

CSE 360: GROUP 14 TEAM PROJECT

Project Management

Process Model and Architecture Model

For the project, we utilized a waterfall model as our process model. Given that each component of the project built off the prior component, we concluded that the waterfall model would provide us with the best structure as far as planning our process was concerned. As such, we followed through with the project in a linear fashion. We created a schedule and from there, we planned out the course of this project. To facilitate this, we used a Gantt chart to scheme and monitor our progress. How we structured it was based on the aforementioned waterfall model. We did it in a fashion such as this: Design → Implementation → Report and Testing. Under our progress section, the illustration of the waterfall model can be seen in our Gantt chart.

As for the architecture model we used, we found it proper to use the model-view-controller. As the project itself required a lot of GUI implementation, the model-view-controller was perfect for this. The project itself was essentially an implementation of it. Here is a breakdown:

Model: Program that stores and manages the data that is inputted by the controller and notifies the view upon any changes to it

View: GUI application that features buttons and can display data from the model that is introduced from the controller

Controller: User can select buttons and input information that can be displayed, manipulating the view, and in turn, the model

Roles

To determine who would be doing what during the project, we maintained a correspondence with one another using a Discord chat. Discussing amongst one another, we figured out our roles during this project.

Kekaiolu assisted with debugging code, having contributed to the main file and the “Load” feature, while also helping to decide on what would be done. He also assisted in writing the project report, specifically the Project Management section, as well as formatting it.

Daniel started the program, having created the ground base for the coding project. He contributed to creating the About feature, as well as creating the Load feature, completing most of the code for it. He also contributed in creating the Visualize feature and completing it.

Kang Yi contributed by implementing the Add button and the Save button, as well as helping to debug and work through implementing the Visualize feature. He also created the Requirements section of the report, defining user and system requirements, as well as creating user stories and the use case diagram.

David Ruan created the schedule, UML diagrams, implemented parts of the GUI/add data function, testing, and testing documentation.

Group 14

Project Start:

1

[illegible]

Create user stories or use cases to describe the requirements of the project. (Remember to include stories/cases that indicate how the program should behave for valid input, invalid input, etc)

A. User Requirements

- ## B. System Requirements

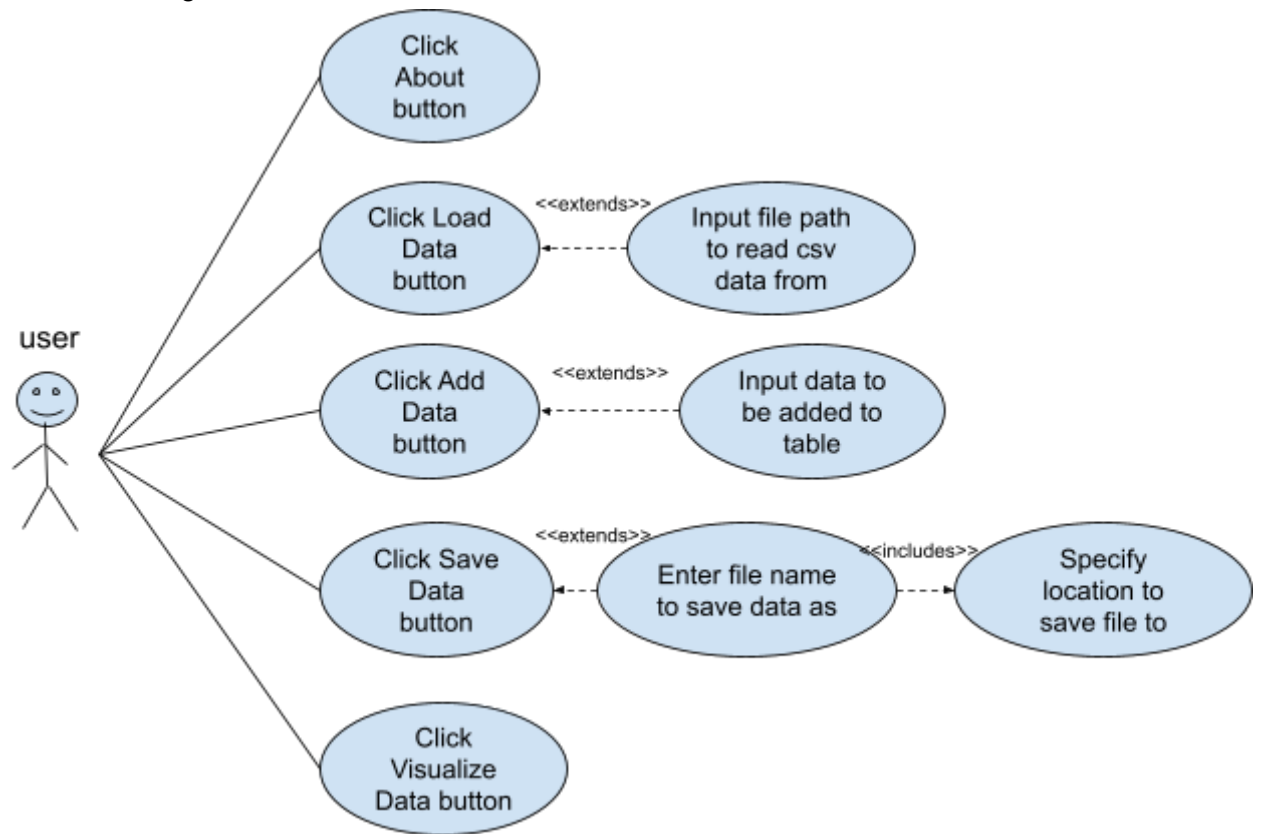
1. The system will use GUI libraries in order to add necessary panels for organizing content to be displayed.

