Software Design Document

NYC Restaurant Inspection Analysis Tool

Liangxian Zhao s2869753

Rafael Alexander s5277157

Table of Contents

[1.0 System Vision 3](#_Toc113218550)

[1.1 Problem Background 3](#_Toc113218551)

[1.2 System Overview 3](#_Toc113218552)

[1.3 Potential Benefits 3](#_Toc113218553)

[2.0 Requirements 4](#_Toc113218554)

[2.1 User Requirements 4](#_Toc113218555)

[2.2 Software Requirements 4](#_Toc113218556)

[2.3 Use Cases & Use Case Diagrams 4](#_Toc113218557)

[3.0 Software Design and System Components 9](#_Toc113218558)

[3.1 Software Design 9](#_Toc113218559)

[3.2 System Components 9](#_Toc113218560)

[3.2.1 Functions 9](#_Toc113218561)

[3.2.2 Data Structures / Data Sources 12](#_Toc113218562)

[3.2.3 Detailed Design 14](#_Toc113218563)

[4.0 User Interface Design 15](#_Toc113218564)

[4.1 Structural Design 16](#_Toc113218565)

[4.2 Visual Design 17](#_Toc113218566)

# System Vision

## Problem Background

The dataset of New York Restaurants shows digital record of restaurants inspection, including the violation of subarea, record date, inspection type (animal control), as well as score of restaurants. The potential problem can be detected by raw data collection from audits who are not able to view the result directly. It is time consuming for audits processing the raw data by key words searching. Our team's aim to solve those problems, specifying the feature of software regarding on customers ‘expectation.

## System Overview

This project is intended to be an analysis tool for the inspection of New York City restaurants. The tools used to assist researchers in making sense of the data they have collected will be briefly described in the section that follows. The NYC Analysis Tool offers access to data for assessment and research purposes, including dashboards, a query builder, and summaries of significant search and restaurant results. The user can achieve the visualized data result from this tool. The main purpose of this project is to enabling client to adjust their requirements, give greater personalisation, as well strengthen relationships with them. The information in database may provide light on your clients' preferred methods of interest, their concern and much more.

## Potential Benefits

It is possible to classify the advantages one expects to get as either material or immaterial. Financial gains are an example of a tangible benefit, whereas non-monetary gains are an example of an intangible benefit.

Tangible:

* The software tool analyses the data automatically by timesaving.
* A simple automated system can also improve user’s satisfaction in order to overview of score grade of NYC restaurant.
* Data analytics may assist a restaurant in identifying problems and implementing preventative actions. For instance, use a statistical model that can anticipate future behaviours or occurrences — to determine which locations are at the most risk for violation.

Intangible

* Visualizing the database of NYC restaurant allow clients to fulfil their requirements
* Digital data analysis that monitor the total number of products bought, as well as digital grade that contribute to the city health department.
* The restaurant owner can build prediction model to determine business strategy regarding on score variables

# Requirements

## User Requirements

* User should have a working computer, a monitor, a mouse and a keyboard to access the python application
* User should have OS that can accommodate the python application
* User should know relevant keywords to the "NYC Inspection Results" dataset to input

## Software Requirements

* The program shall give the user 5 options for each feature correlating to data visualisation of the dataset
* The program shall give the user a text field to freely input a keyword(s)
* The program shall give the user start to end dates to choose from
* The program shall give the user information of the time period of the inspections (between the earliest and the latest inspection date)
* The program shall give the user a list of options for animals to choose from
* The program shall give the user a list of options for boroughs to choose from
* The program shall give the user the chance to reset the functions and corresponding filters they selected
* The program shall display the required analysation/visualisation based on the filter entered by the user in the same page
* The program shall give the user the option to continue to another session
* The program shall give the user the option to exit the application

