Software Design Document
New York Restaurant Inspection

Heang Sok_s5204340

Table of Contents

1.0 System Vision

1.1 Problem Background

Food safety has always been important and placed as a top priority in the hospitality industry. Therefore, it is crucial that the Department of Health Mental Hygiene (DOHMH) of New York do regular inspections on each restaurant in New York to find out any violation type committed by the restaurant and record it into a dataset. This dataset comprises very useful information about each violation, such as the restaurant's unique identifier, name, address, date of inspection, violation code, violation description, critical flag, score, grade, grade date, record date, and type of inspection.

However, as shown in the New York Restaurant Inspection dataset, the information is overwhelmed (from the year 2011 to 2017) and stored in a spreadsheet that has rows and columns which makes it hard and complicated for analysts to analyse and make assumptions. This conventional spreadsheet might be able to be analysed by application, such as Microsoft Excel, but the predefined functions that embedded in Microsoft Excel have limitation and do not fit for this dataset. This is a reason why a new analytical application should be built to give a better visual view of this dataset.

1.2 System Overview

The most important aspect of this application is to provide a decent standard of Graphic User Interface (GUI) in the form of buttons, input box, window view, and charts so that the analysts could generate data with just a few clicks of buttons. Additionally, it is also crucial that those generated data are meaningful and accurate. Thus, this new application should be capable of:

- retrieving all inspection details by selected date (filter by date)
- plotting the distribution of violations over the different suburbs by selected date
- retrieving all violations by selected keyword and date from user input
- be able to analyse all the cases which are related to animals, such as rats, mice, and others, that could lead to the contamination of the food
- produce a chart to show numbers of violations based on the violation code and selected date

1.3 Potential Benefits

This application brings a lot of benefits to the Department of Health Mental Hygiene (DOHMH) of New York because it enables fast and accurate decision making for decision makers to decide on which restaurant should be closed due to the poor practice and level of the violation. In addition, by using this application, the user can generate statistics of the violation type that happened most in certain neighbourhoods. This information is very useful for DOHMH because they can utilise the information to find the root cause of the violation and then come up with solutions to act and prevent it from happening again. In conclusion, the deployment of this application is expected to help the DOHMH of New York to ensure that the food which is served in the restaurant in New York is met the required standard and free of any contamination.

2.0 Requirements

2.1 User Requirements

The primary user for this analytical software is the staffs at Department of Health Mental Hygiene (DOHMH) of New York. In order to interact with this software, the users need to set up a new profile first to gain the right to use this software. At this point, the users provide their personal information, such as username, password, and contact details. Then the users can log in to this software with their credentials through a keyboard.

After logging in to this software, the users will see a home window that allows them to select different options (also see the visual design in section 4.2 of this paper), such as:

- View by Date
- View by Date + Suburb
- View by Date + Keyword
- View Animal Related Case
- View Chart by Violation Code

If the users request to view all the inspection details by date, they shall select the "View by Date" option, then the software will pop up another window for the users to select the start date and end date so that they can view specific periods that they want to analyse. Moreover, if the users require to view all the violations in a specific suburb and date, they shall select the "View by Date + Suburb" option. Once again, the software will pop up another window for the users to select the start date and end date. In addition to this window, there is a dropdown button for the users to select the required suburb.

The "View by Date + Keyword" and "View Animal Related Case" options are similar because if the users select either one of these options, the software will pop up a new window that contains a start date button, an end date button, and a search box. However, the generated reports are not the same because the "View Animal Related Case" will generate a type of information that focuses on the violations that are related to animals while "View by Date + Keyword" will generate general information.

Lastly, if the users want to view the bar charts by violation code, they shall select the "View Chart by Violation Code" option. Then the software will go to the next window and allow the users to input the start date, end date, and violation code.

2.2 Software Requirements

In this section, this team describes both functional and non-functional requirements for this new system.

The functional requirements are listed as:

R1 The software shall retrieve all inspection details from the datasets based on a selected date.

R2 The software shall plot the distribution of violations over the different suburbs by a selected date.

R3 The software shall view all violations by keyword input and a selected date.

R4 The software shall retrieve all animal related violations and show it on the screen based on keyword input and selected date.

R5 The software shall be able to generate charts or graphs according to the violation code and selected date.

R6 The software shall allow the users to set up a new profile.

R7 The software shall check the user credential when users log in.

R8 The software shall display an error message when the users provide an invalid username or password.

R9 The software shall allow the users to edit their profile and change their password.

R10 The software shall be able to execute all datasets that have csv as the extension

R11 The software shall only accept English keyword when searching through the dataset

In terms of non-functional requirements, this team utilised the FURPS+ framework, which includes usability, reliability, performance, security, and design constrain. All non-functional requirements are listed in the table below.

Non-Functional Requirements	
FURPS+ Category	Requirement Description
Usability	• This analytical software can be run smoothly on Window and Mac Operating System • This analytical software can be executed on a local machine without the need of the internet • This analytical software is designed with the appropriate size of fonts and buttons to reduce stresses on users' eyes
Reliability	• This analytical software can run in the background of the operating system which means that the users can run other software at the same time • This system does not crush with other software when running
Performance	• This software supports one user at a time • This software can run immediately without delay time (loading time) • This software consumes small memory resources because the system size is less than 100MB

Security	• Since this software can run on the local machine without the internet, it is free from all kinds of internet attacks

Design Constraints	• Standard computer machine requirements for this software:
	  ○ Processor: 1 gigahertz (GHz) or faster processor
	  ○ RAM: 4GB
	  ○ Storage: 16GB
	• This system requires the following working environment to run:
	  ○ Python version 3.9
	  ○ Python libraries: Pandas, Matplotlib, Tkinter, csv
	  ○ Operating system: Window 10, Mac (Big Sur)

2.3 Use Cases

In order to gain a better understanding of this project, this team decided to build the use case diagram to illustrate how the system will be built and worked at each step as shown in figure 1.

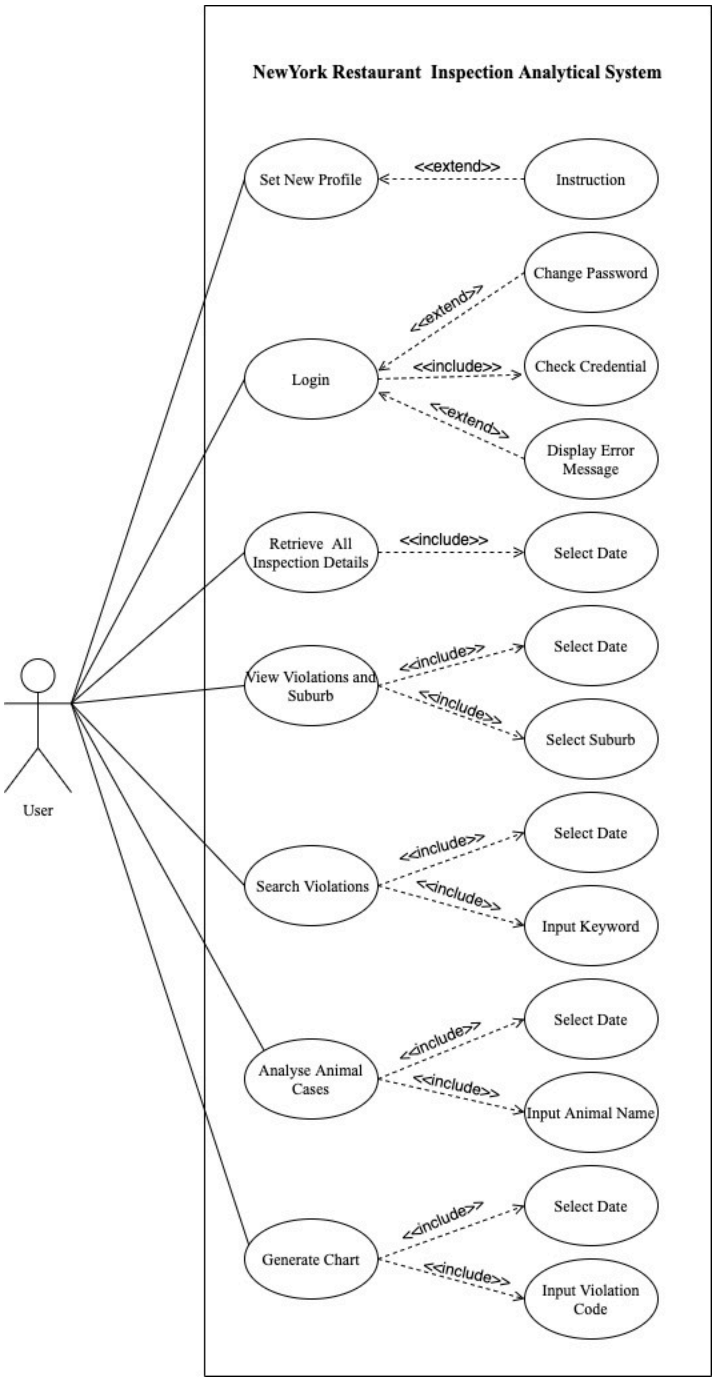

Figure 1: A Full Use Case Diagram

The “Fully Developed Use Case Description” tables below will provide detail on how each use case is executed.

