. If pipeline has not started, use all data in database.

Query details:

Given the list of input, the current patient IDs in the pipeline will be narrowed down by selecting the patient IDs that are within the age limit of the boundaries given by the user.

Function Output:

Patient ID's of new cohort of patients given specifications. Output number of unique patients in current pipeline.

Graphics:

Recalculation of the graphics for toxicity with the new patient ID's.

Query Saving:

A button that would download a file that would contain the current filters being used. This text file should be able to be imported to obtain the same filters when the user decided to save the query.

Query Loading:

A button that will prompt the user to select a file. The file is what was obtained from the Query Saving option. Upon importing the text file successfully, the website would load the same results that were obtained when the query was saved.

Cohort Extraction:

A button that saves all the patient IDs that have been filtered out.

Double Searching:

This is an extension of discrete and continuous analysis. There should be an option called "Double Searching," which would allow the user to perform two simultaneous pipeline of filters at the same time on a single webpage. Each pipeline would have its own menu, where filters can be added and removed. The results/graphics from each of the pipeline of filters should be shown side by side. In addition there should be a button called "Copy Filters" that copies all the filters from one pipeline to another.

Installation Documentation

1) Necessary OS
The development of this application requires a Windows OS. If you do not have a Windows OS, a virtual machines with Windows OS works as well. You can set up a windows VM by following the steps on this page:
https://www.extremetech.com/computing/198427-how-to-install-windows-10-in-a-virtual-machine

2) IDE Setup
With a windows machines, to start off we would need to download the Eclipse IDE for Java EE Developers, which would be the developing environment we used for this project.
Link:
https://www.eclipse.org/downloads/packages/release/kepler/sr2/eclipse-ide-java-ee-developers

3) Download JDK
Next you would need the Java SE Development kit, you can download the most recent version by following this link:
https://www.oracle.com/technetwork/java/javase/downloads/jdk12-downloads-5295953.html

4) Application Server
Next you would need to download the Tomcat Server, which would be the server where our web application will run on. Download link is here: https://tomcat.apache.org/download-90.cgi

5) Setting up Tomcat in Eclipse
Next to add Tomcat to our IDE. With eclipse open, find and click the servers tab on the bottom and click on the line that says “click this link to create a new server”. When a windows opens, click on the version of Tomcat you installed and link to the area where tomcat was installed. Then click Finish.

6) Creating Dynamic Web Project
Find the file tab on the top left, go to new, and select new Dynamic Web Project. Enter a project name, check to have web.xml, and finish.

7) Convert to Maven Project
Right click on the directory of your project, go to configure, and then click on “Convert to Maven Project”.

8) Adding files of our Project
From the files downloaded in github. Move over all the files into their respective directory. Make sure the directory format is conserved. The files which need to be moved are all in the src directory, the WebContent directory, and pom.xml file.

9) Build the Project
After you have moved all the files over, you can now build the project. To build the project right click project directory, go to run as, and then click maven install.

10) Allow the Tomcat Server to run the project.
Right click on the server you created in part 5. Then click on Add and Remove. On the window that pops up, click on the your project name and then click add, then click finish.

11) Run the server and see the webpage
Start the server by pressing the green arrow. Open any browser and type in localhost:8080/(name of your project)

Code Documentation

File	Description
pom.xml	Used for dependency injection, which mean it contains all the libraries which the application uses.
template.xhtml	Sets up the template in design that all .xhtml files will follow.
searchpatientage.xhtml	Creates the webpage for Search Patient Age
searchdiagnosisicd9.xhtml	Creates webpage for Search DiagnosisICD9
cohortsearchcontinuous.xhtml	Creates webpage for Continuous Analysis
cohortsearchdiscrete.xhtml	Creates webpage for Discrete Analysis
doublesearch.xhtml	Creates Webpage for Double Search
styles.css	CSS file which divides up the webpage into sections and determines how they look
hibernate.cfg.xml	Used to connect to the database
search.xhtml	Creates Webpage for Search
SearchPatients.java	Performs the business logic for both continuous and discrete analysis. In addition, creates both the bar and line plots that are shown to the user. Input: User specifications Output: New Cohorts and Visualizations.
DoubleSearch.java	Performs the business logic for double searching. Sets up the divide between the two searching paths. In addition, plots the bar graphs. Input: User specifications (two different pathways) Output: New Cohorts and Visualizations.
PatientService.java	Calls the specific dao based on the values the users had inputted.

	Input: User specifications Output: Extracted Data from Database
Assessment.java	Creates a class to represent the assessment table in the database.
Patient.java	Creates a class to represent the patients table in the database.
Patientrepresentation.java	Creates a class to represent the patientrepresentations table in the database.
Regionsofinterest.java	Creates a class to represent the regionsofinterest table in the database.
AssessmentHbmDAO.java	Generates the SQL query to extract data from assessment table. Input: User specifications Output: Extracted Data from Database
RegionsofinterestHbmDAO.java	Generate the SQL query to extract data from regionsofinterest table and patientrepresentation table. Input: User specifications Output: Extracted Data from Database
PatientHbmDAO.java	Generate the SQL query to extract data from patients table. Input: User specifications Output: Extracted Data from Database
Regionsofinterest.hbm.xml	Mapping file from database table to class created in Regionsofinterest.java
Patientrepresentation.hbm.xml	Mapping file from database table to class created in Patientrepresentation.java
Patient.hbm.xml	Mapping file from database table to class created in Patient.java
Assessment.hbm.xml	Mapping file from database table to class created in Assessment.java
IPatientService.java	Interface for PatientService.java

IAssessmentDAO.java	Interface for AssessmentHbmDAO.java
IPatientDAO.java	Interface for PatientHbmDAO.java
IRegionsofinterestHbmDAO.java	Interface for RegionsofinterestHbmDAO.java