**CSP 586 project report**

**Requirement and Design Document**

**Team 20 :**

Bin Sun A20376746

Xiaofei Xie A20406972

Peng Guo

**1.Overview**

Our project is a dashboard mainly focus on processing the dataset relevant to environment . In our project, users are able to import the datasets of environment from the local file, then the data will be displayed in a table. If users are only interested in some categories rather all of the datasets, user can filter the data by selecting columns and filter the rows. Then group by the filtered datasets. After group by operation, users can chart the filtered data in bar chart, line chart, pie chart, doughnut chart, stacked bar chart. If the datasets include location geometry, users could use heat map to display the location on dashboard. All of the functions are very easily and kindly operated by users to analyze the environment datasets and all requirements would be ideally realized. Apart from that, our teammates carefully do manual test for the test case according to the project requirements and all of the bugs have been fixed and no problem exist now, the project could run ideally just as we imagine at the begin. It would be our glory if our project could be used to do environmental condition analysis for making the environment better.

**2.Requirement/Feature List**

1. The user can import the data file in CSV format from local files

2. The user can select data from a list of datasets in certain category

3. The selected dataset can be displayed as a table in the browser (Firefox or Chrome)

4. Datasets can be read from a data file in CSV or JSON format

5. The user can plot the data on the Dashboard in 5 chart types: Line, Bar, Pie, Stacked, Doughnut Chart.

6. The user can select/deselect columns for inclusion/exclusion of data from the dataset for Charting the selected data

7.The user must be able select/deselect columns for inclusion/exclusion of selected data from the

dataset for charting

8. The user can filter rows for inclusion/exclusion of the data from the dataset for charting

9. The user can chart basic stat data, like min, max, average, count, standard deviation.

10. The user can create and plot/chart complex analytical queries based on the column names and values for a given data set