Use case1:

Fully Developed Use Case Description		
Use Case Name:	Set New Profile	
Scenario:	Staffs need to set a new profile before they can use the analytical system	
Triggering Event:	Staffs want to use the analytical system	
Brief Description:	When setting a new profile, the staffs need to provide some personal information, such as username, password, and contact detail.	
Users/Actor:	Staffs at the Department of Health Mental Hygiene (DOHMH)	
Related Use Cases:	Login Use Case	
Preconditions:	• The staffs need to execute the program first • The staffs select the set up new profile button	
Postconditions:	• The staffs can log in to this analytical system	
Flow of Activities	Staffs	System
	1. A staff select the set up new profile button	1.1. The system shows a new widow and prompts for the staff’s personal information
	2. The staff type in his personal information, such as username, user password, and contact detail; then click save	2.1. The system saves and archives the staff’s personal information in a json file
Exception Conditions	2.1. The system cannot save the staff’s personal information if there is any blank entry. The staff needs to fill all the entries first before he/she can continue to the next step	

Use case2:

Fully Developed Use Case Description		
Use Case Name:	Login	
Scenario:	Staffs need to log in before they can use the analytical system	
Triggering Event:	Staffs want to use the analytical system	
Brief Description:	When the staffs log in to the analytical system, they need to provide a valid username and password	
Users/Actor:	Staffs at the Department of Health Mental Hygiene (DOHMH)	
Related Use Cases:	Set New Profile Use Case	
Preconditions:	• The staffs need to have their profile set in the system first	
Postconditions:	• The system will show a new screen for the staffs to retrieve all inspection details, view violations, search violations, analyze animal related cases, and generate a chart	
Flow of Activities	Staffs	System
	1. A staff log in to the analytical system	1.1. The system prompts for the staff’s username and password
	2. The staff type in username and password	2.1. The system checks the credential and continues to the next window screen
Exception Conditions	2.1. The system cannot move to the next window screen if the credential is invalid	

Use case3:

Fully Developed Use Case Description		
Use Case Name:	Retrieve All The Inspection Details	
Scenario:	A staff wants to retrieve all the inspection details from 2015 to 2017	
Triggering Event:	A staff intends to analyze all the inspection details from 2015 to 2017	
Brief Description:	For this use case, the staff can view all the inspection details by selected date (filtered by date)	
Users/Actor:	Staffs at the Department of Health Mental Hygiene (DOHMH)	
Related Use Cases:	Login Use Case	
Preconditions:	• The staff needs to log in first • The staff needs to select a specific date first	
Postconditions:	• The system will show all the inspection details filtered by selected date	
Flow of Activities	Staffs	System
	1. The staff selects the "View by Date" button on the home window	1.1 the system shows the "View by Date" window and prompts the user to select the start date and end date
	2. The staff selects the start date (01/01/2015) and end date (31/12/2017); then click the view report button	2.1. the system views a report on the window
Exception Conditions	2.1. The system will not show the inspection details that are archived in the dataset if the staff does not select a specific date	

Use case4:

Fully Developed Use Case Description		
Use Case Name:	View Violations And Suburbs	
Scenario:	A staff wants to view the violation information of restaurants in Queens suburb, New York in July 2016	
Triggering Event:	A staff intends to analyze the violation information of restaurants in Queens suburb, New York in July 2016	
Brief Description:	The staff can view the violation information of restaurants by selected suburb and date	
Users/Actor:	Staffs at the Department of Health Mental Hygiene (DOHMH)	
Related Use Cases:	Login Use Case	
Preconditions:	• The staff needs to log in first • The staff needs to select a specific date • The staff needs to select a specific suburb	
Postconditions:	• The system will show all the inspection details filtered by selected date	
Flow of Activities	Staffs	System
	1. The staff selects the "View by Date + Suburb" button on the home window	1.1 The system shows the "View by Date + Suburbs" window and prompts the user to select the start date, end date, and suburb
	2. The staff selects the start date (01/07/2016), end date (31/07/2016), and suburb (Queens); then click the view report button	2.1. The system shows the violation information of restaurants in Queens, New York in July 2016
Exception Conditions	2.1. The system will not show the violation information of the restaurants in Queens, New York, if the staff does not select any date or suburb	

Use case5:

Fully Developed Use Case Description		
Use Case Name:	Search Violation	
Scenario:	A staff wants to search for the violation information of all Chinese restaurants in New York in April 2015	
Triggering Event:	A staff intends to analyze all the violations that committed in all Chinese restaurants in New York in April 2015	
Brief Description:	The staff can search and analyze all the violation information of restaurants by keyword input and a selected date	
Users/Actor:	Staffs at the Department of Health Mental Hygiene (DOHMH)	
Related Use Cases:	Login Use Case	
Preconditions:	• The staff needs to log in first • The staff needs to select a specific date • The staff needs to type in a keyword	
Postconditions:	• The system will show all the search results	
Flow of Activities	Staffs	System
	1. The staff selects the "View by Date + Keyword" button on the home window	1.1 The system shows the "View by Date + Keyword" window and prompts the user to select the start date, end date, and input a keyword
	2. The staff types the word "Chinese restaurants" in the search box and selects the start date (01/04/2015) and end date (31/04/2015); then he clicks the view report button	2.1. The system shows all the search results on the screen
Exception Conditions	2.1. The system will show nothing if the staff does not type in the search box. 2.1. If the staff types the word "Chinese restaurants" in the search box, but he does not select a specific date, the system will show the entire period of all the violations that were committed in the Chinese Restaurant in New York.	

Use case6:

Fully Developed Use Case Description		
Use Case Name:	Analyze Animal Cases	
Scenario:	A staff wants to view animal-related violation information in New York restaurants in 2016.	
Triggering Event:	Animals, such as rats, mice, and others, could lead to food contamination; therefore, it is essential to analyze these types of violations	
Brief Description:	The staff is able to analyze all the cases which are related to animals that could lead to the contamination of the food	
Users/Actor:	Staffs at the Department of Health Mental Hygiene (DOHMH)	
Related Use Cases:	Login Use Case	
Preconditions:	• The staff needs to log in first • The staff needs to select a specific date • The staff needs to type in an animal name	
Postconditions:	• The system will show all the analyzing results	
Flow of Activities	Staffs	System
	1. The staff selects the "View Animal Related Case" button on the home window	1.1. The system shows the "View Animal Related Case" window and prompts the user to select the start date, end date, and input an animal name

	2. The staff types the word “mice” in the analyzing box and selects the start date (01/01/2016) and end date (31/12/2016); then he clicks the view report button	2.1. The system shows all the analyzing results on the screen
Exception Conditions	2.1. The system will show nothing if the staff does not type in the analyzing box 2.1. If the staff type the word “mice” in the analyzing box, but does not select a specific date, the system will show the entire periods of all the animal-related violation information in New York	

Use case7:

Fully Developed Use Case Description		
Use Case Name:	Generate Chart	
Scenario:	A staff wants to generate a chart that shows numbers of violations based on the violation code “10F” in 2014	
Triggering Event:	The staff wants to know how many cases that related to violation code “10F” in 2014	
Brief Description:	For this use case, the staff can produce a chart that tells how many times does the violation code “10F” occurs in 2014	
Users/Actor:	Staffs at the Department of Health Mental Hygiene (DOHMH)	
Related Use Cases:	Login Use Case	
Preconditions:	• The staff needs to log in • The staff needs to select a specific date • The staff needs to type in a violation code	
Postconditions:	• The system will show all the search results	
Flow of Activities	Staffs	System
	1. The staff selects the “View Chart by Violation Code” button on the home window	1.1. The system shows the “View Chart by Violation Code” window and prompts the user to select the start date, end date, and input a violation code
	2. The staff types the code “10F” in the search box and selects the start date (01/01/2014) and end date (01/01/2014); then he clicks the view report button	2.1. The system shows all the chart results on the screen
Exception Conditions	2.1. The system will show nothing if the staff does not type in the input box 2.1. If the staff type the code “10F” in the analyzing box, but he does not select a specific date, the system will show the entire periods of all the violation regarding code “10F”	

3.0 Software Design and System Components

3.1 Software Design

In order to illustrate how this new software will work, this team has created an activity diagram to show the flow of each process when running the software.

