Project Document

1. Personal Information:

Project: Data Dashboard – Programming Studio A Project.

Name: Nguyen Manh Hung

Degree Program: Data Science – First Year

Date of submission: 6th of April, 2022

2. General description and user interface:

The moderate level is completed:

This project is meant to implement an interactive data-dashboard of the Covid-19 situation in Finland. This includes scatter plot, column plot and pie chart with legends – which are all interactive to user (clicking/hovering on a single data point will show more information about the data point).

The program interface contains in 3 main components: Diagram Area, Data Table and File Handler. Users can hide/show/resize these components.

In File Handler, users can upload 2 of the following file types:

- .json file: (Please only take .json file from the folder "COVID files for testing") Data files contain numerical data, which contains in multiple dates. Users can select which time interval to display to the Dashboard using the Date chooser in File Handler. You can select which time interval to be visualized as a 7-day interval by typing in the start date. (For example, if the input start date is 1/1/2021, the end date will be calculated by the program: 7/1/2021) (The default date is 06/09/2021)
- .hihi file: (Customed file type) (Please only take .hihi file created by the program) Data files contain categorical data to display the saved state of the Dashboard at some specific date. (static) Note that as the dashboard is static, the functions Date Chooser and Save file will not be presented as they no longer make sense.

Any other file type will not be accepted.

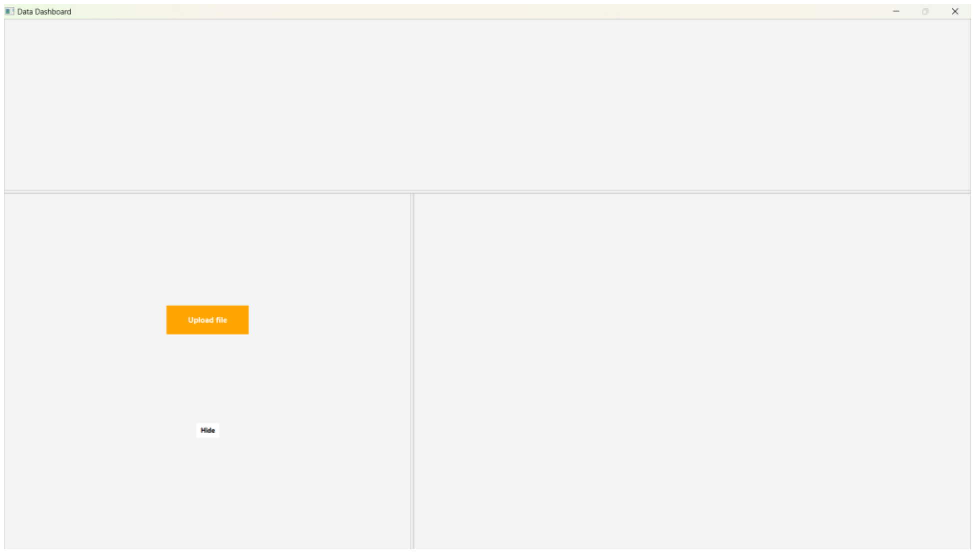
From file handler, users can save using the button the current 7-day interval state of the Dashboard into .hihi files, which can later be uploaded to review.

In Data Table, users can see calculations of the data of the chosen 7-day interval of the sum of deaths and cases on big cards and min/max/average/standard deviation on smaller cards.

In the Diagram Area, 3 diagrams are displayed using the data of the uploaded 7-day interval scatter plot, column plot, pie charts along with legends. The diagrams are interactive, hovering on a single data point will show more information about the data point. Users can choose to add/remove/duplicate/hide charts from the Diagram Controller.