11. The user can check the heat map according to the datasets.

**3. use cases and use case diagram**

**Actor:**

User

**Use case:**

1 Load Dataset in CSV or JSON Format

2 Select Dataset from Category

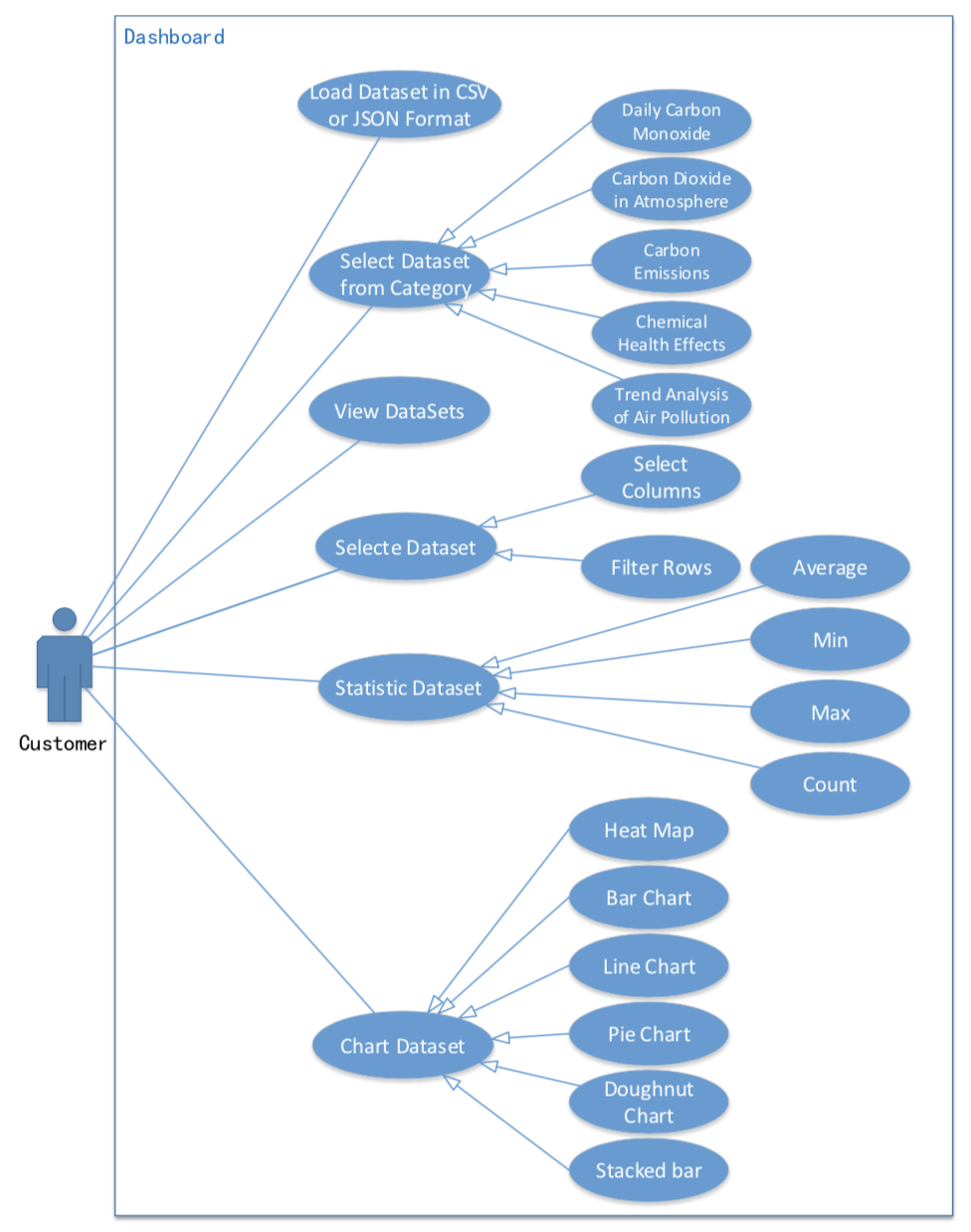
3 View Datasets

4 Select Datasets

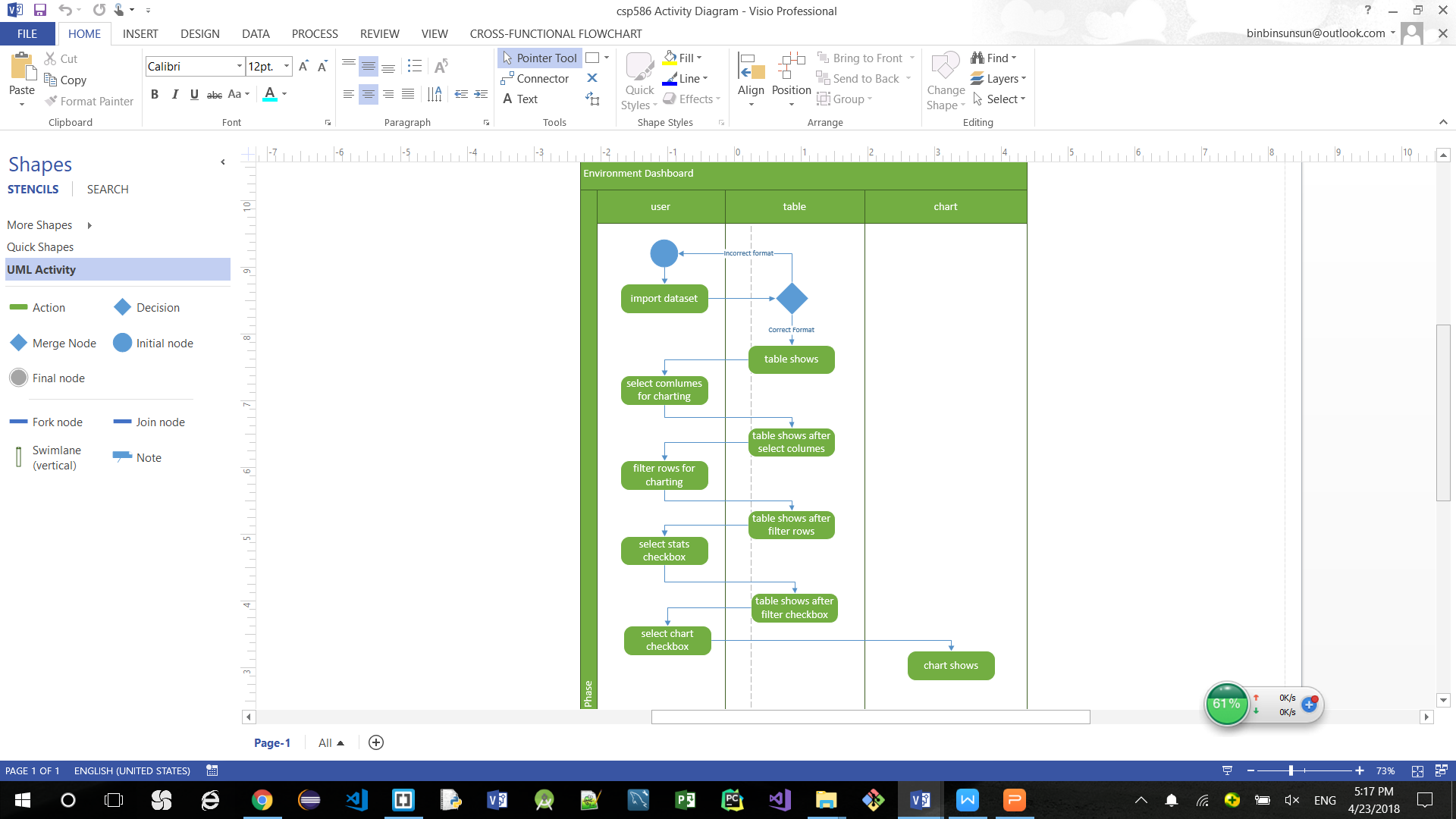
5 Statistic Filter Dataset

6 Chart Dataset

**Use case diagram :**

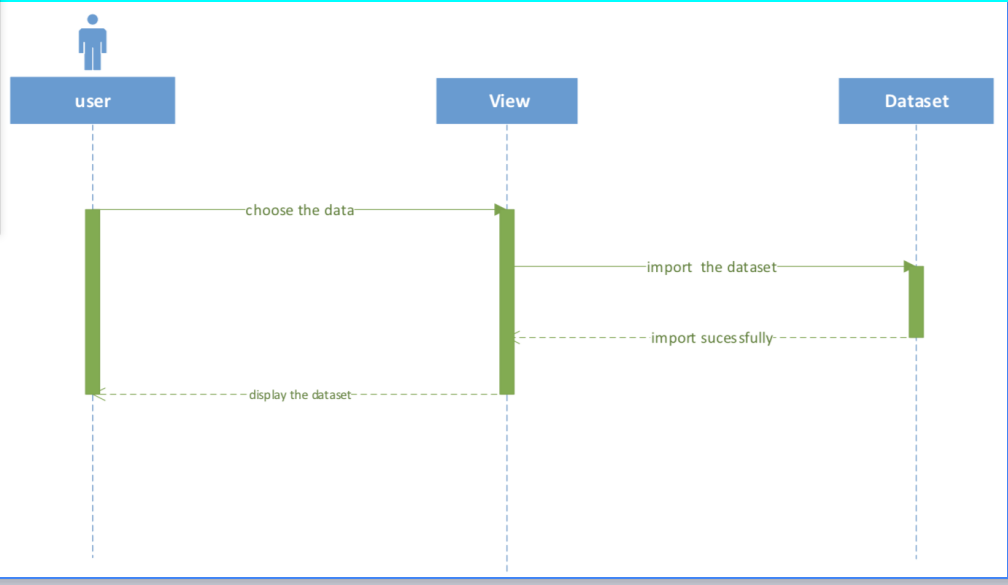


