# Assignment 1

## Overview

The goal of assignment 1 is to create a graphical user interface that will allow users to read information from a MySQL database and display it as chart data. The information should be ***anything you are interested in***. For example, it could be comparing aspects of video games, weather data, processor capabilities, etc… Each student will need to register a unique data set prior to building their program.

The MySQL database should be remotely accessible. Your program must be built using Intellij, JDK 17, JavaFX .

When the application is launched, it should show a graph of information on a styled JavaFX application.



Figure - Initial launch of project shows a graph

The application must support at least 1 graph and change to a scene with a TableView object that displays all the data from the database.

bar chart showing covid cases by age
pie chart showing covid case resolutions


Figure -Project showing 2 different graphs

OR

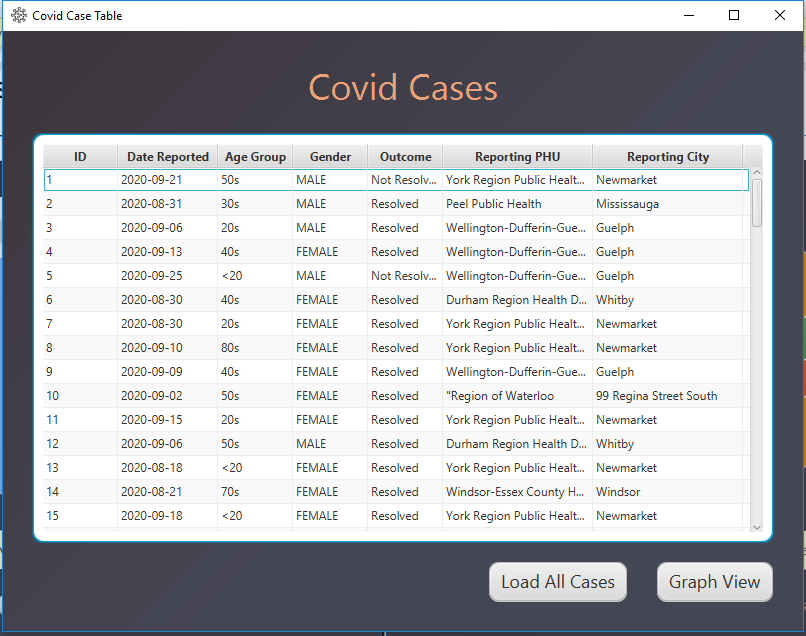
bar chart of covid cases broken down by age


Figure - A graph and a TableView Scene

## What to Do:

1. Register your unique project idea with your Professor. No duplicate projects will be allowed, so do not invest significant energy in your project until you receive approval from me.
2. When the application launches, it should display a scene with a Chart object populated from information in a MySQL database.
3. There should be a utility to change scenes and display a TableView object with objects from the database.
4. All scenes should be styled using CSS. Do not use the default grey background for everything. Add some colour, round some corners, add images to buttons, change some fonts… have fun with it!!
5. The icon on the stage should be something unique.

illustration showing a revised logo for the application in the top left of the title


Figure - Image showing changed icon

## Grading

All of your marks will be based on the rubric defined below (and visible in Blackboard).

| Criteria | Level 0 | Level 1 | Level 2 | Level 3 |
| --- | --- | --- | --- | --- |
| Code Style | 3 or more items incorrect | 2 items incorrect | 1 item incorrect | Follows Java best practices for naming and spacing conventions.  There are Javadoc style comments prior to each method in the model class. |
| The Model class(es) is well structured | The model class(es) is either not present or is not used. | The model class has poor naming conventions and/or poor instance variable data types. All instance variables must be private. | Level 1 plus all data types are logical. All method and variable names are logical and follow camelCase style.  The constructor uses the set methods. | Level 2 plus all “set” methods contain ***useful*** validation. Do not use isBlank() (or equivalent) for validation in all set methods. Validate against known lists, number ranges, etc… |
| User Experience | When the program launches, an exception is thrown or the Chart is not populated. | Functions, but it visually not appealing | Generally easy to use, but inconsistent in design or has minor formatting issues such as truncated text or areas that do not line up | There is a polished user experience. The application is intuitive and easy to use.  There is consistency throughout the app with font choices, colour schemes, & application size.  CSS with at least 6 items is used to style the app.  There is a new icon that matches the theme of the app |
| When launched, a Chart object is populated with data | The first scene does not contain a graph. | 2 items could use improvement | 1 item could use improvement | A chart is populated from a DB. Chart labels and legend are easy to read/visible. If labels/legend are visible, but a DB query is not used, it is level 2. |
| Database | There is no remote database connectivity in the project and/or the necessary SQL was not provided for a local DB | There is local database access only, but ALL necessary SQL statements are provided in a file that can be run in top to bottom. | There necessary data has been populated on a remote DB. Code reflects all necessary info to connect to the database (i.e. URL, port, user name and password and necessary maven imports) |  |
| Populating TableView | A DB query does not exist or creates an error | The DB query returns a valid ResultSet, but it is not used to populate the TableView | Level 1 plus it is used to populate a the TableView object | Level 2 plus queries are used to populate the chart and TableView objects. The Connection, Statement and ResultSet are closed at the completion of the method.  All columns are populated with correct data. |

## Reverse Rubric

In the event that the project submitted does not follow Java programming best practices, 1 mark will be deducted for EACH infraction noted below:

* -1 mark for any class name that does not start with a capital letter and follow camel case practices.
* -1 mark for any variable or method name that starts with a capital letter.
* -1 mark for each line of code that must be change to make the project compile. A maximum of 10 lines will be changed at which point assessment of the project will stop
* There is a 20% penalty per calendar day for late submissions

## Recommended Steps and timing

|  |  |
| --- | --- |
| Lab week 5 | Create the DBUtility class with methods that can access the data |
| Lab week 6 | Code should be able to launch a JavaFX scene with a working Chart object |
| Full Submission (week 7) | All requirements as defined by this document |