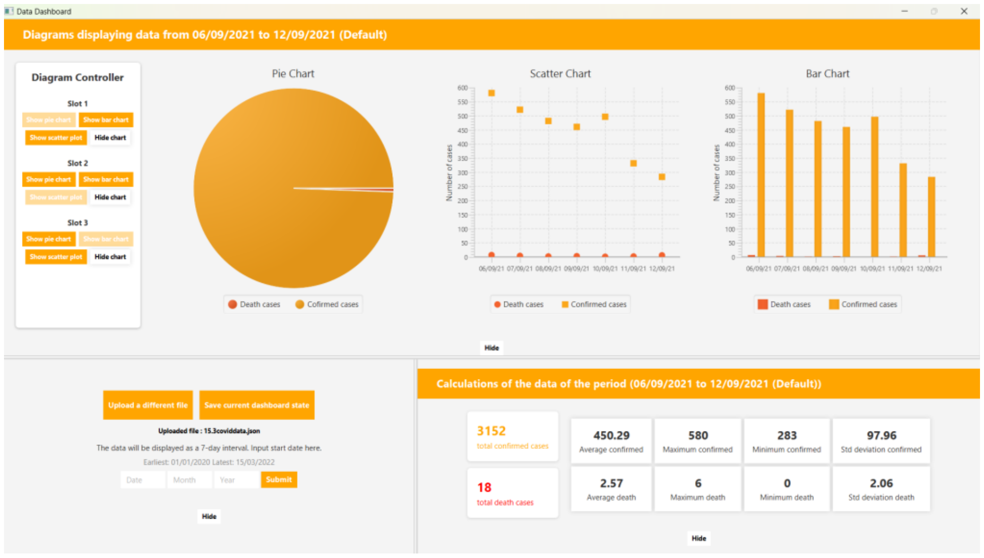
The data regarding the covid-19 situation will be fetched from the website of European Centre for Disease Prevention and Control: https://www.ecdc.europa.eu/en/publications-data/data-daily-new-cases-covid-19-eueea-country

Below is the initial state of the program.



The program starts with user uploading a file (".json" or ".hihi").

If users upload .json file (given in the "COVID files for testing" folder):



Diagrams displaying data from 06/09/2021 to 12/09/2021 (Default)

Uploaded file : 15.3coviddata.json

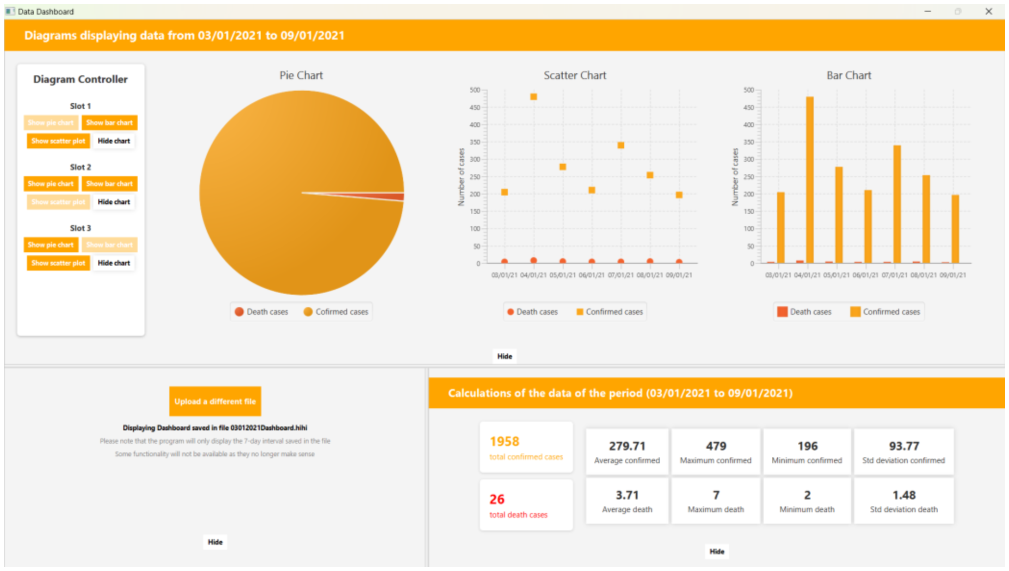
The data will be displayed as a 7-day interval. Input start date here.