**4. Activity Diagram**

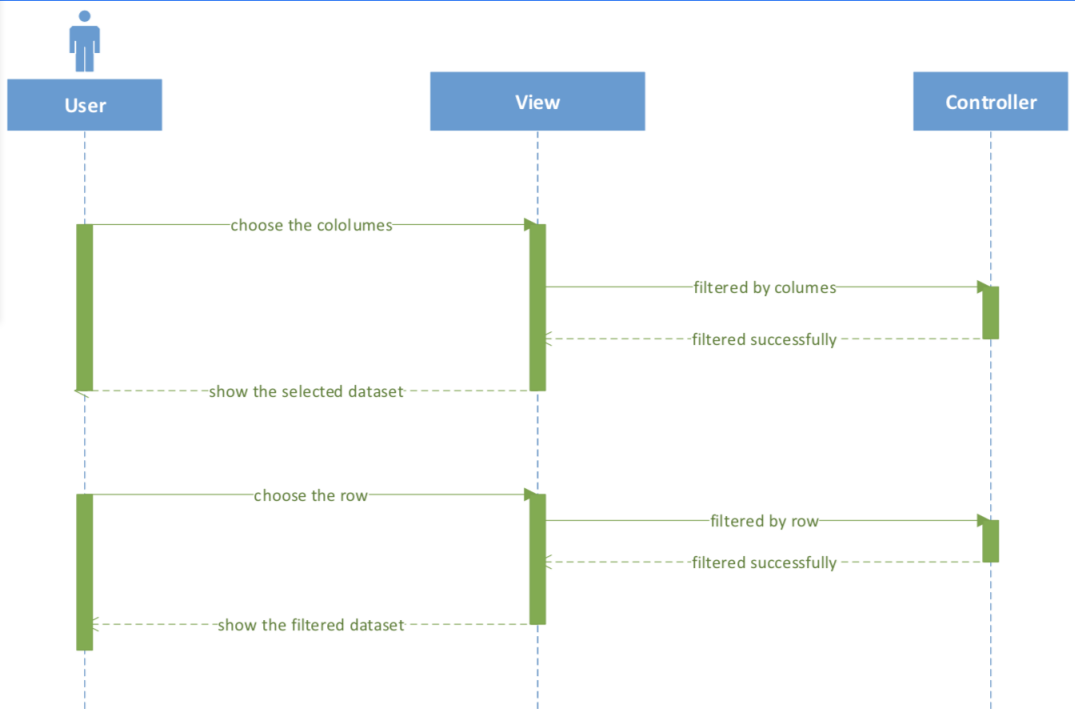


**5. Sequence Diagram**

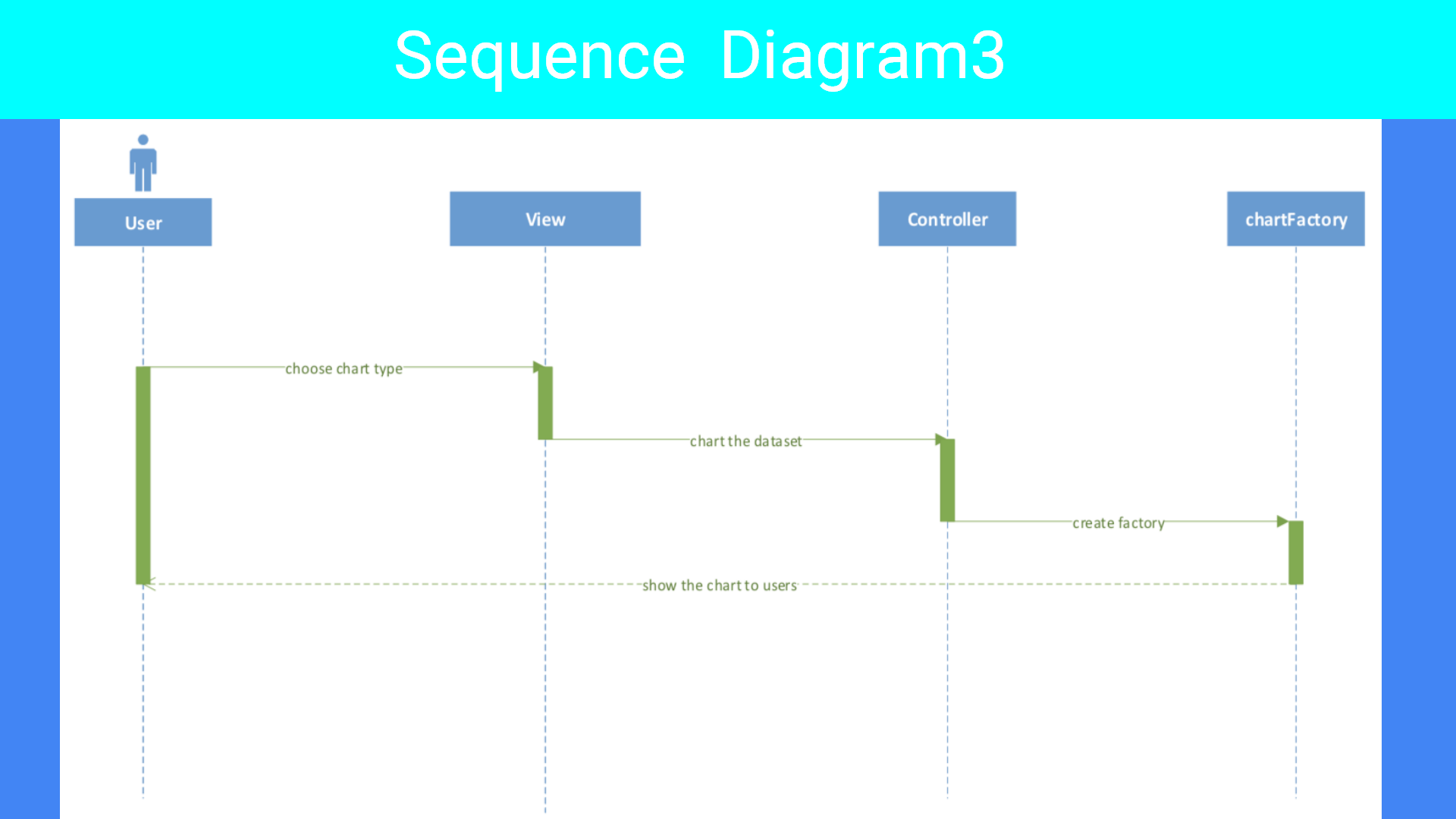
**1.User import the datasets and data displayed on Dashboard successfully**