2. Interactable buttons will be created by attaching event listeners to execute methods when clicked.
3. The [about] button will display labels containing the names of the people working on this project.
4. The [load data] button will have a textfield which users can input a file path to load csv data from, and then parse the file for the data, and display the table. This table should also behave like a spreadsheet, allowing users to modify the entries in it. Additionally, loading multiple csv files will append the new data to the current table, allowing multiple csvs to be combined.
5. The [add data] button will display a number of text fields corresponding to every column of data in the table, and allow the user to type data into these text fields to be appended to the table. This will not modify the csv file, only the table loaded in the application. The data will be passed into a method attached to the table class, which appends the data given.
6. The [save data] button will display a text field which the user can enter the name the file should be saved as. Upon clicking the save button, it will open up a prompt which allows the user to select the directory to which the file should be saved, and then save the file. Before the saving is finalized, the table will call a method to convert its contents to a string in the format of a csv, and this string will be written to the file to be saved.
7. The [visualize data] button will have two different tabs shown for the purpose of allowing the user to choose what data to display. The data will be read from the table, and a bar chart will be generated and displayed to show this data. The two types of charts to be created is one displaying the number of vaccinations per location, and one displaying the number of vaccines per manufacturer.
8. There needs to be a table model object that is initialized at the start, but will remain empty and cannot be added to until some data has been loaded in with the [load data] button. This table will contain a DefaultTableModel from the javax.swing library, which will be the basis for storing and accessing data.

II. User Stories

- A. As a user, I want to be able to load in data on Covid vaccinations, and view the data in a table or displayed with a graph.
- B. As a user, I want to be able to modify and add information to the table, and have my changes be saved to a file.