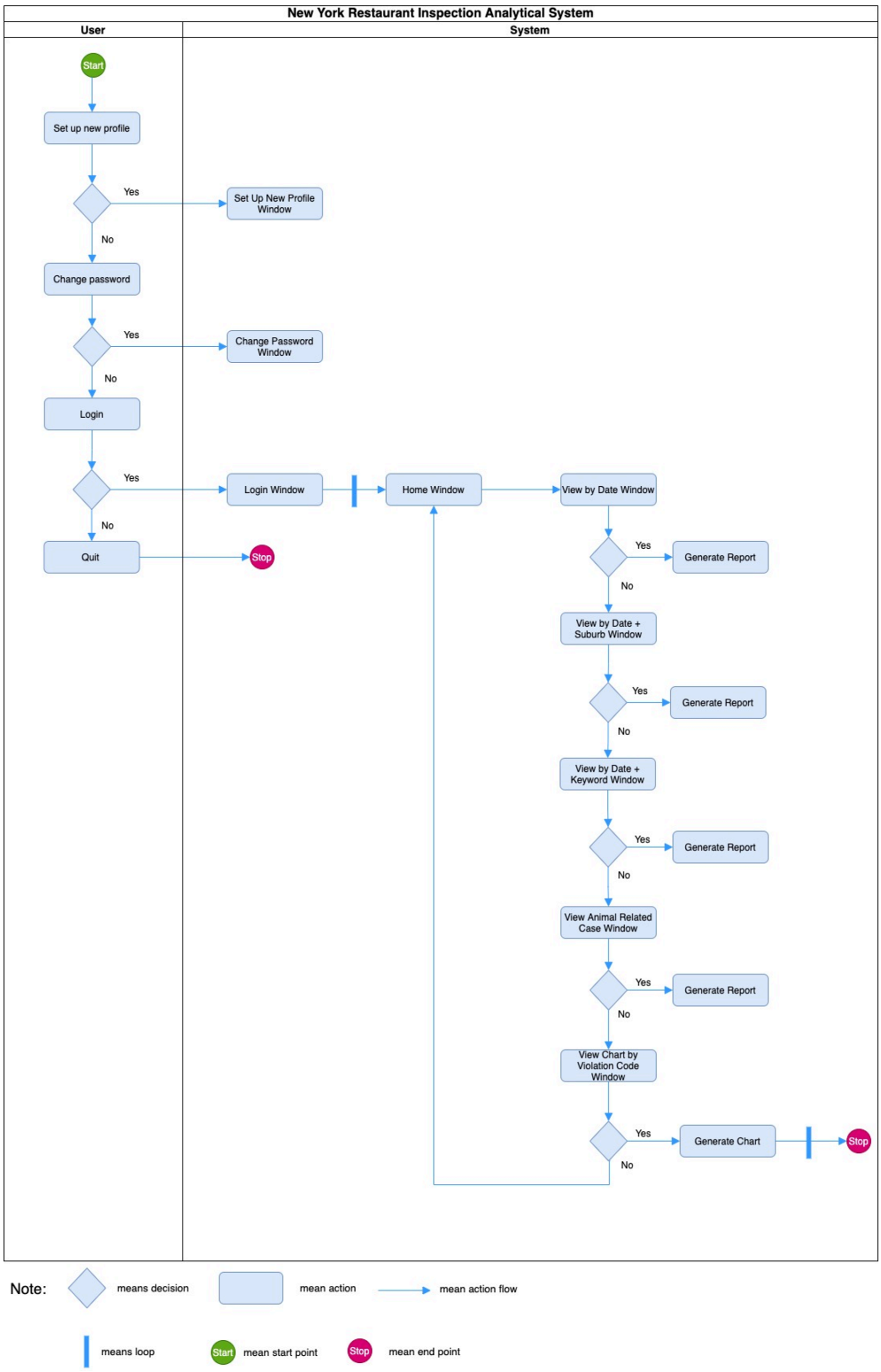

Figure 2: Activity Diagram

The class diagram below shows the main block of each class that is required in this new analytic software. This class diagram is a general concept that is used as a model to code the software.

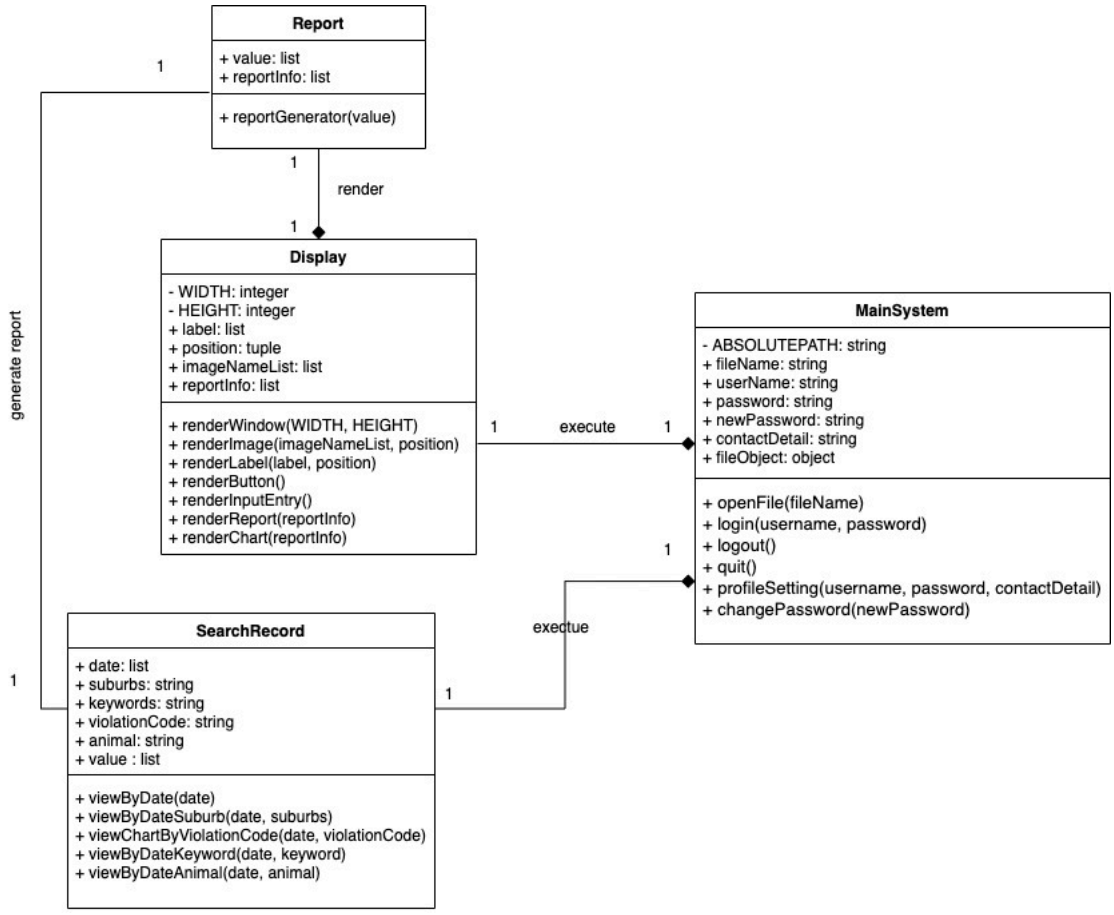

Figure 3: Class Diagram

3.2 System Components

3.2.1 Functions

Function: openFile(fileName)

Description: this function is used to open files, such as csv file and json file.