****

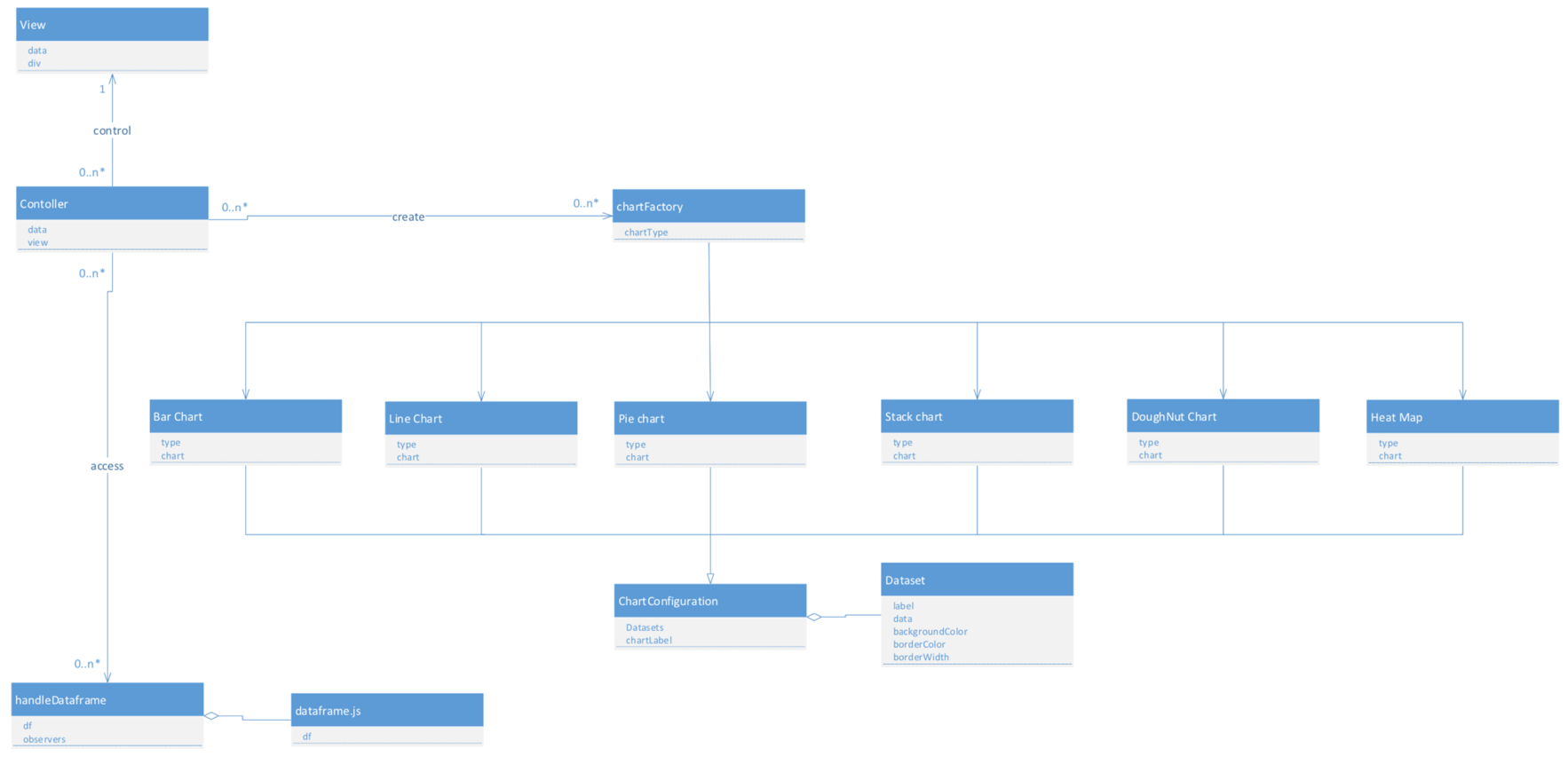
**2 user filter the dataset by columns and row, then filtered successfully.**