## Use Cases & Use Case Diagrams

|  |  |  |
| --- | --- | --- |
| **Use Case #1** | | |
| Use case name | Select feature | |
| Scenario | Selects feature to use for current session | |
| Triggering Event | User opens program | |
| Brief Description | The user decides to analyse "NYC Inspection Result" dataset with a certain feature of the program | |
| User/Actor | End User (Healthcare Department Officers, Restaurant Business Owners, General Population (Customer of the Restaurants) | |
| Stakeholders | Healthcare Department Officers, Restaurant Business Owners, General Population (Customer of the Restaurants) | |
| Related use cases | Retrieve inspection details within a time period, Plot distribution of violations over different suburbs within the time period, Retrieve inspection details with violation keyword, Retrieve inspection details related to an animal, Retrieve inspection details within a borough | |
| Precondition | The program opens and does not crash | |
| Postcondition | System displays page of selected feature | |
| Flow of Activities | End User | System |
|  | 1. Opens program |  |
|  | 1.1 Displays features to choose from |
| 2. Selects feature | 2.1 Displays feature page |
| Exception conditions | 2.1 End User does not select a feature | |

|  |  |  |
| --- | --- | --- |
| **Use Case #2** | | |
| Use case name | Retrieve cases within the time period | |
| Scenario | Retrieve inspection details within a time period from "NYC Inspection Result" dataset | |
| Triggering Event | User selects on "Inspection Details (Time Period)" button | |
| Brief Description | The user decides to analyse "NYC Inspection Result" dataset within a certain time period | |
| User/Actor | End User | |
| Stakeholders | End User | |
| Related use cases | Select feature | |
| Precondition | Start Date, End Date fields and Enter button is present | |
| Postcondition | System displays "NYC Inspection Result" dataset within the time period selected by the user | |
| Flow of Activities | End User | System |
|  | 1. Selects Start Date |  |
| 2. Selects End Date |  |
| 3. Press Enter button | 3.1 Receives startDate and endDate variable |
|  | 3.2 Enters variables into inspectionDetailsTime function |
|  | 3.3 Enters output to display function |
|  | 3.4 Displays data |
| 4. View data |  |
| 5. Choose to enter new session or close program |  |
| Exception conditions | 3.4 No matches between query and SQLite data is found | |

|  |  |  |
| --- | --- | --- |
| **Use Case #3** | | |
| Use case name | Plot violation distribution within the time period | |
| Scenario | Plot the distribution of violations over different suburbs of NYC from the “NYC Restaurant Inspection” dataset within the time period selected by the user. | |
| Triggering Event | User selects on "Violation Distribution" button | |
| Brief Description | The user decides to visualise (plot) "NYC Inspection Result" dataset based on violation distribution within a certain time period | |
| User/Actor | End User | |
| Stakeholders | End User | |
| Related use cases | Select feature | |
| Precondition | Start Date, End Date fields and Enter button is present | |
| Postcondition | System plot "NYC Inspection Result" dataset based on violation distribution within a certain time period | |
| Flow of Activities | End User | System |
|  | 1. Selects Start Date |  |
| 2. Selects End Date |  |
| 3. Press Enter button | 3.1 Receives startDate and endDate variable |
|  | 3.2 Enters variables into violationDistribution function |
|  | 3.3 Enters output to plot function |
|  | 3.4 Displays plot |
| 4. View data |  |
| 5. Choose to enter new session or close program |  |
| Exception conditions | 3.4 No matches between query and SQLite data is found | |

|  |  |  |
| --- | --- | --- |
| **Use Case #4** | | |
| Use case name | Retrieve cases with violation keyword | |
| Scenario | Retrieve inspection details with violations containing a certain keyword | |
| Triggering Event | User selects on "Violation Keyword" button | |
| Brief Description | The user decides to retrieve inspections with violations containing a certain keyword from the "NYC Inspection Result" dataset | |
| User/Actor | End User | |
| Stakeholders | End User | |
| Related use cases | Select feature | |
| Precondition | The user is familiar with keywords related to violations in the "NYC Inspection Result" dataset; Keyword field and Enter button is present | |
| Postcondition | System displays "NYC Inspection Result" dataset with violations containing a keyword entered by the user | |
| Flow of Activities | End User | System |
|  | 1. Selects Enters Keyword |  |
| 2. Press Enter button | 2.1 Receives keyword variable |
|  | 2.2 Enters variables into keywordViolation function |
|  | 2.3 Enters output to display function |
|  | 2.4 Displays data |
| 3. View data |  |
| 4. Choose to enter new session or close program |  |
| Exception conditions | 2.4 No matches between query and SQLite data is found | |