Parameters: this function takes one parameter called “fileName” which has a string data type. We use this parameter to call the file name that we want to open.

Side Effects: none

Return Value: this function returns a file object that can be used by other functions

Function: login(username, password)
Description: this login function requires the user to fill in two input entry boxes (username and password boxes), and the software will check the provided username and password whether it is valid or not. If it is valid, the software will move to the next window. If it is invalid, a warning message will be popped up.
Parameters: this function accepts two parameters, such as username and password; the data type is a string. These two parameters will be used to check the credential.
Side Effects: none
Return Value: the return value of this function is the Boolean value: True or False.

Function: logout()
Description: this function allows the user to logout the software with a single click on the logout button
Parameters: this function do not take any parameter
Side Effects: none
Return Value: there is no return value

Function: quit()
Description: this function allows the user to exit the software with a single click on the cross button
Parameters: this function do not take any parameter
Side Effects: none
Return Value: there is no return value

Function: profileSetting(username, password, contactDetail)
Description: this function allows the user to set up a new profile
Parameters: this function takes 3 parameters that have a string data type, such as username, password, and contacDetail. These parameters tell the user's personal information and will be saved in a json file.
Side Effects: none
Return Value: there is no return value because this function will automatically save these information to a json file

Function: changePassword(newPassword)
Description: this function provides related services for users who need to change their password. The software will require the user to log in first. Then the user need to fill the new password in the input entry box of the new password.
Parameters: the parameter of this function is called newPassword, and its data type is a string. This parameter is used to override the old password.
Side Effects: it will override the old password in a json file, the global variable called "password" is also overridden
Return Value: there is no return value in this function

Function: viewByDate(date)
Description: this function allows the user to retrieve all inspection details by selected date
Parameters: this function take a parameter called date; this parameter has a list data type which tell the start date and end date that the user wants to search for.
Side Effects: this function will change the value of a global variable called "value"
Return Value: this function returns a value that has a data type as a list

Function: viewByDateSuburb(date, suburbs)
Description: this function allows the user to view the distribution of violations over the different suburbs by selected date.
Parameters: this function take two parameters, such as date (data type= list) and suburbs (data type = string); these parameters are used to search for a suburb and specific range of date.
Side Effects: this function will change the value of a global variable called "value"
Return Value: this function returns a value that has a data type as a list

Function: viewByDateKeyword(date, keyword)
Description: this function allows the user to retrieve all violations by selected keyword and date from user input
Parameters: this function take two parameters, such as date (data type= list) and keyword (data type = string); these parameters are used to search for a keyword and specific range of date.
Side Effects: this function will change the value of a global variable called "value"
Return Value: this function returns a value that has a data type as a list

Function: viewByDateAnimal(date, animal)
Description: this function allows the user to analyse all the cases which are related to animals, such as rats, mice, and others, that could lead to the contamination of the food.
Parameters: this function take two parameters, such as date (data type= list) and animal (data type = string); these parameters are used to search for an animal and specific range of date.
Side Effects: this function will change the value of a global variable called "value"
Return Value: this function returns a value that has a data type as a list

Function: viewChartByViolationCode(date, violationCode)
Description: this function allows the user to produce a chart to show numbers of violations based on the violation code and selected date.
Parameters: this function take two parameters, such as date (data type= list) and violation code (data type = string); these parameters are used to search for a violation code and specific range of date.
Side Effects: this function will change the value of a global variable called "value"
Return Value: this function returns a value that has a data type as a list

Function: reportGenerator(value)
Description: this function will select only useful information from the Parameter and then produce a new list of those information
Parameters: this function take one parameter called value (data type= list); this parameter is used to generate a report.
Side Effects: this function will change the value of a global variable called "reportInfo"
Return Value: this function returns a value that has a data type as a list

Function: renderWindow(WIDTH, HEIGHT)
Description: this function is responsible for displaying the window
Parameters: this function has two parameters which are WIDTH (data type= integer) and HEIGHT (data type= integer); these parameters are used to define the width and height of the window.
Side Effects: none
Return Value: there is no return value in this function

Function: renderImage(imageNameList, position)
Description: this function will render images/icons on the window screen
Parameters: this function take two parameters, such as imageNameList (data type= list) and position (data type= tuple); the "imageNameList" parameter includes all the required image names in a list while the "position" parameter tells where the image should be placed on the screen.
Side Effects: none
Return Value: there is no return value in this function

Function: renderLabel(label, position)
Description: this function will render all the labels on the window screen
Parameters: this function take two parameters, such as label (data type= list) and position (data type= tuple); the "label" parameter includes all the labels' information in a list while the "position" parameter tells where the label should be placed on the screen.
Side Effects: none
Return Value: there is no return value in this function

Function: renderButton()
Description: this function will render all the buttons on the screen
Parameters: this function take no parameter
Side Effects: none
Return Value: there is no return value in this function

Function: renderInputEntry()
Description: this function will render all the input entries on the window
Parameters: this function take no parameter

Side Effects: none
Return Value: there is no return value in this function

Function: renderInputEntry()
Description: this function will render all the input entries on the window
Parameters: this function take no parameter
Side Effects: none
Return Value: there is no return value in this function

Function: renderReport(reportInfo)
Description: this function will print the report on the window
Parameters: this function take one parameter which is called reportInfo (data type= list); this parameter contains required information that should be printed on the screen
Side Effects: none
Return Value: there is no return value in this function

Function: renderChart(reportInfo)
Description: this function converts the values of its parameter into a chart; then print it on the window screen.
Parameters: this function take one parameter which is called reportInfo (data type= list); this parameter contains required information that can be converted into a chart.
Side Effects: none
Return Value: there is no return value in this function

3.2.2 Data Structures

Since this team use datatypes, such as list and tuple, to store information, the data structure that is used in this project is an array. Note that in python programming language the datatype called list and tuple is an array in other programming language.

Variable: imageNameList
Data Structure: Array
Description: this variable stores an array of images' name
Data Members: string (note: the string is stored in this array)
Use by Function: renderImage(imageNameList, position)

Variable: position
Data Structure: Array
Description: this variable stores an array of positions
Data Members: integer (note: the integer is store in this array)
Use by Function: renderImage(imageNameList, position), renderLabel(labe, position)

Variable: label
Data Structure: Array
Description: this variable stores an array of labels
Data Members: string (note: the string is stored in this array)
Use by Function: renderLabel(label, position)

Variable: reportInfo
Data Structure: Array
Description: this variable stores an array of report's information
Data Members: string (note: the string is stored in this array)
Use by Function: renderReport(reportInfo), renderChart(reportInfo)

Variable: date
Data Structure: Array
Description: this variable stores an array of start date and end date
Data Members: string (note: the string is stored in this array)
Use by Function: viewByDate(date), viewByDateSuburb(date, suburbs), viewChartByViolationCode(date, violationCode), viewByDateKeyword(date, keyword), viewByDateAnimal(date, animal)

Variable: value
Data Structure: Array
Description: this variable stores an array of value that return from: viewByDate(date), viewByDateSuburb(date, suburbs), viewChartByViolationCode(date, violationCode), viewByDateKeyword(date, keyword), viewByDateAnimal(date, animal)
Data Members: string (note: the string is stored in this array)
Use by Function: reportGenerator(value)