III. Use Case Diagram

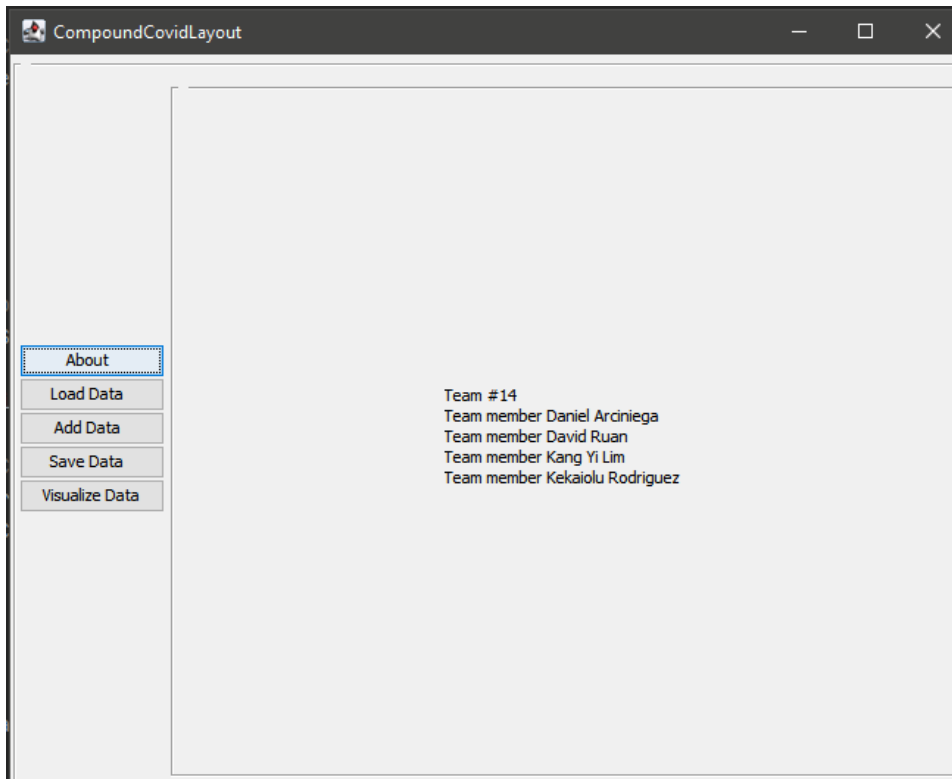


Class Diagram

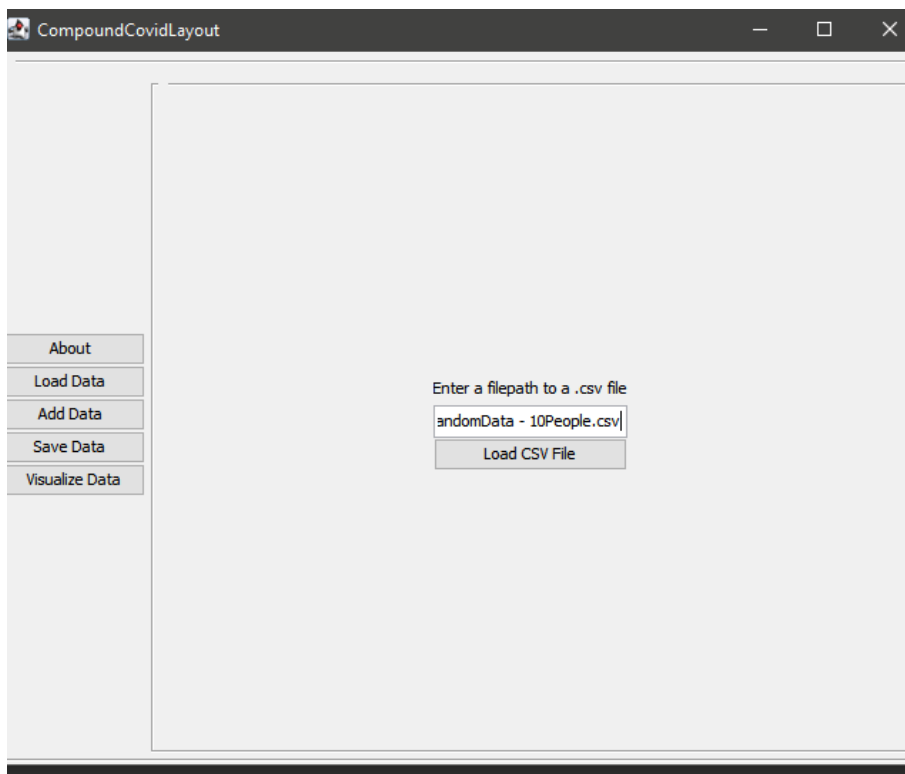
The diagram was too small to read, so we have included a PDF of the diagram in the zip file titled "UMLDiagram.pdf".