|  |  |  |
| --- | --- | --- |
| **Use Case #5** | | |
| Use case name | Retrieve details related to an animal | |
| Scenario | Retrieve inspection details related to a certain animal | |
| Triggering Event | User selects on "Animal Related Cases" button | |
| Brief Description | The user decides to retrieve inspections related to a certain animal(s) from the "NYC Inspection Result" dataset | |
| User/Actor | End User | |
| Stakeholders | End User | |
| Related use cases | Select feature | |
| Precondition | Animal Type field and Enter button is present | |
| Postcondition | System displays "NYC Inspection Result" dataset with cases related to animal(s) selected by the user | |
| Flow of Activities | End User | System |
|  | 1. Selects Animal Type dropdown button |  |
|  | 1.1 Displays animal type to choose from |
| 2. Selects Animal Type |  |
| 3. Press Enter button | 3.1 Receives keywordAnimal variable |
|  | 3.2 Enters variables into animalType function |
|  | 3.3 Enters output to plot function |
|  | 3.4 Displays data |
| 4. View data |  |
| 5. Choose to enter new session or close program |  |
| Exception conditions | 3.4 No matches between query and SQLite data is found | |

|  |  |  |
| --- | --- | --- |
| **Use Case #6** | | |
| Use case name | Retrieve cases within a borough | |
| Scenario | Retrieve inspection details within a borough from "NYC Inspection Result" dataset | |
| Triggering Event | User selects on "Inspection Details (Borough)" button | |
| Brief Description | The user decides to analyse "NYC Inspection Result" dataset within a certain borough | |
| User/Actor | End User | |
| Stakeholders | End User | |
| Related use cases | Select feature | |
| Precondition | Borough field and Enter button is present | |
| Postcondition | System displays "NYC Inspection Result" dataset within the borough selected by the user | |
| Flow of Activities | End User | System |
|  | 1. Selects Borough dropdown button |  |
|  | 1.1 Displays borough to choose from |
| 2. Selects borough |  |
| 3. Press Enter button | 3.1 Receives borough endDate variable |
|  | 3.2 Enters variable into inspectionDetailsBorough function |
|  | 3.3 Enters output to display function |
|  | 3.4 Displays data |
| 4. View data |  |
| 5. Choose to enter new session or close program |  |
| Exception conditions | 3.4 No matches between query and SQLite data is found | |