****

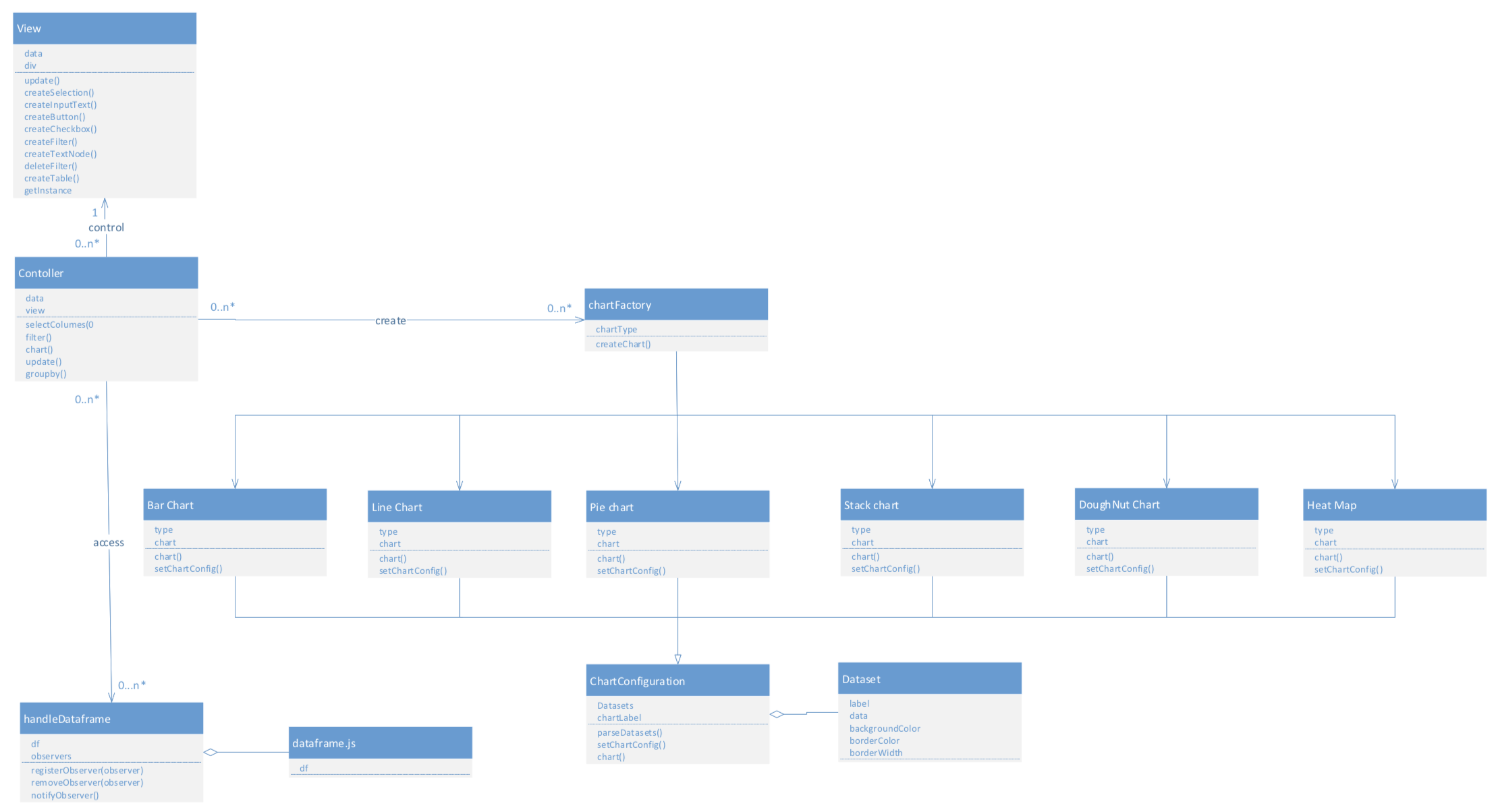
**3 user chart the dataset and chart shows successfully**