3.2.3 Detailed Design

The main purpose of this section is to write a pseudo code that tells the algorithm behind this new software. According to Wikipedia, pseudo code can be written in JAVA like style or Cobol like style. Since this course focuses on Python, this team decided to use Python style to write this pseudo code to show the algorithm behind the software.

```
# Todo: -----------------define classes------------------
class Display:
    """This class will render images/icons, labels, buttons
       input entry, report, chart"""
        def __init__(self):
            # define all the attribute that are needed in this class

        def renderWindow(self, WIDTH, HEIGHT):
            #use tkinter in-built function to draw a window (set WIDTH and
             HEIGHT)
            #set the in-built function called geometry to False so that
             the user cannot resize the window

        def renderImage(self, ImageNameList, position):
            # use tkinter in-built function called canvas.create_image to
            render image on the screen

        def renderLabel(self, positon):
            # use Label gadget in tkinter to render the labels

        def renderInputEntry(self):
            # use Entry gadget in tkinter to render input entry boxes

        def renderReport(self, reportInfo):
            if reportinfo is not equal to Null and not returned from
            viewChartByViolationCode(date, violationCode):
                 # render the report on the report on the window
            else:
                 # print a message to tell the user to input some values in
                  the input entry and try again

        def renderChart(self, reportInfo):
            if reportinfo is not equal to Null and the reportInfo is
            returned from viewChartByViolationCode(date, violationCode):
                 # render a chart and print it on the window
            else:
                 # print a message to tell the user to input some values in
                 the input entry and try again

class SearchRecord:
    """This class allows user to search for specific information based on
       user input"""
        def __init__(self):
            # define all the attribute that are needed in this class

        def viewByDate(self, date):
            if date is not equal to Null:
                 if date is in between 01-01-2011 and 31-12-2017 :
                     # use regular expression search for all inspection
                      details by selected date
                     # return an array of value to use in other functions
                 else:
```

```
                    # print a message to tell the user to try again



        def viewByDateSubub(self, date, suburbs):
            if date and suburbs are not Null:
                if date is in between 01-01-2011 and 31-12-2017 :
                    # use regular expression to search the distribution of
                    violations over the different suburbs by selected date
                    # return an array of value to use in other functions
                else:
                    # print a message to tell the user to try again

        def viewChartByViolationCode(self, date, violationCode):
            if date and violationCode are not Null:
                if date is in between 01-01-2011 and 31-12-2017 :
                    # use pandas to retrieve all inspection details by
                      selected violationCode
                    # then use matplotlib to generate the chart
                    # return an array of value to use in other function
                else:
                    # print a message to tell the user to try again

        def viewByDateKeyword(self, date, keyword):
            if date and keyword are not Null:
                if date is in between 01-01-2011 and 31-12-2017 :
                    # use regular expression to retrieve all violations by
                    selected keyword and date from user input
                    # return an array of value to use in other function
                else:
                    # print a message to tell the user to try again

        def viewByDateAnimal(self, date, animal):
            if date and animal are not Null:
                if date is in between 01-01-2011 and 31-12-2017 :
                    # use regular exprssion to retrieve all violations
                    that are related to animal
                    # regular expression might be needed in this function
                    # return an array of value to use in other function
                else:
                    # print a message to tell the user to try again

class Report:
    """This class is used to filter only require information to print on
       the screen"""
       def __init__(self):
            # define all the attribute that are needed in this class

       def reportGenerator(self, value):
            # Automatically filter the "value" parameter and select only
            useful information
            # to filter the information, regular expression and pandas will
            be used here
            # return a new array of information

class MainSystem:
    """This class responsible for general functions, such as open file,
    login, logout, quit, and much more"""
       def __init__(self):
            # define all the attribute that are needed in this class
```

```
def openFile(filename):
     if the user set up a new profile:
          # use open() to open a json file
          # use try and except function to handling errors if the
          file is not found
     elif the user want to analyse the dataset:
          # use pandas.read_csv to open a csv file
          # use try and except function to handling errors if the
          file is not found
          return a file object to use with other functions

     def login(self, username, password):
         # use regular expression to check if the username and
         password are in the json file or not
         if the username and password in the json file:
              # return True
         else:
              # print a warning message that the credential is
              invalid and the user might want to try again
              # return False

     def logout(self):
          # if the user clicks the logout button the system will go
          to the login window again
          # call the login function inside this function

     def quit(self):
          # if the user clicks the cross button the system will be
          stopped running

     def profileSetting(self, username, password, contactDetail):
          # use tkinter.get() to get values from input entries
          # then use file.write to write these values to the json
          file
          # when adding a new information to the json file, we
           should check whether there is a duplicate username or not
          # if there is a duplicated name; print a message to tell
          the user to use another name

     def changePassword(self, newPassword):
          # use tkinter.get() to get the old and new passwords from
          input entries
          # use regular expression to match the old password with
          the passwords in the json file
          if match:
               # use regular expression, match_object.span(), to get
               the starting index of the old password in the json
               file
               # password = newPassword
               # then use file.write to override the old password
               with the new password at that index in the json file
          else:
               # print a warning message and tell the user to try
               again
```

4.0 User Interface Design

In today modern world, the phrase “User Interface Design” has a broad meaning because it covers three main aspects:

- Graphical User Interface (GUI): this aspect tells how the user interacts with the system using digital control panels, such as keyboard and mouse.
- Voice Controlled Interface (VUI): this aspect allows the user to interact with the system using voice commands, such as Siri on iPhone and Google Assistance on Android.
- Gesture Base Interface (GBI): this aspect allows the user to interact with the system using bodily motion.

For this project, this team will focus mainly on Graphical User Interface Design. To visualise the initial design of this analytical software, this team utilised the advancement of “WPS” software, draw.io, and “balsamiq.cloud” platform. Moreover, to keep this software simple and consistent, this team also uses common GUI elements which allows the users to learn how to understand this software seamlessly.

GUI elements that are used in this project include but are not limited to:

- Input Components: input entries, text fields, date fields, search boxes, and dropdown lists.
- Navigational Components: buttons
- Informational Components: icons, widow titles, and message boxes

4.1 Structural Design

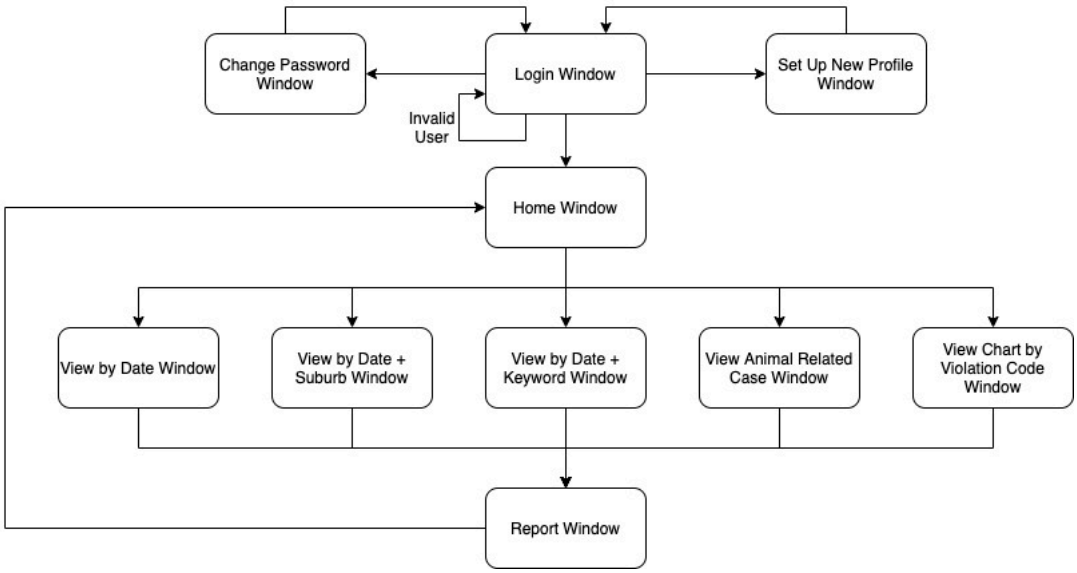

Figure 4: Structural Design Diagram

The figure 4 above shows a hierarchical schematic diagram that constructs this analytical software. Each rectangle in the figure represents a window in this software, and the order of

their presentation in the software is illustrated by arrows. In the software, each window will have its own interface. When the users execute the software, a “Login Window” will be appeared. Then the users can log in to the software if they already have an account. If they do not have an account, they can set up a new profile first and log in again. This software also allows the users to change their passwords at any time. Note that the users must provide a valid credential before they can log in to the “Home Window”. The users can navigate to other interfaces or windows through the “Home Window”. These windows include “View by Date Window”, “View by Date + Suburb Window”, “View by Date + Keyword Window”, “View Animal Related Case Window”, and “View Chart by Violation Code Window”. The users can also generate a standard report from each of these interfaces or windows. Lastly, on the report window, the users can either go back to the previous window or return to the “Home Window”. This software structure design is simple but has strong practicability and meets the final requirements of this project.