**Use case diagram:**

Diagram

Description automatically generated

# Software Design and System Components

## Software Design

Diagram

Description automatically generated

## System Components

### Functions

* F1: loadData
  + This functions loads the csv data to a SQLite database
  + Parameters:
    - CSV file
    - sqlite database file name of "NYC Inspection Results" dataset
  + Side effects
    - Creation of sqlite database file with "NYC Inspection Results" dataset
  + Return value
    - -
* F2: connectSQLite
  + This functions makes a connection to the SQLite database containing the "NYC Inspection Results" dataset
  + Parameters:
    - sqlite database file name of "NYC Inspection Results" dataset
  + Side effects
    - The program is now connected to "NYC Inspection Results" SQLite Database
  + Return value
    - conn (Connection Object)
* F3: inspectionDetailsTime
  + This functions retrieve inspection details from “NYC Restaurant Inspection” dataset within the time period selected by the user.
  + Parameters:
    - conn: connection object to sqlite database
    - startDate, format: dateTime
    - endDate, format: dateTime
  + Side effects
    - Change in global variable: startDate
    - Change in global variable: endDate
    - Change in global variable: output
  + Return value
    - output: sqlite query to retrieve inspection details in the given time period selected by the user from the database
* F4: violationTime
  + This functions retrieves violations from the “NYC Restaurant Inspection” dataset within the time period selected by the user.
  + Parameters:
    - conn: connection object to sqlite database
    - startDate, format: dateTime
    - endDate, format: dateTime
  + Side effects
    - Change in global variable: startDate
    - Change in global variable: endDate
    - Change in global variable: output
  + Return value
    - output: sqlite query to retrieve violations over different suburbs of NYC in the given time period from the database
* F5: violationKeyword
  + This functions retrieves violations from the “NYC Restaurant Inspection” dataset that contains the keyword entered by the user.
  + Parameters:
    - conn: connection object to sqlite database
    - keywordViolation, format: String
  + Side effects
    - Change in global variable: keywordViolation
    - Change in global variable: output
  + Return value
    - output: sqlite query to retrieve inspection details with violations that contains the given keyword entered by the user from the database
* F6: animalType
  + This functions cases from the “NYC Restaurant Inspection” dataset that is related to animal(s) selected by the user, their trend over time and distribution over suburbs
  + Parameters:
    - conn: connection object to sqlite database
    - keywordAnimal, format: String
  + Side effects
    - Change in global variable: keywordAnimal
    - Change in global variable: output
  + Return value
    - output: sqlite query to retrieve cases related to an animal determined by the user from the database
* F7: inspectionDetailsBorough
  + This functions retrieve inspection details from “NYC Restaurant Inspection” dataset within the borough area/cuisine type selected by the user.
  + Parameters:
    - conn: connection object to sqlite database
    - borough, format: String
  + Side effects
    - Change in global variable: borough
    - Change in global variable: output
  + Return value
    - output: sqlite query to retrieve restaurants and their respective inspection details in the given borough selected by the user from the database
* F8: display
  + This functions displays the output from aforementioned features in rows using the pandas library
  + Parameters:
    - output, format: string (of sql query)
  + Side effects
    - Change in the dataset displayed by the program
  + Return value
    - Display using wxpython's ListCtrl
* F9: plot
  + This functions plots the output from violationTime function and animalType function and shows their distribution
  + Parameters:
    - output, format: string (of sql query)
  + Side effects
    - Change in the plot displayed by the program
  + Return value
    - Display using wxpython's ListCtrl

### Data Structures / Data Sources

* output
  + Type: list
  + This is the data structure that records the output of sql query from functions related to the program features
  + Data member: Retrieved from queries which are CAMIS, Restaurant Name, Borough, Building, Street, Zip Code, Phone, Cuisine Type, Inspection Date, Action, Violation Code, Critical Flag, Score, Grade, Grade Date, Record Date and Inspection Type.
  + Functions: F3, F4, F5, F6, F7, F8, F9
* NYC Restaurant Inspection dataset
  + Type: External Data Source (sqlite database)
  + This is the main dataset used in this program to retrieve all the data that is going to be analyzed and visualized by the app for the user through queries. This dataset is stored in sqlite file external to the python file.
  + Data member: CAMIS, Restaurant Name, Borough, Building, Street, Zip Code, Phone, Cuisine Type, Inspection Date, Action, Violation Code, Critical Flag, Score, Grade, Grade Date, Record Date and Inspection Type.
  + Functions: F3, F4, F5, F6, F7, F8, F9

***Extra, Data types:***

* *startDate*
  + *Type: dateTime*
  + *This is the data structure that records the date range specified by the user*
  + *Functions: F3, F4*
* *endDate*
  + *Type: dateTime*
  + *This is the data structure that records the date range specified by the user*
  + *Functions: F3, F4*
* *keywordViolation*
  + *Type: string*
  + *This is the data structure that records the keyword specified by the user*
  + *Functions: F5*
* *keywordAnimal*
  + *Type: string*
  + *This is the data structure that records the animal type specified by the user*
  + *Functions: F6*
* *borough*
  + *Type: string*
  + *This is the data structure that records the borough specified by the user*
  + *Functions: F7*