Earliest: 01/01/2020 Latest: 15/03/2022

Calculations of the data of the period (06/09/2021 to 12/09/2021 (Default))

3152 total confirmed cases	450.29 Average confirmed	580 Maximum confirmed	283 Minimum confirmed	97.96 Std deviation confirmed
18 total death cases	2.57 Average death	6 Maximum death	0 Minimum death	2.06 Std deviation death

If user uploads .hihi file (saved from the program by users):



Diagrams displaying data from 03/01/2021 to 09/01/2021

Displaying Dashboard saved in file 03012021Dashboard.hihi

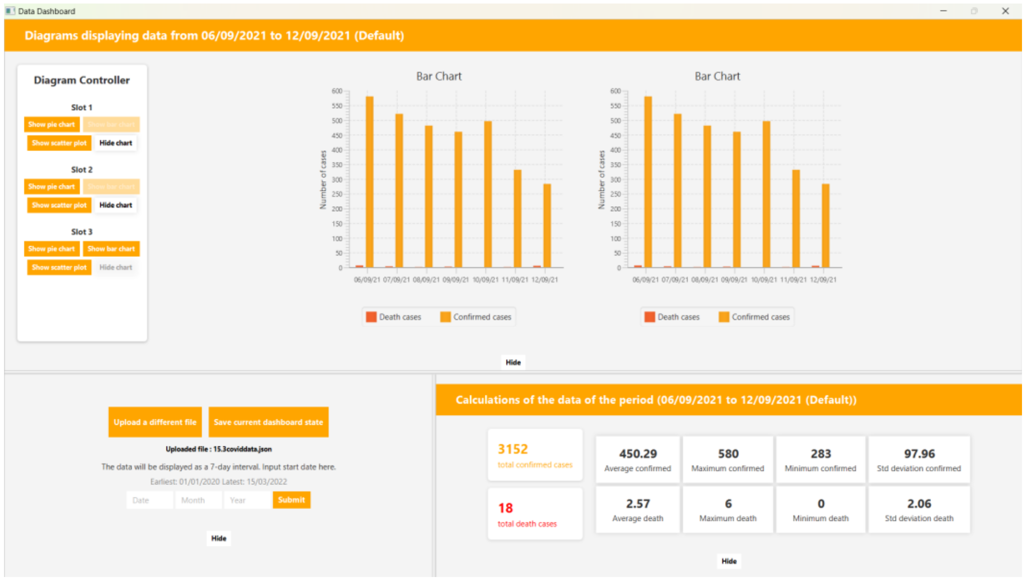
Please note that the program will only display the 7-day interval saved in the file

Some functionality will not be available as they no longer make sense

Calculations of the data of the period (03/01/2021 to 09/01/2021)

1958 total confirmed cases	279.71 Average confirmed	479 Maximum confirmed	196 Minimum confirmed	93.77 Std deviation confirmed
26 total death cases	3.71 Average death	7 Maximum death	2 Minimum death	1.48 Std deviation death

User can choose to show/hide/duplicate diagrams:



Diagrams displaying data from 06/09/2021 to 12/09/2021 (Default)

Uploaded file : 15.3coviddata.json

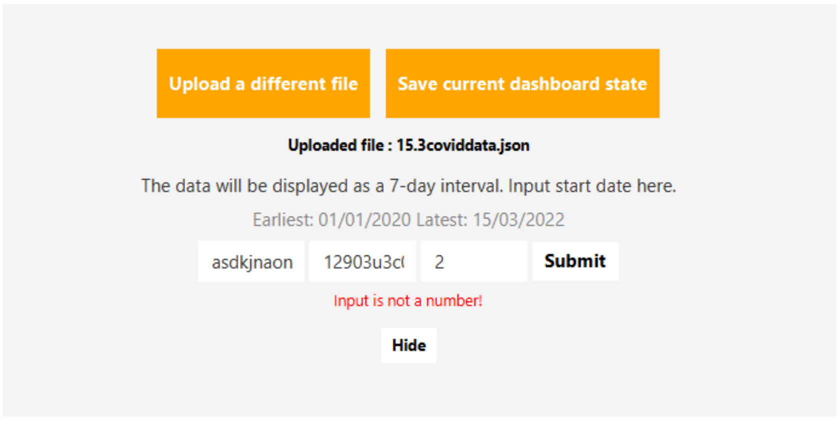
The data will be displayed as a 7-day interval. Input start date here.