Report Execution

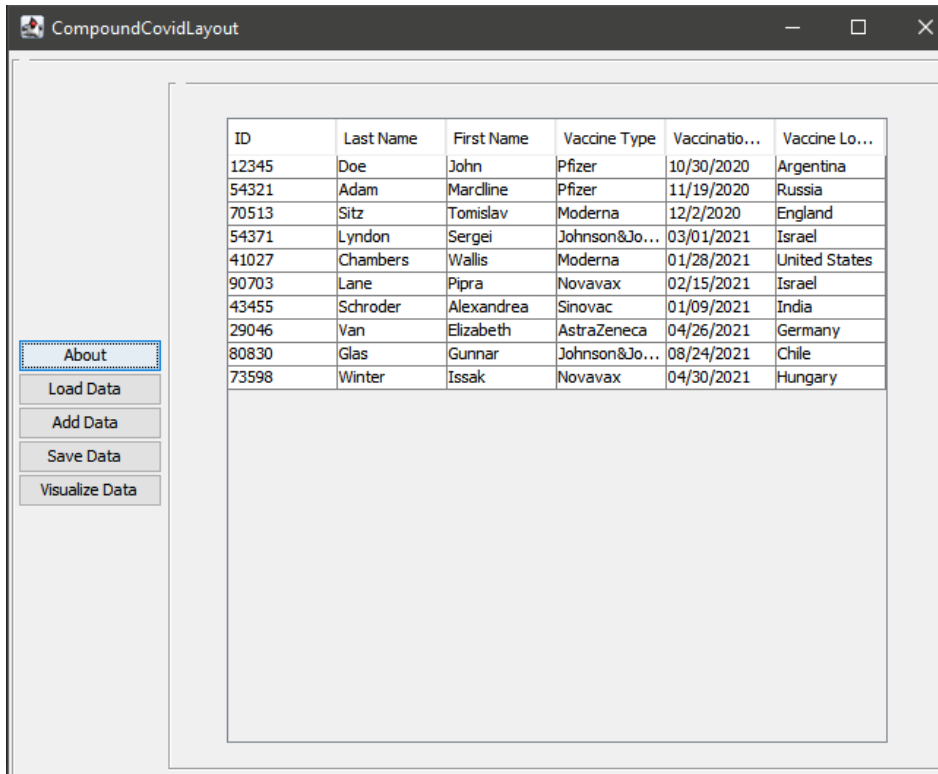
Clicking “About”:



Clicking “Load Data”:



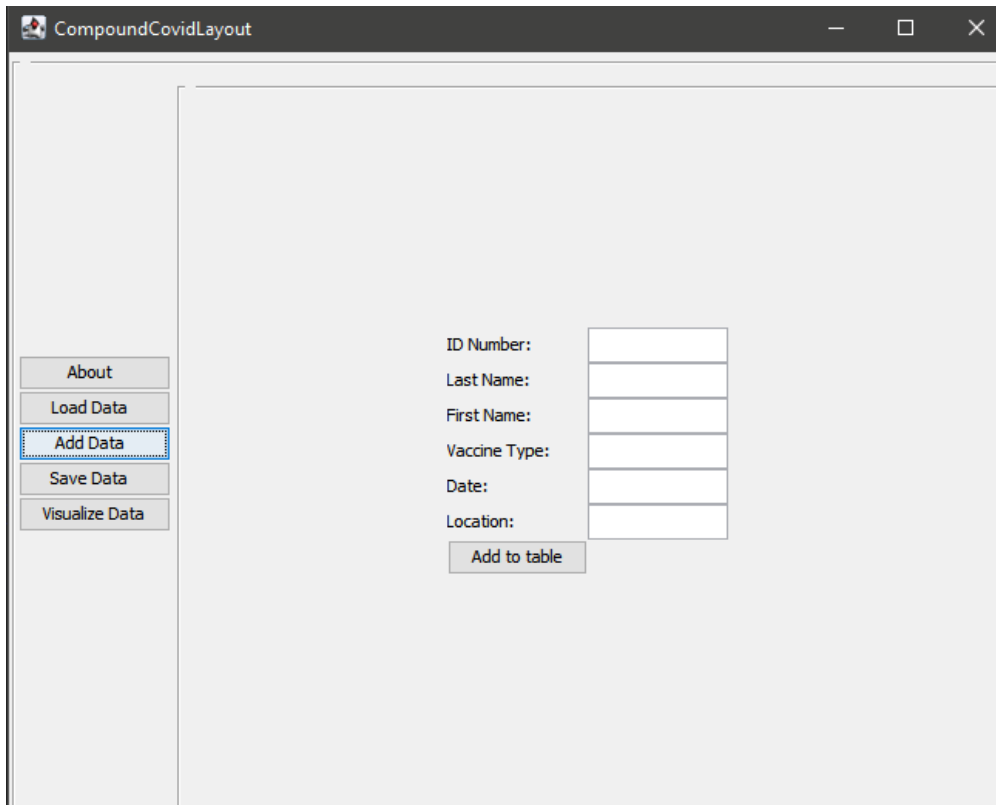
Once we load the csv file:



The screenshot shows the 'CompoundCovidLayout' application window. On the left is a sidebar with five buttons: 'About', 'Load Data', 'Add Data', 'Save Data', and 'Visualize Data'. The 'About' button is highlighted with a blue dashed border. The main area of the window displays a table with the following data:

ID	Last Name	First Name	Vaccine Type	Vaccinatio...	Vaccine Lo...
12345	Doe	John	Pfizer	10/30/2020	Argentina
54321	Adam	Mardline	Pfizer	11/19/2020	Russia
70513	Sitz	Tomislav	Moderna	12/2/2020	England
54371	Lyndon	Sergei	Johnson&Jo...	03/01/2021	Israel
41027	Chambers	Wallis	Moderna	01/28/2021	United States
90703	Lane	Pipra	Novavax	02/15/2021	Israel
43455	Schroder	Alexandrea	Sinovac	01/09/2021	India
29046	Van	Elizabeth	AstraZeneca	04/26/2021	Germany
80830	Glas	Gunnar	Johnson&Jo...	08/24/2021	Chile
73598	Winter	Issak	Novavax	04/30/2021	Hungary

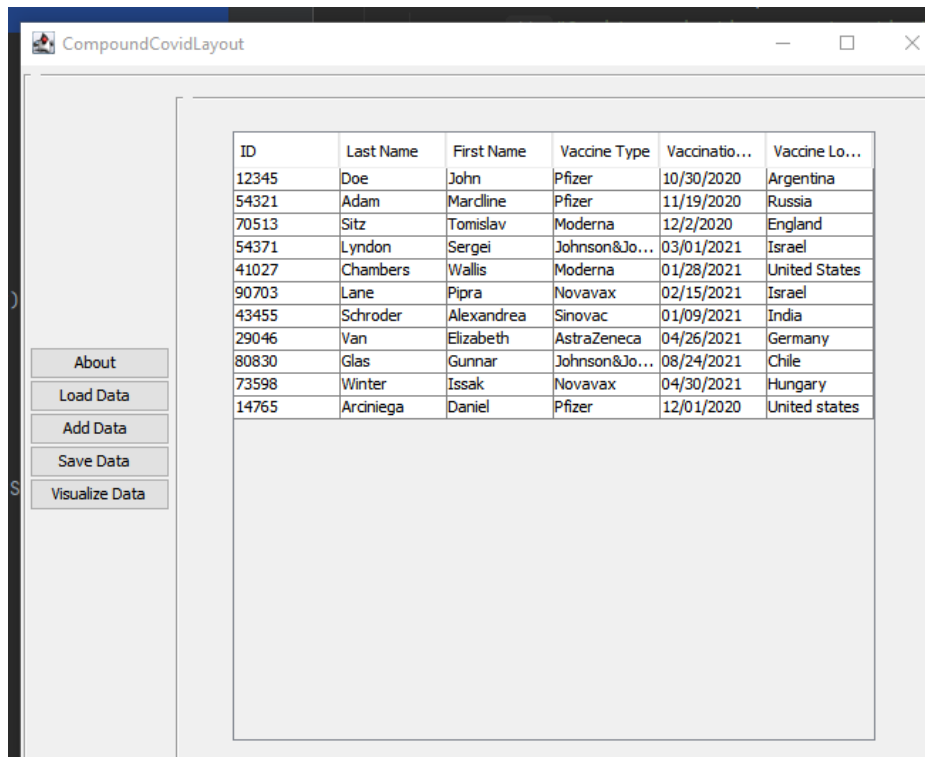
Once we click the add gui:



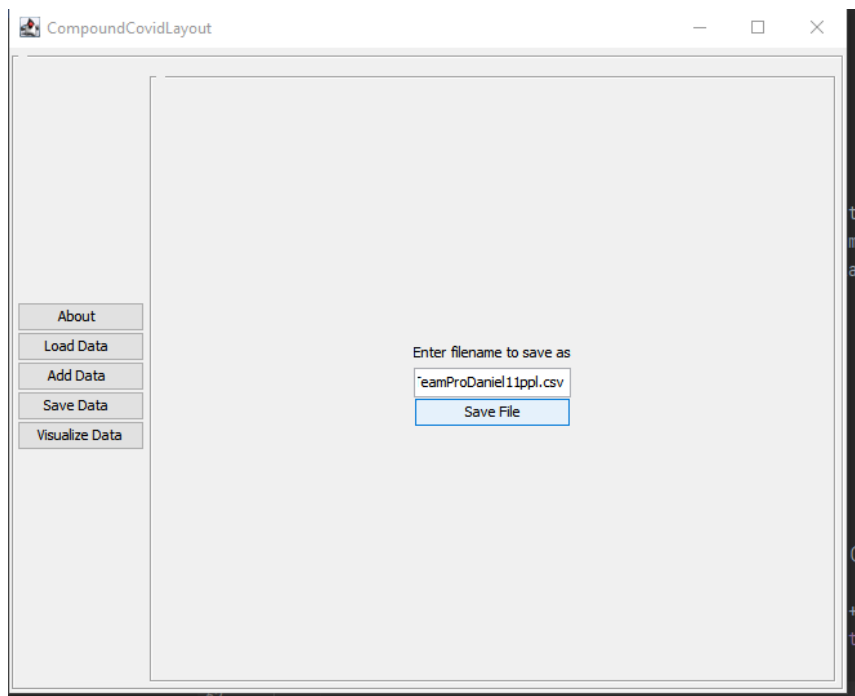
The screenshot shows the 'CompoundCovidLayout' application window with the 'Add Data' button highlighted in the sidebar. The main area of the window contains a form for adding new data. The form consists of the following fields and controls:

- ID Number:
- Last Name:
- First Name:
- Vaccine Type:
- Date:
- Location:
-

Once we click the add to table gui after adding my name(Daniel Arciniega):



Once we click the Save Data Gui and enter data in the textbox:



Once we save file it asks for a file location and we can then create a new file with user given input data:

AutoSave On | TeamProDaniel11ppt

File Home Insert Page Layout Formulas Data Review View Help

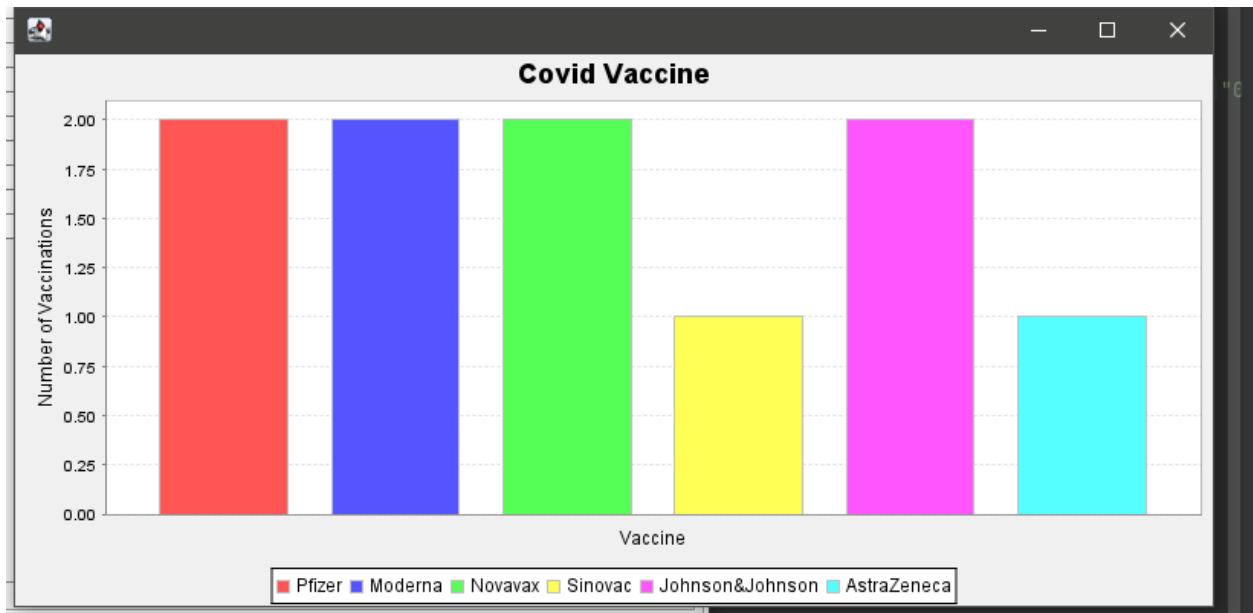
Clipboard Font Alignment Number Styles Cells Editing Analysis Sensitivity

POSSIBLE DATA LOSS Some features may be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it as an Excel file format.