### Detailed Design

* Pseudocode for "Retrieve cases within the time period" feature:

Retrieve start date and end date value from user input

Pass the values into the inspection details based on time function

Receive the output

**If** output is empty:

Display "No results found"

**Else**:

Pass the output into display function

* Pseudocode for "Plot violation distributions within the time period" feature

Retrieve start date and end date value from user input

Pass the values into the violation distribution function

Receive the output

**If** output is empty:

Display "No results found"

**Else**:

Pass the output into display function

* Pseudocode for "Retrieve cases with violation keyword" feature

Retrieve keyword value from user input

Pass the values into the violation distribution function

Receive the output

**If** output is empty:

Display "No results found"

**Else**:

Pass the output into plot function

* Pseudocode for "Retrieve cases related to an animal" feature

Retrieve animal type value from user input

Pass the values into the animal related cases function

Receive the output

**If** output is empty:

Display "No results found"

**Else**:

Pass the output into plot function

* Pseudocode for " Retrieve cases within a borough" feature

Retrieve borough value from user input

Pass the values into the inspection details based on borough function

Receive the output

**If** output is empty:

Display "No results found"

**Else**:

Pass the output into display function

# User Interface Design

The user interface design for this project can be divided by two components (command line interface and graphical user interfaces). Firstly, the command line interface enables users to type in commands that can produce results immediately. This strategy is speedy and straightforward for those who are accustomed to working with command lines. For example, A batch file is another method that may be used to automate tasks. However, this computer-readable process can be defined as function on python programming language, in order to complete the request that clients are seeking.

Secondly, the nature of graphical user interfaces, which is autonomous from the application functionality, sometimes known as the "skin,". an operating system or application software may be altered at any time. Standard visualized formats and text are an additional component of the traditional graphical user interface. These formats are feasible for programmers to communicate data with one another while operating under graphical user interface design software of the same kind. Moreover, testing graphical user interface system (also known as GUI testing) is a methodical procedure that involves the generation of test cases in order to assess both the practicality of the system and its component designs. The testing tools for graphical user interfaces may either be manual or automated, and they are often built by third-party operators, such as Squish GUI Tester, Unified Functional Testing (UFT).

## Structural Design

Diagram

Description automatically generated

In the above flowchart of the structural design of the program, the navigational and information structure of the program starts from a singular and collective page which will then divide to pages based on the 5 features of the program. This is to simplify and to make clear what each feature does and which feature and results the end user wants to get from the program. It was considered that if all the 5 features were combined to one page it would make confusion to the end user of what the scope of the program actual is and the end user may try to combine elements of each feature to get their ideal result. Each feature leads to a different page with its appropriate input fields as described in the flowchart. After the end user has given their input and pressed the search button, the program will display the appropriate data frame/plot on the same page. This is to avoid confusions to results in the event where several result pages may be displayed after a few sessions are done. After a session is done, the user may choose to exit the program or do another session which will lead the end user to the first singular and collective page which is the Choose Feature page. Even though the end user needs to finish their search before entering a new session, this structure simplifies and makes it clear what the scope of the program actually is.