Earliest: 01/01/2020 Latest: 15/03/2022

Calculations of the data of the period (06/09/2021 to 12/09/2021 (Default))

3152 total confirmed cases	450.29 Average confirmed	580 Maximum confirmed	283 Minimum confirmed	97.96 Std deviation confirmed
18 total death cases	2.57 Average death	6 Maximum death	0 Minimum death	2.06 Std deviation death

Example of a faulty input being handled:



Uploaded file : 15.3coviddata.json

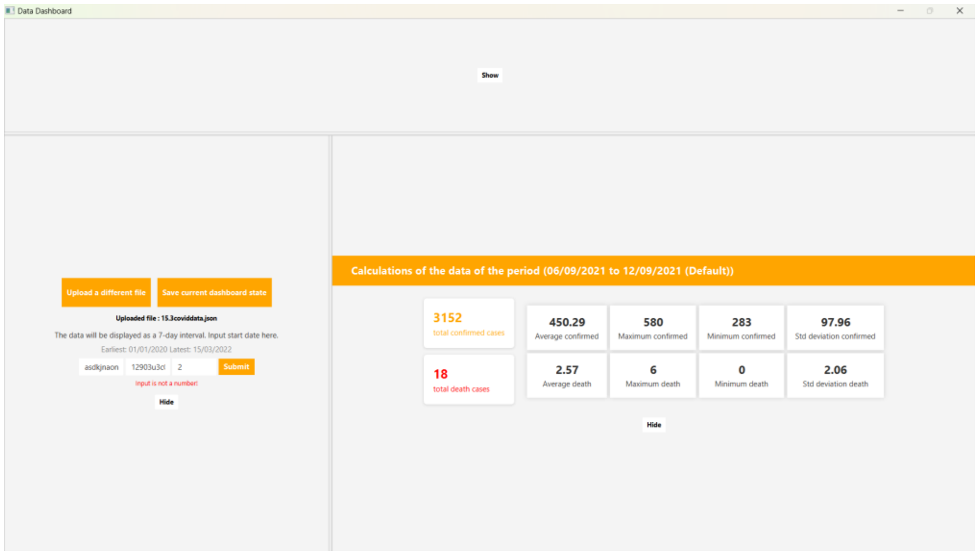
The data will be displayed as a 7-day interval. Input start date here.

Earliest: 01/01/2020 Latest: 15/03/2022

asdkjnaon | 12903u3c(| 2

Input is not a number!

User can choose hide/show/resize components:



Uploaded file : 15.3coviddata.json

The data will be displayed as a 7-day interval. Input start date here.

