London COVID-19 Statistics

Our group’s name is Bojo’s Brogrammers. The members of the group are:

* Maiwand Nikmal - 23083460
* Muhammed Keeka - 23071383
* Rishi Hundia - 23014784
* Finn Corney - 23139796

# Description and Functionality

In this project, we have produced an application that presents data surrounding the COVID-19 pandemic's impact in and around London. The project has two windows: the main window, and the borough data window.

### The Main Window

The main window is where most of the user's interactions with the application take place. The main window consists of a space for the current panel in the middle, buttons on the bottom to navigate between panels, and two date picker objects to select the date range that the user would like to look at. Upon the application being run, the main window opens, and is set on the welcome panel.

##### The Welcome Panel

The welcome panel explains to the user the purpose of the application, as well as outlines to them our suggestions for the date range that they select. This is since data in all fields is not available for all dates. We chose not to restrict the date range to the dates which have complete data, but this does mean that there are some date ranges where some statistics will be unavailable.

The main window shows this welcome panel initially at the start of the program, but then again if the dates are changed to an invalid date range at any point during the running of the program. Whenever there is an invalid date range / no date range, the buttons that are used to navigate between panels are disabled until a valid date range is then selected. Once a valid date range has been selected, the welcome panel also displays this date range.

##### The Map Panel

The second panel in the application is the map panel. This panel has a background of an image of London (this provides no functionality, but we thought it looked cool). In front of that is the representation of the boroughs of London, which we modelled off the graphic provided in the task sheet.

The hexagons that represent boroughs are coloured according to the number of deaths in the boroughs in the selected date range. The borough or boroughs with the most deaths are coloured in red. The other hexagons are all coloured relative to this borough. If that have more than two thirds of the cases of the borough with the most cases, then they are coloured in dark orange. If they have between one and two thirds of the cases of the borough with the most cases, then they are coloured in light orange. If they have less than one third of the cases of the borough with the most cases, then they are coloured in green. If it is not possible to calculate the number of deaths in a borough due to incomplete data, the hexagon is set to the same colour as the background (such that it appears transparent).

Upon clicking on a borough, the borough window opens (this is addressed later in the report). In addition to this, the borough is shown in the main window using Google Maps integration. When submitting this project, our Google Maps API key has been removed from the program. We have left the constant in the MapPanelController class, so you can enter your API key and it will load.

##### The Statistics Panel

Our statistics panel shows five statistics, these being:

* The total deaths across all London boroughs
* The percent change in visits to parks (an average of GMR data)
* The percent change in visits to public transport hubs (an average of GMR data)
* The average amount of cases per borough
* The average amount of cases per day

We did two statistics around average cases due to unclear wording in the task sheet.

##### The Graph Panel

The fourth panel in our application is the graph panel. This has two buttons either side, and a graph in the middle. The user can change between a graph of total deaths during the selected period, and a graph of total cases. These graphs were made using JavaFX components and so introduce no additional dependencies.

### The Borough Window

Upon clicking on a borough in the map panel, the borough window opens. The title of the window is the full name of the clicked borough. In the window, the records are displayed in a scrollable list, and at the top of the window is a drop-down which allows the user to pick a field to sort the records by. The default sorting is by date.

### Other Details

We created an SQLite database and imported all the data from the CSV into this. We did this because it allowed for better fetching of data and calculation of statistics (through the ability to use SQL's aggregate functions). In doing this, the entire CovidDataLoader class was rewritten. One additional dependency of the JDBC for the database has been added.

One issue that we came across was that for many entries in the data provided, there were large amounts of data missing from various fields. To combat this, all ints were changed to type Integer, and missing cells were set to null in the database.

# Unit Testing

In our migration to an SQLite database, the entire CovidDataLoader class was rewritten. None of the original methods or lines of code are there, not even any of the same library imports. Therefore, I consider this equivalent to a new class entirely of our own design.

The CovidDataLoader class contains fourteen methods that fetch data from the database. Each of these methods has a corresponding test method in the class CovidDataLoaderTest. We have attempted to do comprehensive testing, by testing boundary data, testing for nulls in places where data is not present, testing at some points against known statistics from the database. Thanks to the test class, we can be sure that all 36399 records are present in the database, and that we should not experience any runtime errors during the running of the application. The tests are as follows:

* assertRecordCount(): checks that all 36399 records are present in the database
* assertCasesAllBoroughs(): checks that the number of cases across all of London between two dates are fetched correctly
* assertCasesSpecificBorough(): checks that the number of cases in a specific borough between two dates are fetched correctly
* assertMaxCases(): checks that the maximum number of cases in any London borough between two dates are fetched correctly
* assertDeathsAllBoroughs(): checks that the number of deaths across all of London between two dates are fetched correctly
* assertDeathsSpecificBorough(): checks that the number of deaths in a specific borough between two dates are fetched correctly
* assertMaxDeaths(): checks that the maximum number of deaths in any London borough between two dates are fetched correctly
* assertParksGMR(): checks that the percentage change in visits to parks between two dates is fetched correctly
* assertTransitGMR(): checks that the percentage change in visits to public transport hubs between two dates is fetched correctly
* assertEarliestDate(): checks that the date of the earliest entry in the database is correct
* assertLatestDate(): checks that the date of the latest entry in the database is correct
* assertCasesPerDay(): checks that the cases per day over a range of dates is fetched correctly
* assertDeathsPerDay(): checks that the deaths per day over a range of dates is fetched correctly
* assertBoroughData(): checks that the CovidData objects are being fetched correctly

# Contributions

In addition to the listed contributions below, all members worked across classes cleaning up code, fixing code, and ensuring that the different pieces of the code and GUI work together seamlessly.

### Muhammed Keeka

* Worked on the documentation.
* Wrote the report.
* Set up the SQLite database and ensured that all data from the CSV is present in the database.
* Rewrote the CovidDataLoader class.
* Wrote the test class for the CovidDataLoader.
* Implemented the statistics.
* Created the borough data window.
* Worked on the graph panel.

### Maiwand Nikmal

* Worked on the documentation.
* Created the main window, both the FXML file as well as the controller.
* Created the welcome panel, both the FXML file as well as the controller.
* Designed the statistic panel and class structure.
* Designed the graph panel.

### Rishi Hundia

* Created the map panel, both the FXML file as well as the controller.
* Created the map of London.
* Created the Google Maps integration.
* Worked on the graph panel.
* Created the heatmap.

### Finn Corney

* Worked on the documentation.
* Worked on the FXML files.
* Worked on the main window.