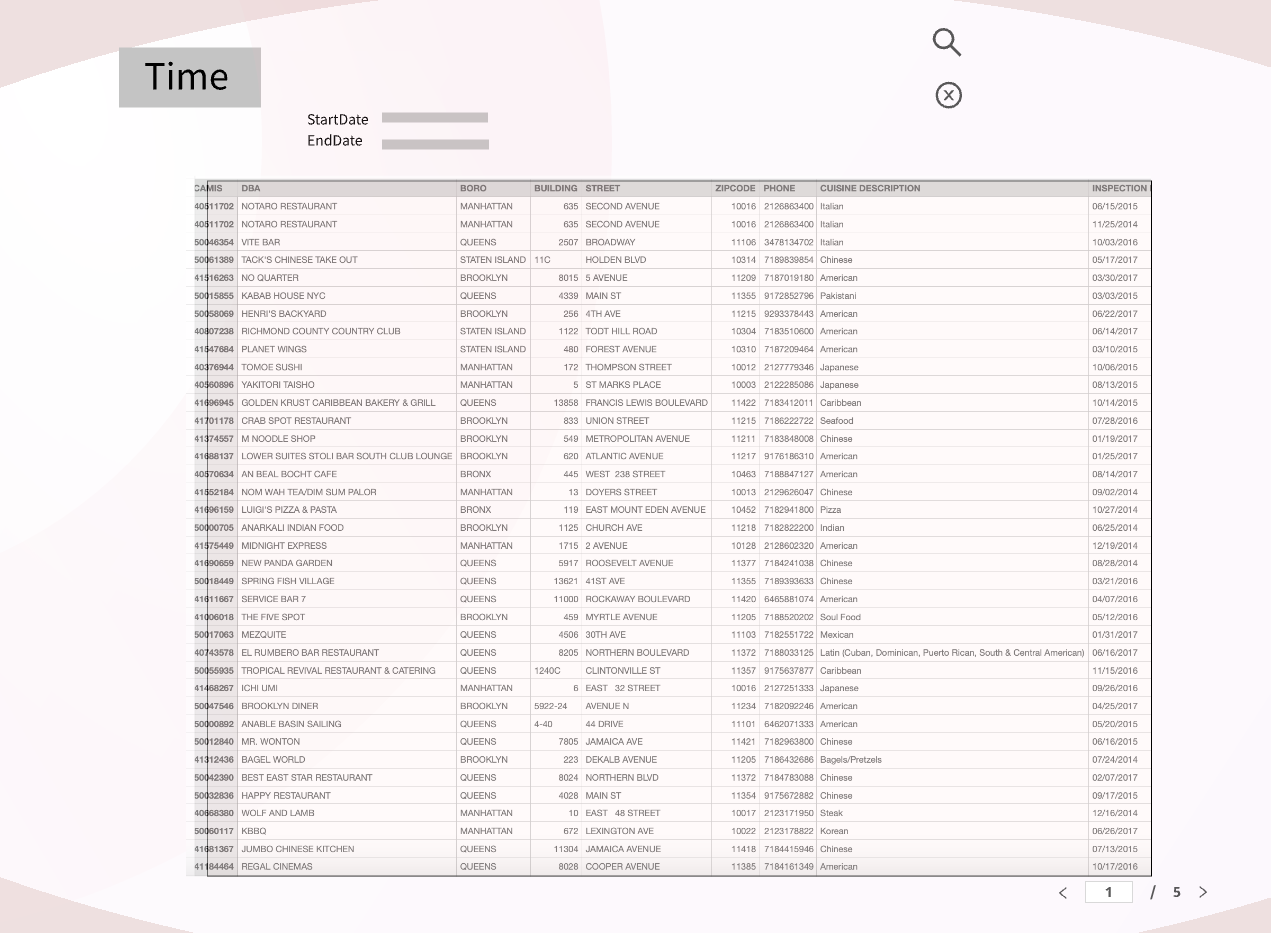
## Visual Design

The goal of this approach is to make it simpler for the human brain to comprehend and get insights from data. The primary objective of data visualization is to simplify the process of locating and understanding patterns, trends, and outliers in massive data sets. This project is also considered as the practice of information converting, such as a map or graph.

The visual design of NYC Restaurant Inspection Analysis Tool is demonstrated below. The homepage is designed as two click buttons (“Find”& “Exit”), 5 blank columns that allow customers to filled up. The most effective function related to time-period can be visualized as “StartDate”and “End Data”. Our team are seeking the simple access to retrieve raw data, additionally, the function of trend can be analysis by selected time-period.