4.2 Visual Design

The Visual Design of this analytical software is drawn based on the concept of simplicity and practicality. This team has used appropriate sizes of fonts, icons, buttons, input entries, and search boxes. In terms of colours, this team used a white background and black components because it makes the users easy to read. For the icons, the light green colour is being used.

The software contains a total of 11 interfaces, and the GUI elements of each interface have been shown in the following figure.

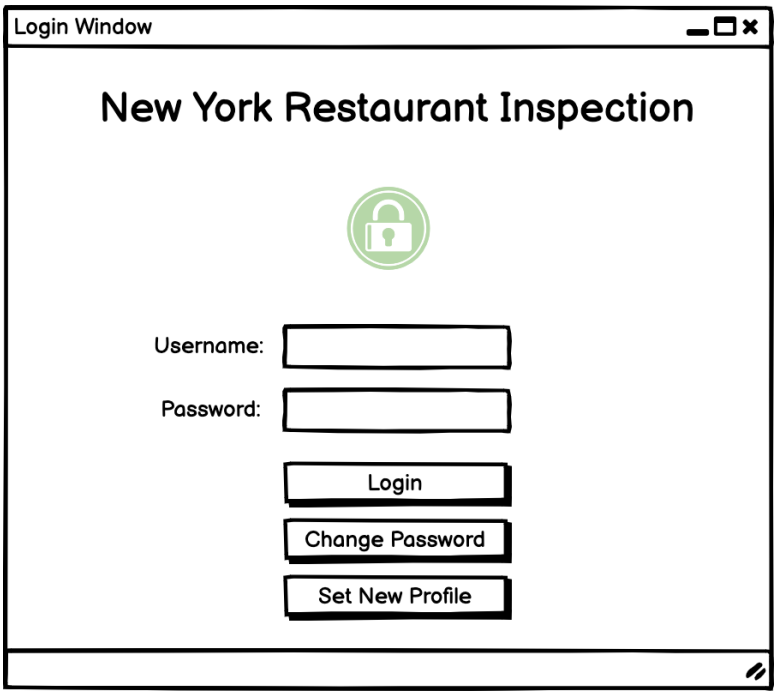

Figure 5: Login Window

The Login Window allows the user to login, change password, and set new profile.

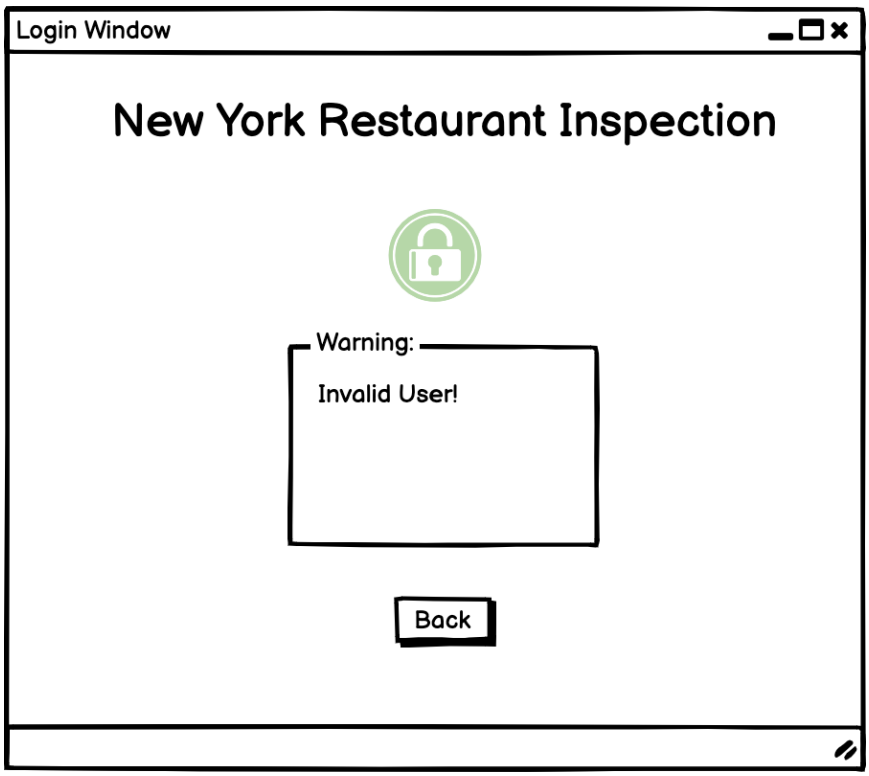

Figure 6: Warning Message-Login Window

When the user gives an invalid credential, the system will post a warning message on the Login Window.

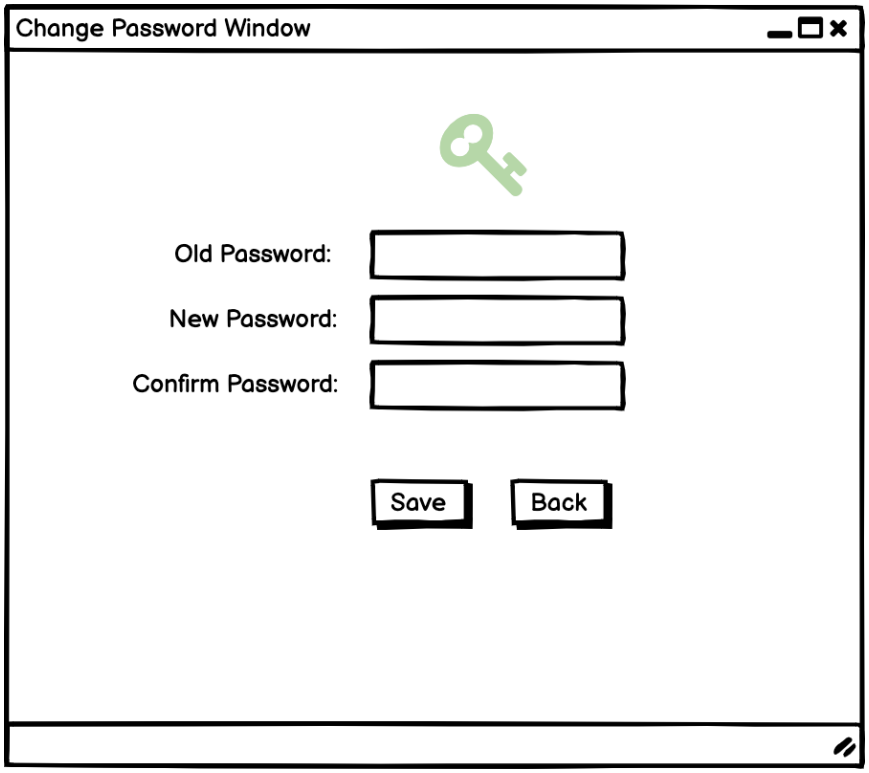

Figure 7: Change Password Window

The user can also change his password with this software if he needs to.