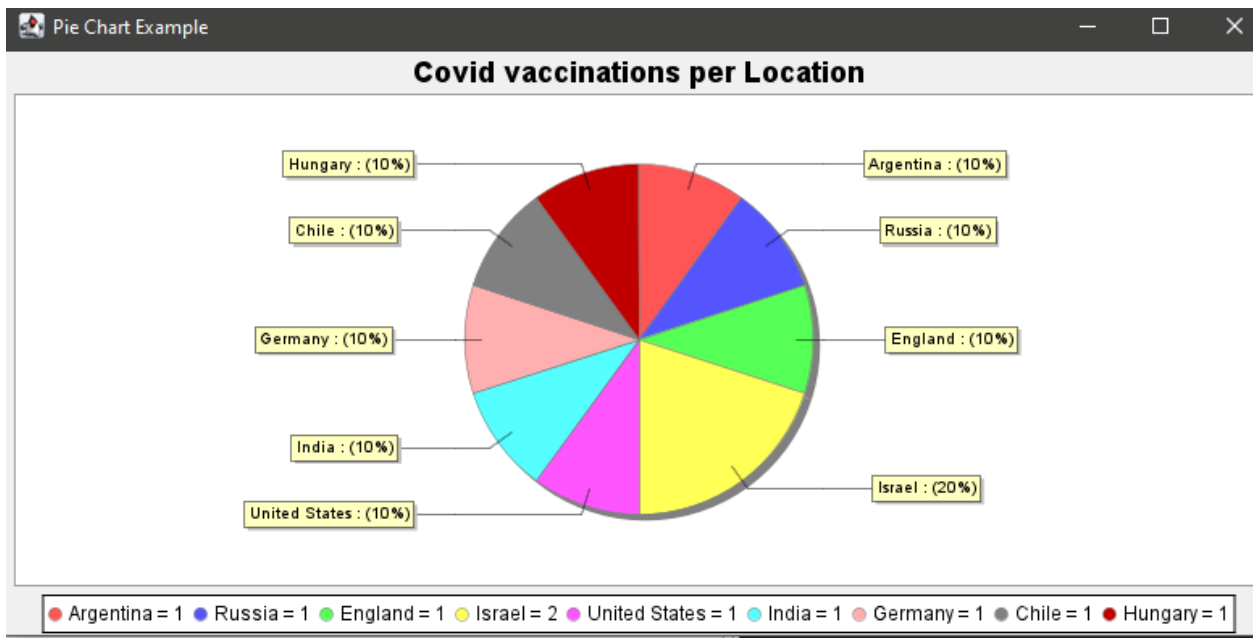
A1 ID

ID	Last Name	First Name	Vaccine Type	Vaccination Date	Vaccine Location
12345	Doe	John	Pfizer	#####	Argentina
54321	Adam	Marciline	Pfizer	#####	Russia
70513	Sitz	Tomislav	Moderna	#####	England
54371	Lyndon	Sergei	Johnson&	3/1/2021	Israel
41027	Chambers	Wallis	Moderna	#####	United States
90703	Lane	Pipra	Novavax	#####	Israel
43455	Schroder	Alexandre	Sinovac	1/9/2021	India
29046	Van	Elizabeth	AstraZeneca	#####	Germany
80830	Glas	Gunnar	Johnson&	#####	Chile
73598	Winter	Issak	Novavax	#####	Hungary
14765	Arciniega	Daniel	Pfizer	#####	United states