**6.Domain Model Class Diagram**

****

**7.Design Model Class Diagram**

****

**8. Documentation and class diagrams for Design Patterns used.**

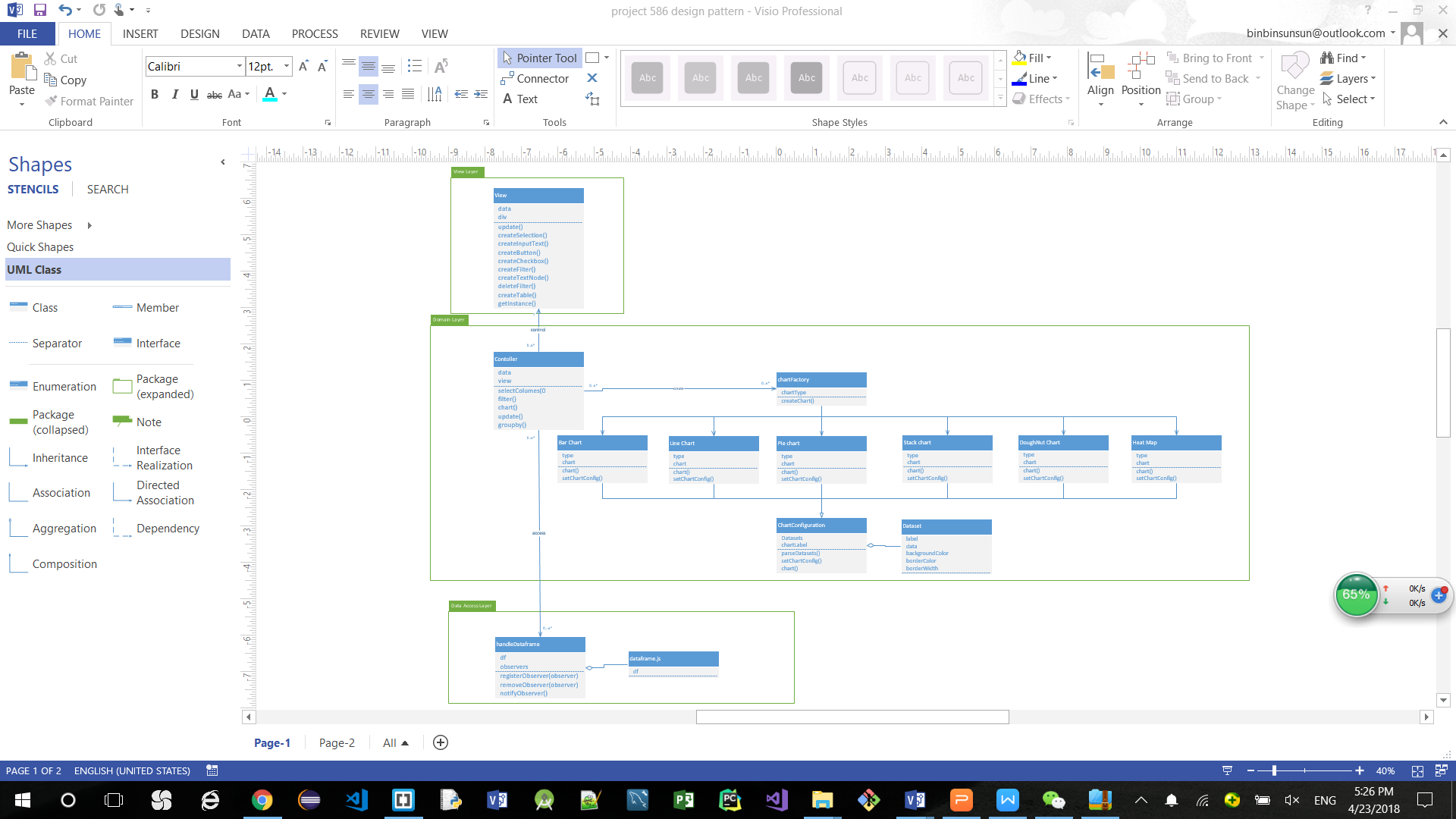
**1. MVC software Architecture**

**Model(**HandleDataFrame**)** - Model represents an object. It can also have logic to update controller if its data changes.

**View(**View**)** - View represents the visualization of the data that model contains.

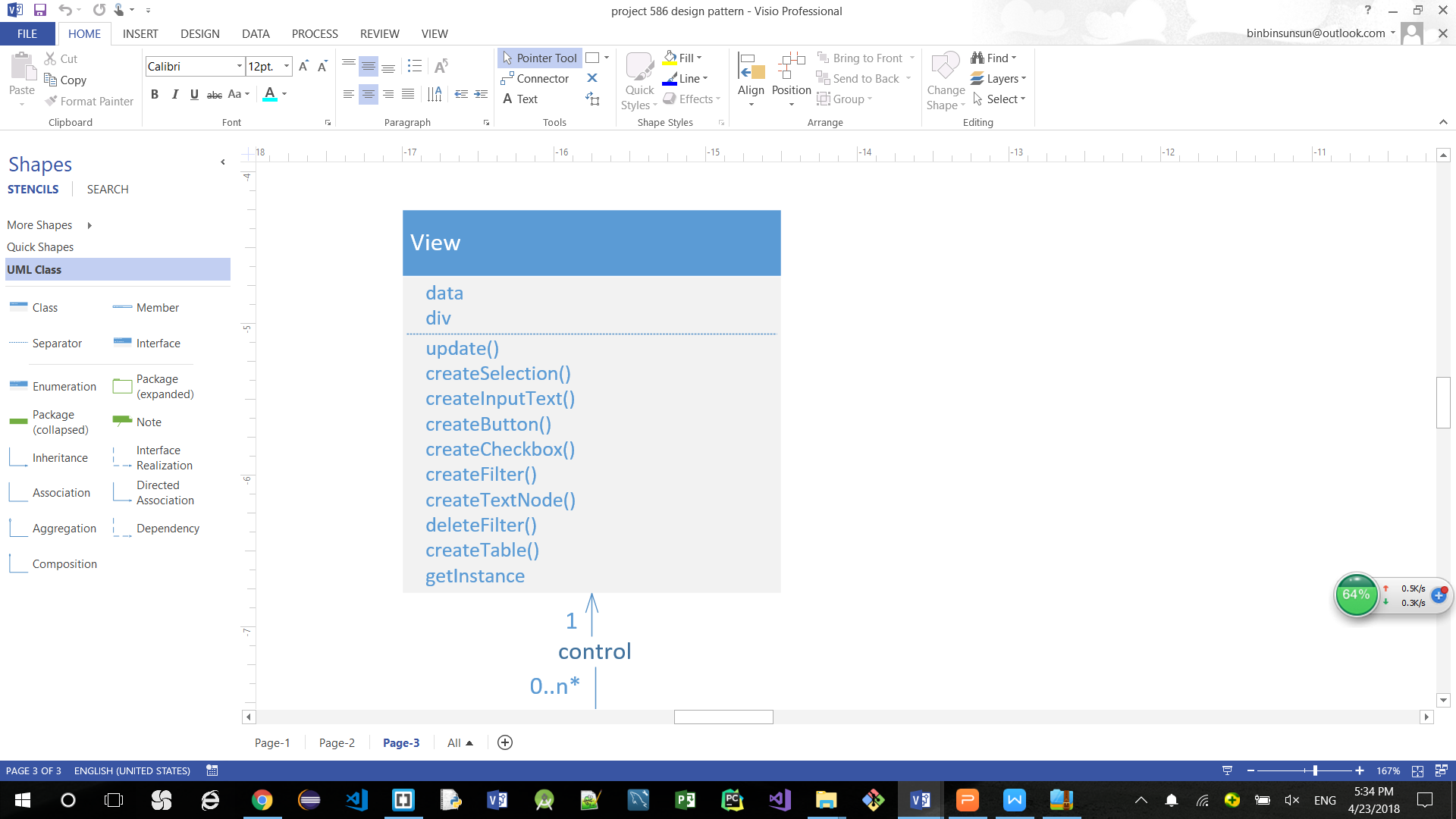
**Controller(**Contoller**)** –Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