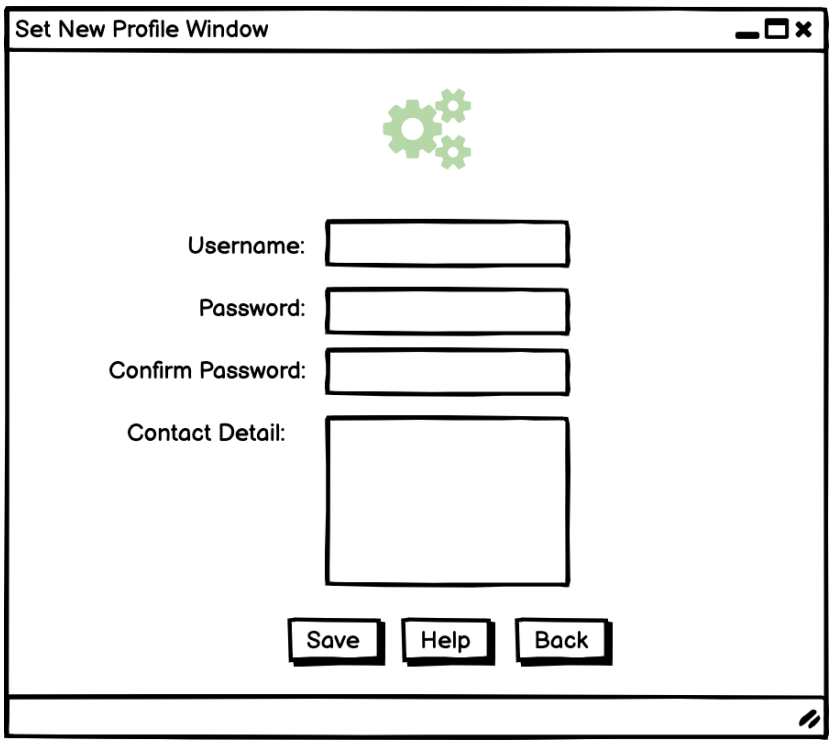

Figure 8: Set New Profile Window

The user needs to set up a new profile in Set Up New Profile Window before he can log in to the system.

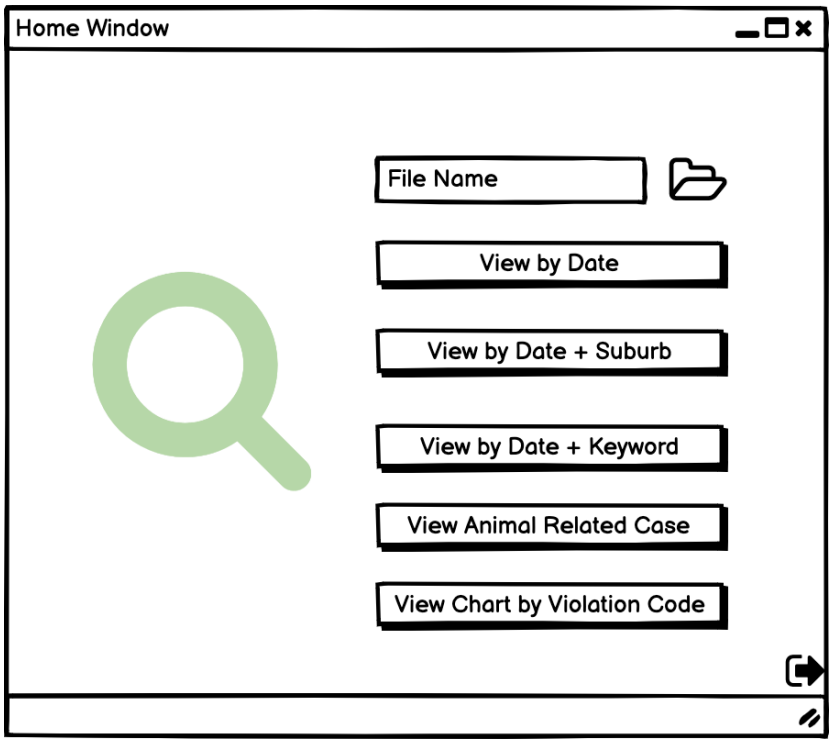

Figure 9: Home Window

After logging in, the system brings the user to the Home Window. At this point, the user might need to open a dataset file first; then chooses an option. There is a logout button on this window that allows the user to logout as well.

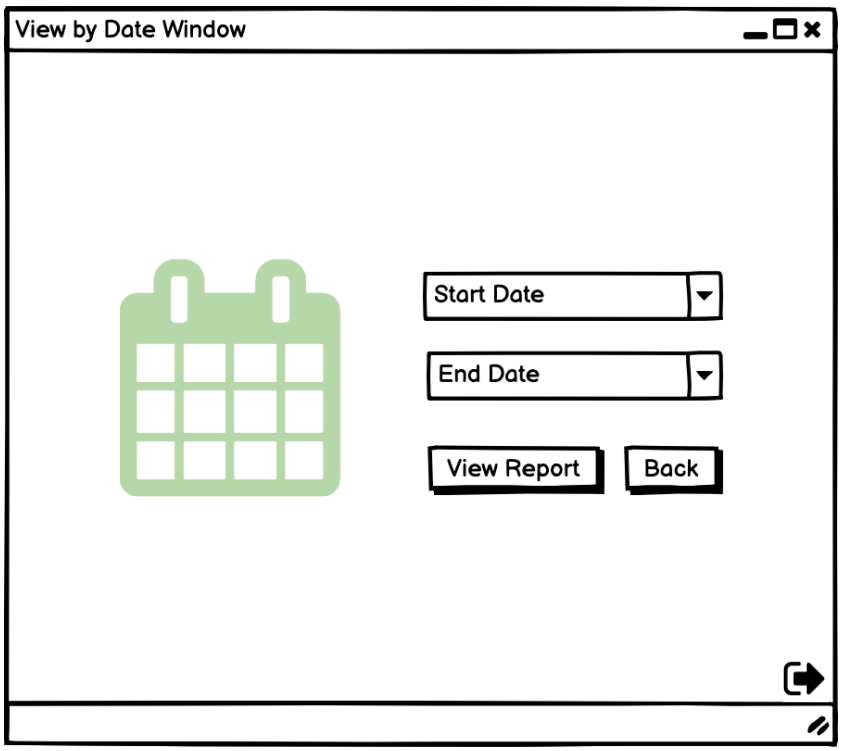

Figure 10: View by Date Window

In the View by Date Window, the user can fill in the start date and end date to view the report.

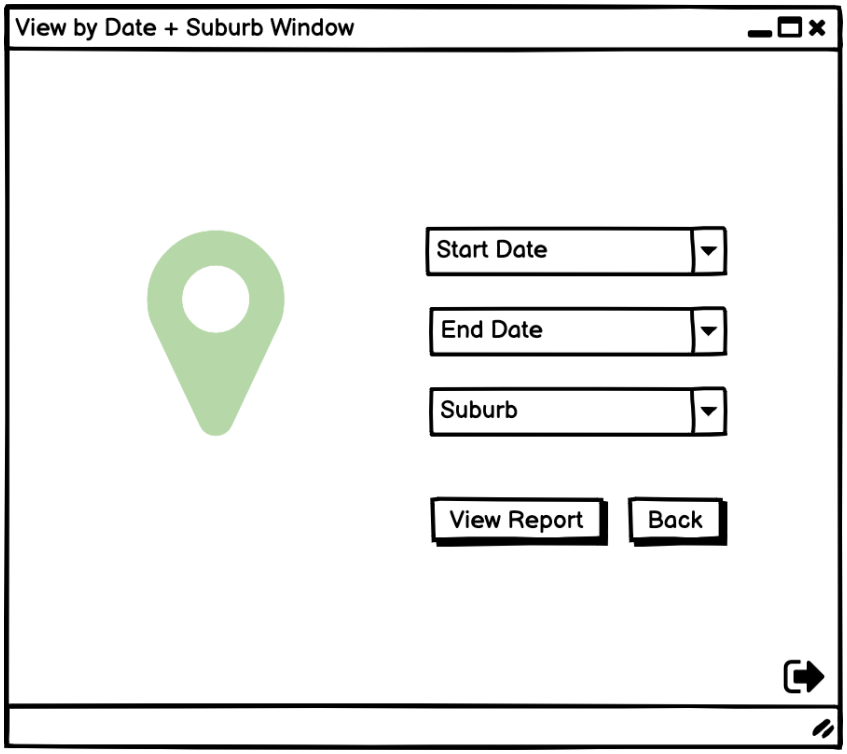

Figure 11: View by Date + Suburb Window

The View by Date + Suburb Window allows the user to search for the violation details based on the selected date and suburbs.