Earliest: 01/01/2020 Latest: 15/03/2022

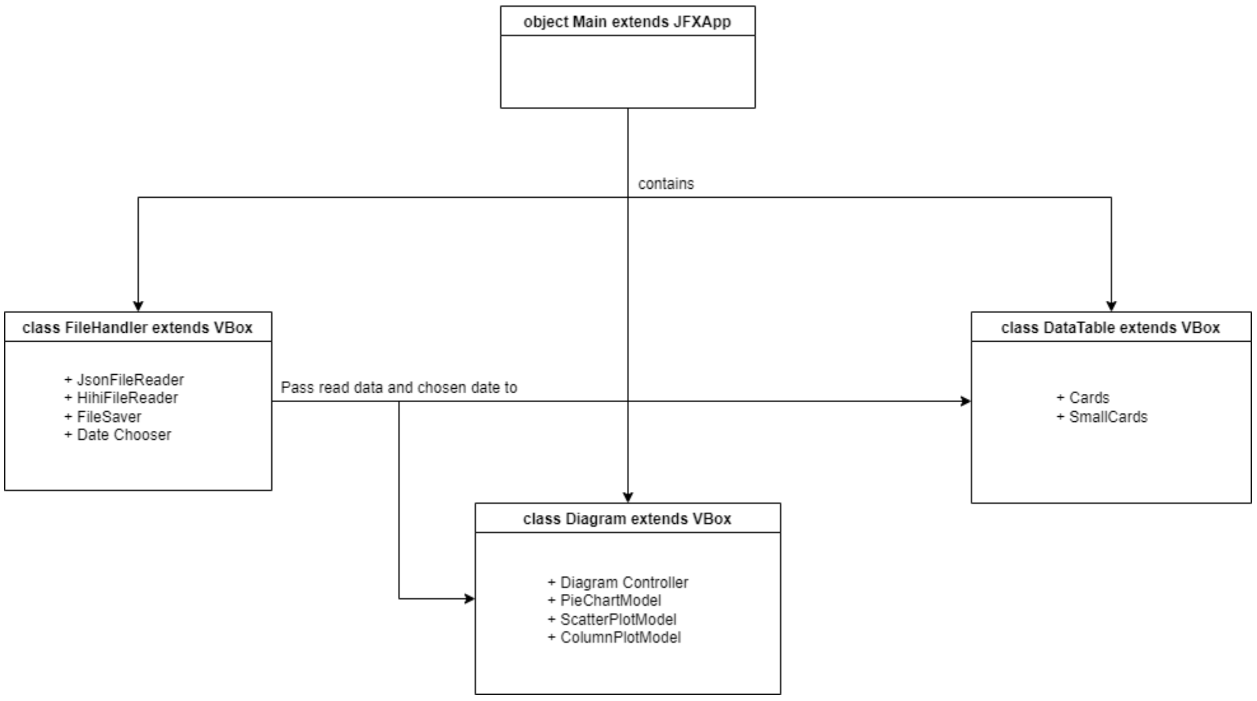
asdkjnaon 12903u3cl 2

Input is not a number!

Calculations of the data of the period (06/09/2021 to 12/09/2021 (Default))

3152 total confirmed cases	450.29 Average confirmed	580 Maximum confirmed	283 Minimum confirmed	97.96 Std deviation confirmed
18 total death cases	2.57 Average death	6 Maximum death	0 Minimum death	2.06 Std deviation death

3. Program Structure:



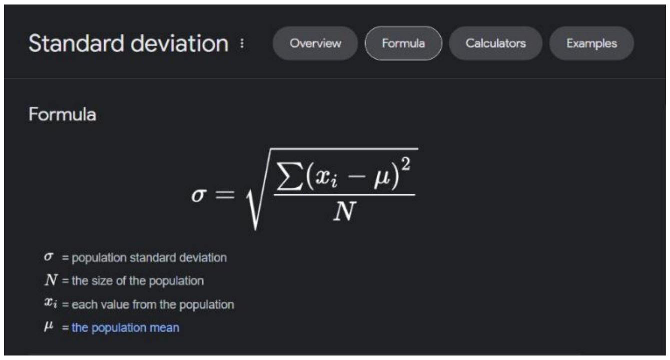
The program contains in 3 main components: class FileHandler, class DataTable and class Diagram. These 3 components will be rendered in object Main to display to the GUI. When the user uploads a file to FileHandler and choose a 7-day interval, the program will render a new Diagram and DataTable with the uploaded data and the start date being passed as parameters.

This program structure has proven its effectiveness: it gives me great control over the program, ensuring that each class will only focus on one main part – which makes the work way easier and more controllable while coding and debugging.

4. Algorithms:

For the calculations of sum, min and max, it's basic knowledge. I simply find the Sum of each datapoint in the array, find the minimum-valued and maximum-valued datapoint. These are done using built-in function of Scala.

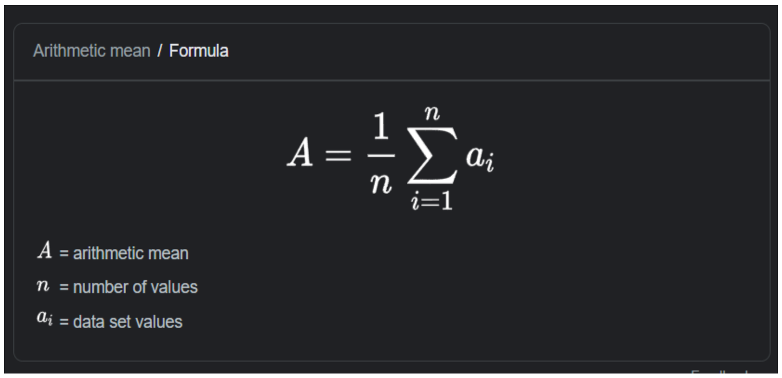
For the calculations of average and standard deviation of the data, I use basic Mathematics formula:



Formula

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}}$$

σ = population standard deviation
N = the size of the population
x_i = each value from the population
μ = the population mean


Arithmetic mean / Formula

$$A = \frac{1}{n}\sum_{i=1}^{n} a_i$$

A = arithmetic mean
n = number of values
a_i = data set values

The mean can be simply calculated as the Sum divides by the size of the data array.