We are going to create a HandleDataFrame object acting as a model. Class View will be a virtual view class which can show datasets on website and class Controller is responsible to manage data in HandleDataFrame object and update class view accordingly.



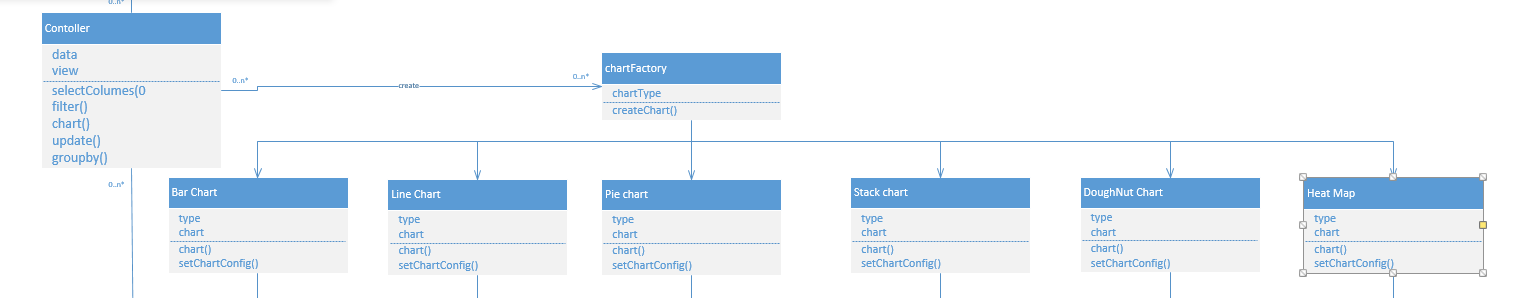
**2. singleton design pattern**

The private constructor view can only called by the method getInstance(), this method will judge whether the view exist, and the view could be created only once.



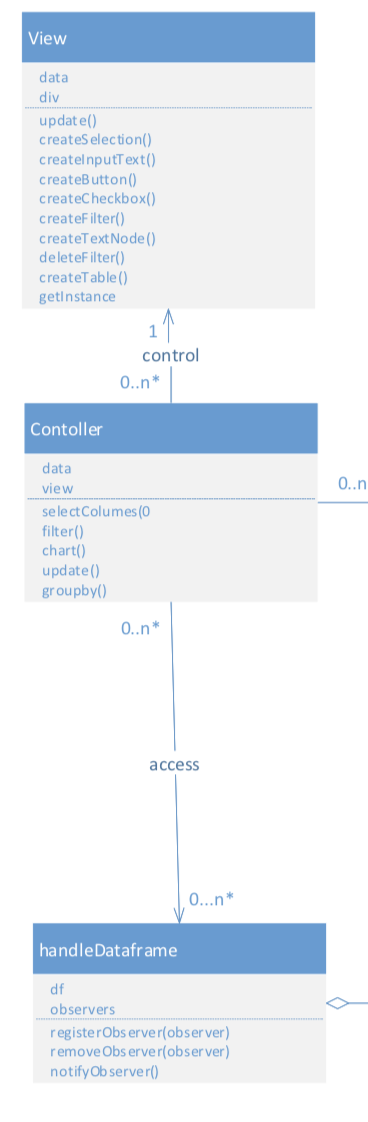
**3 Factory method design pattern**

ChartFactory defines the factory, bar chart, line chart, pie chart stack chart and DoughNut chart are all create by chartFactory class.



**4 observer design pattern**

HandleDataFame, as a subject, can register both view and controller objects as observers. Once user want to change the dataset, the subject will notify observers automatically of any state changes. After view and controller get the notification, they will call their update() methods. Finally, both datasets of View and Controller can be updated after observing.



**5 template design pattern**

The class ChartConfiguration defines the skeleton of operation, deferring some steps to subclasses. The content of setChartConfig(), parseData(), chart() will be defined differently to get the different chart in the subclasses.