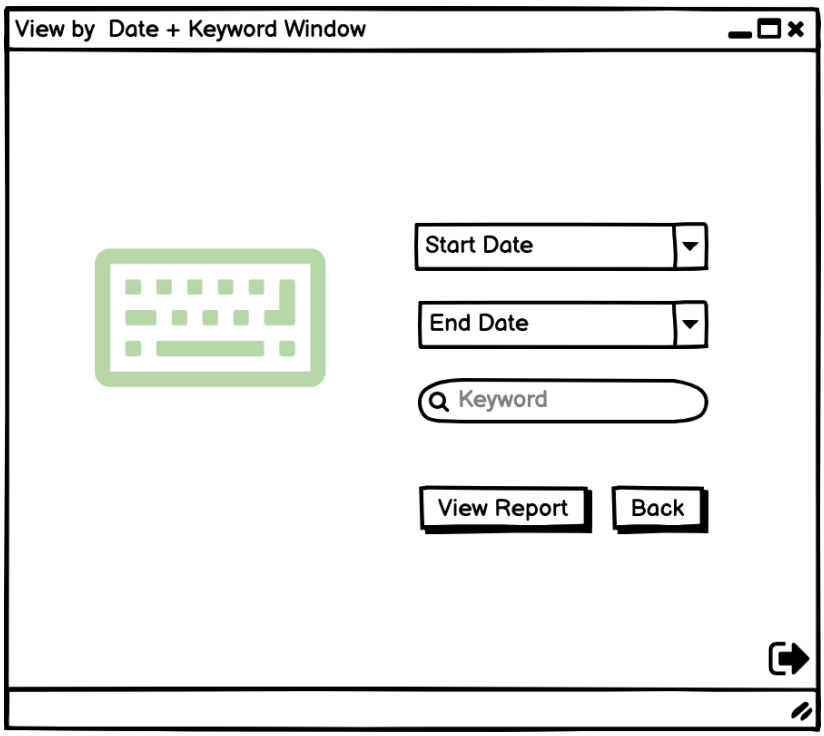

Figure 12: View by Date + Keyword Window

The View by Date + Keyword Window allows the user to used keyword input to search for specific information.

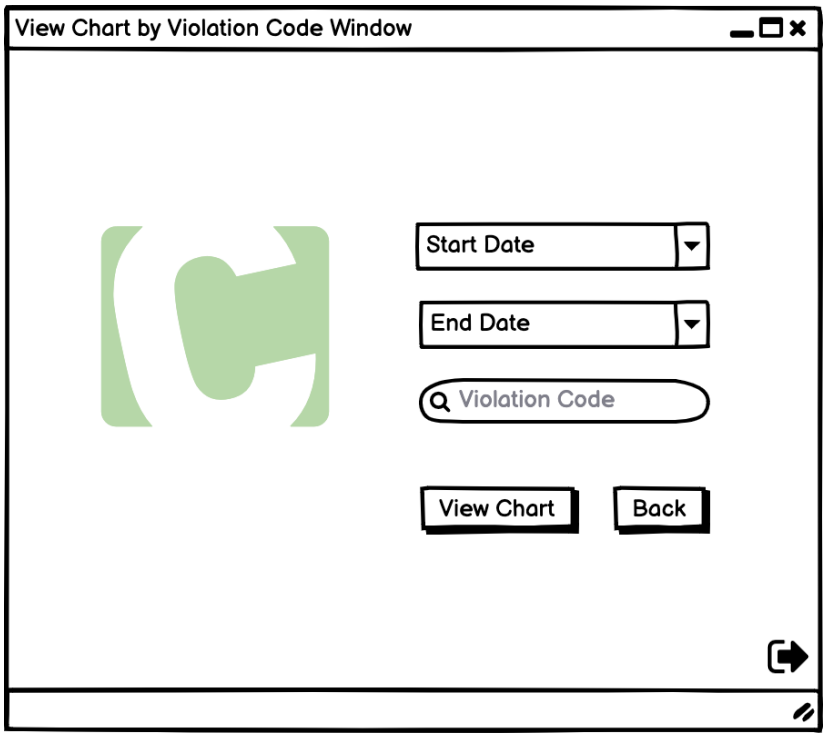

Figure 13: View Chart by Violation Code Window

The View Chart by Violation Code Window allows the user to generate a chart based on the violation code.

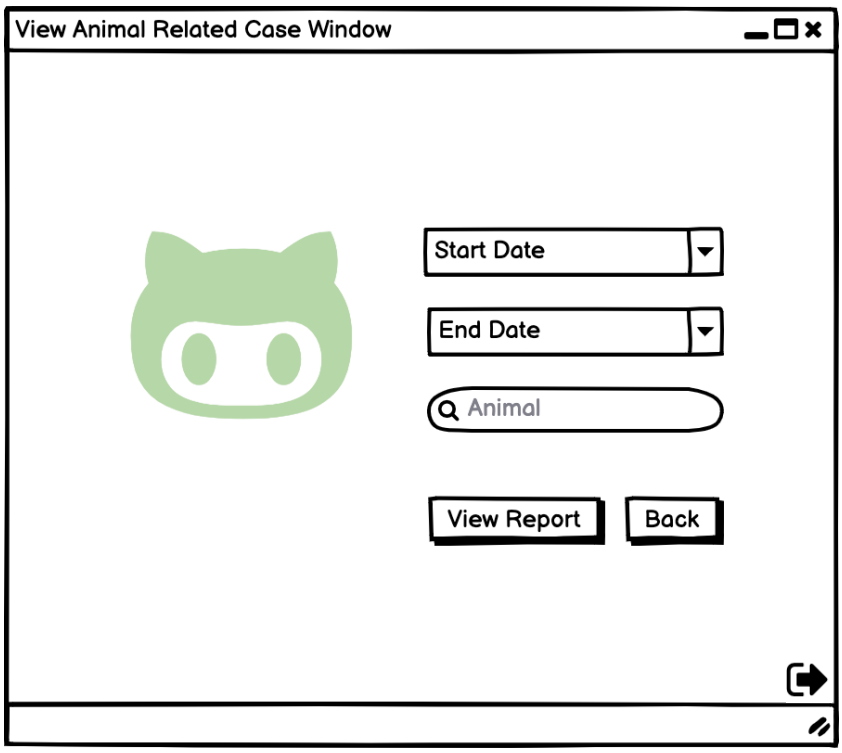

Figure 14: View Animal Related Case Window

The View Animal Related Case Window allows the user to view reports about animal, such as mice and rats which could lead to food contamination.

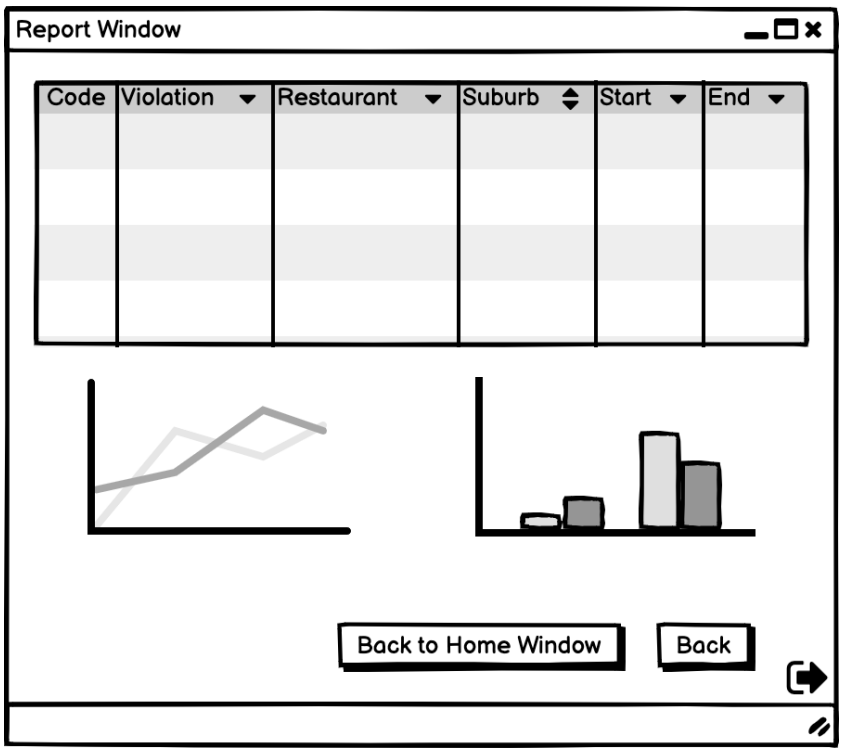

Figure 15: Report Window

The report window can print different reports and charts on the screen based on the user's options.