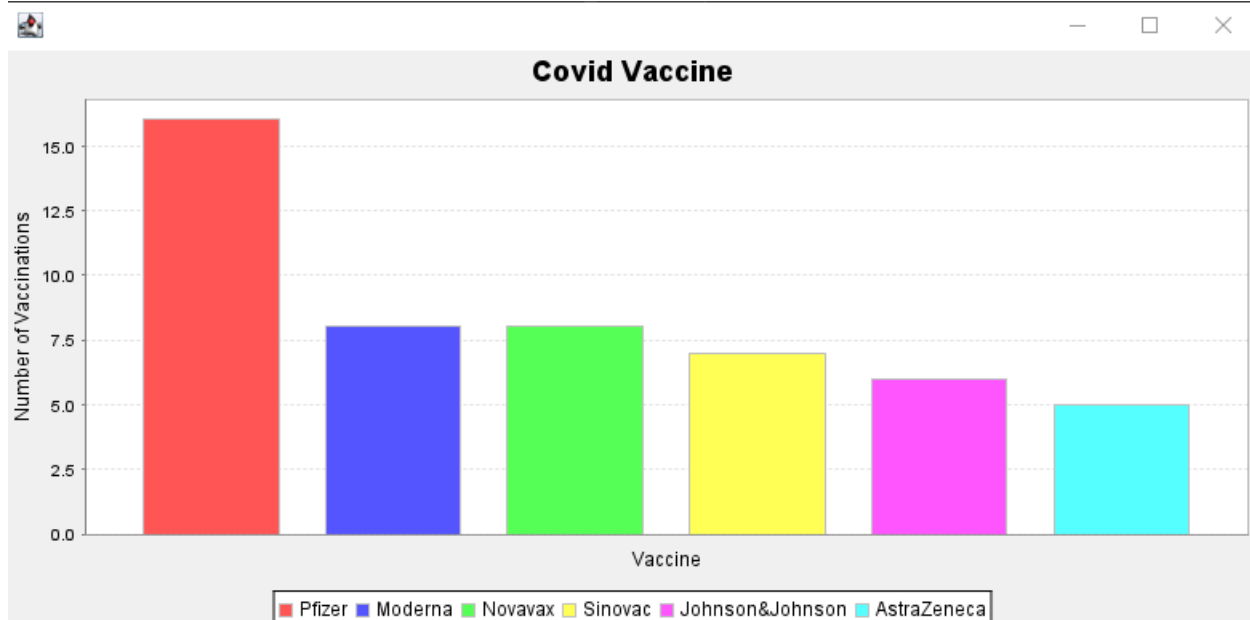
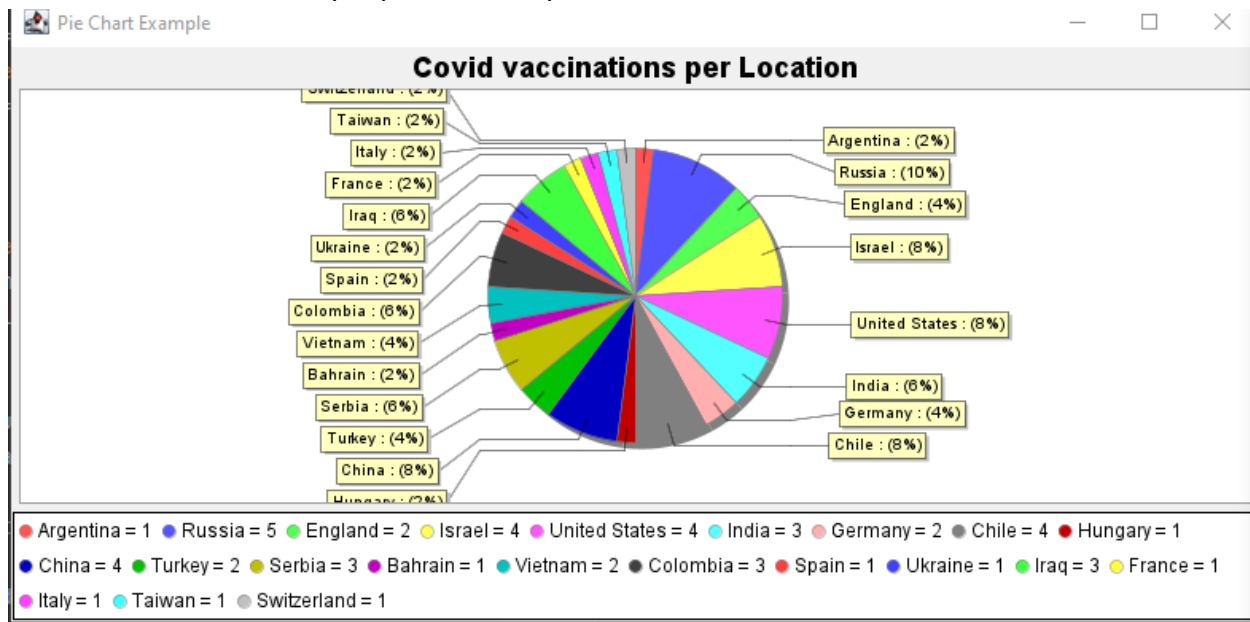
When we click Visualize bar:



When we click visualize pie:



When we use the csv 50 people file chart pics:



Testing

Our testing process:

Deleted source file, downloaded and opened the zip file to ensure working order.

Pressing the “About” button resulted in displaying the project group number and group member names.

Pressing the “Load data” button resulted in a prompt to enter in a file path to a .csv file.

Pressing the “Load CSV File” button in “Load data” resulted in opening a table with the information from the csv file displayed as a table.

Pressing the “Add data” button resulted in a prompt that asks the user to enter in covid vaccination data.

Pressing the “Add to table” button in “Add data” resulted in the entered data being added to the table.

Pressing the “Save data” button resulted in a prompt to save the data as a csv file.

Pressing the “Save file” button in “Save data” resulted in a file being created that saves the updated info as a csv file in a directory of your choosing.

Pressing the “Visualize data” button results in a prompt giving 2 buttons, 1 for bar chart, 1 for pie chart.

Pressing “Visualize bar” in “Visualize data” creates a bar chart displaying the data.

Pressing “Visualize pie” in “Visualize data” creates a pie chart displaying the data.