The standard deviation calculation is more complicated: I map each x_i of the data array into $(x_i\ - \text{mean})^2$, then take the square root of the sum of the mapped data array divided by the size of the data array.

5. Data Structures:

Data are contained using Seq. The reason I use Seq is because it is default to Scala, and I personally find Seq simple to use.

The data is of the type Seq[Tuple4(String, Int, Int, Int)], where ._1 is the date, ._2 is the confirmed cases, ._3 is the death cases, ._4 is the total population of Finland. Using this structure, I can easily access to each data point.

6. Files and Internet access:

- .json file: (Please only take .json file from the folder "COVID files for testing") Data files contain numerical data, which contains in multiple dates. Users can select which time interval to display to the Dashboard using the Date chooser in File Handler. You can select which time interval to be visualized as a 7-day interval by typing in the start date. (For example, if the input start date is 1/1/2021, the end date will be calculated by the program: 7/1/2021) (The default date is 06/09/2021) These files are obtained from the data source above.
- .hihi file: (Customed file type) (Please only take .hihi file created using the program) Data files contain categorical data to display the saved state of the Dashboard at some specific date. (static) Note that as the dashboard is static, the functions Date Chooser and Save file will not be presented as they no longer make sense.
  These files are obtained from the program and are user-created files.

You can find both file types stored in “COVID files for testing” folder. Please note that you can create your own .hihi files by saving the current state of the Dashboard.

7. Testing:

To test the main functioning of the program, which includes in the File Handler and the Date Chooser, I uploaded many versions of .json and .hihi files to see if the Diagram and the Data Table receive the right data. I also upload the wrong file type to see if that case is handled. To test the Date Chooser, I entered faulty input to check if all erroneous inputs are handled, if the right inputs are correctly passed to Diagram and Data Table.

To test other UI functioning, I simply operate them to see if any bug occurs.

8. Known bugs and missing features:

The program is not at all designed to handle erroneous files. Uploading erroneous files (or any intentionally modified files to break the program) will cause the program to crash. I have not found a way to handle this.

Please, only use the files given in the project or the files created by the program.

9. Best sides and weaknesses:

Best side: The programs user interface looks good. I spend quite some time designing it.

Weakness: The program is not designed to work with other .json file, but only with the given .json files from European Centre for Disease Prevention and Control.

10. Deviation from plan, realized process and schedule:

The only thing that is deviated from the plan is the amount of time the project took me to finish. Initially, it was estimated that this project took me 2 weeks. However, it took me 3 weeks to complete – I have some trouble with implementing Tooltips (somehow, I only found information regarding Java implementation of Tooltips) and had to spend a long time asking the authors about this in the GitHub repo of the Scala language.

The order of my work progress: Building basic layout → building functioning of the program → completing the UI (took most of the time) → debugging.

11. Final evaluation:

I personally find this project hard to complete, as there is very limited information regarding ScalaFX implementation on the Internet. I spent most of my time looking for some implementation of something that I am not even sure about its existence in ScalaFX.

For future improvements, they could be: Implementation of KeyFrame so that the animation can be smoother, fetching data directly from Internet, ability to pick which colors for datapoints for diagrams, ability to select datapoints using a tool,…

Looking back at my program, I kind of feel proud of it – although it’s not perfect yet (still could not handle erroneous files), but it’s my work and my design. For the current program, I would not change a single thing in the process if I started the program again, and I feel that the project is good enough – it completed all the requirements of the moderate level (and some of the hard level), bonus a good-looking user interface.

12. Appendixes and References:

Gitlab project link: https://version.aalto.fi/gitlab/nguyenh60/1022029_project

Most of my questions are answered by the great people on StackOverFlow.

Please refer to section 3 if you want to see examples of the program being in used.