Wireframe : Choose Feature Page

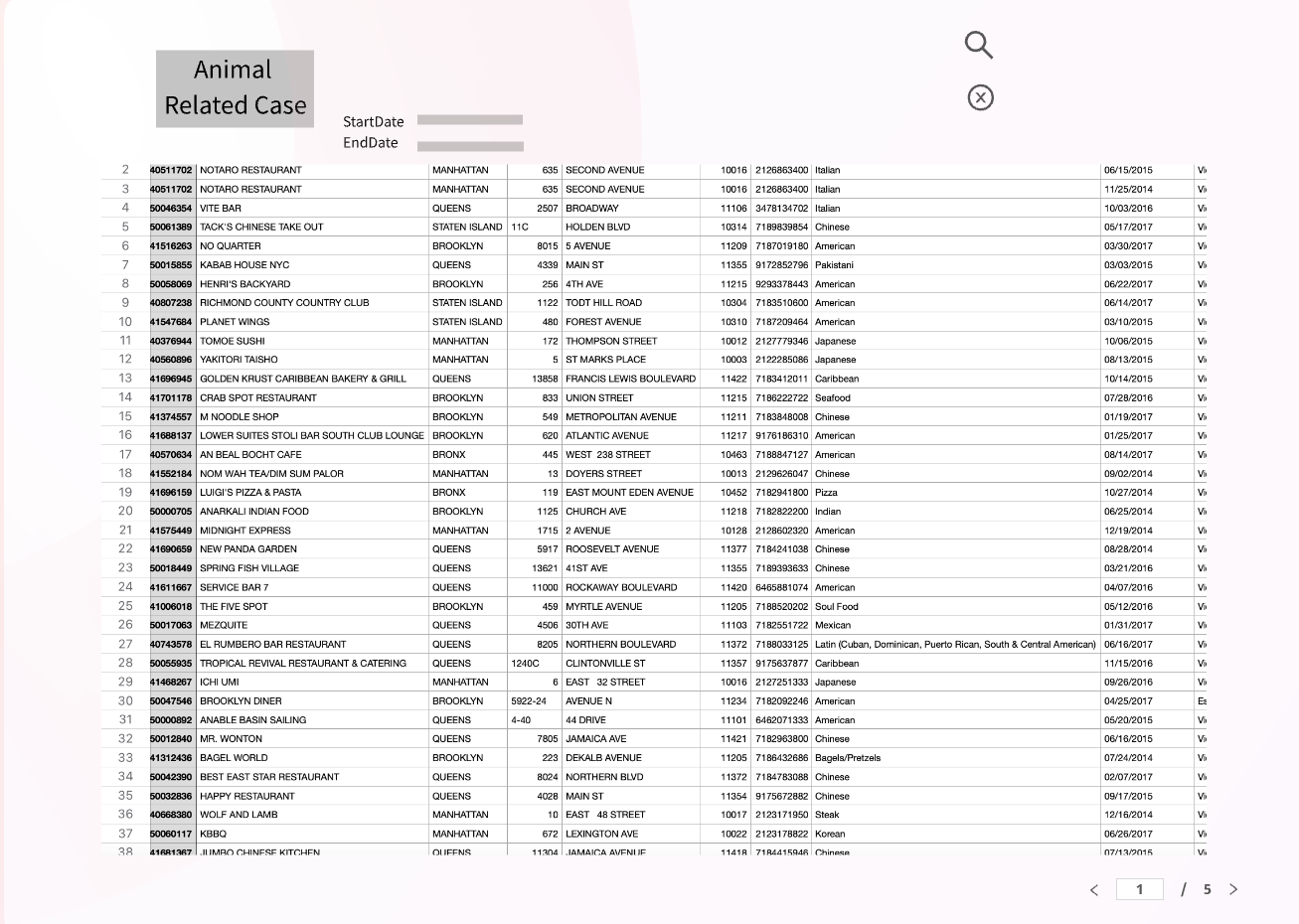


Wireframe 2: Inspection Details (Time) Page



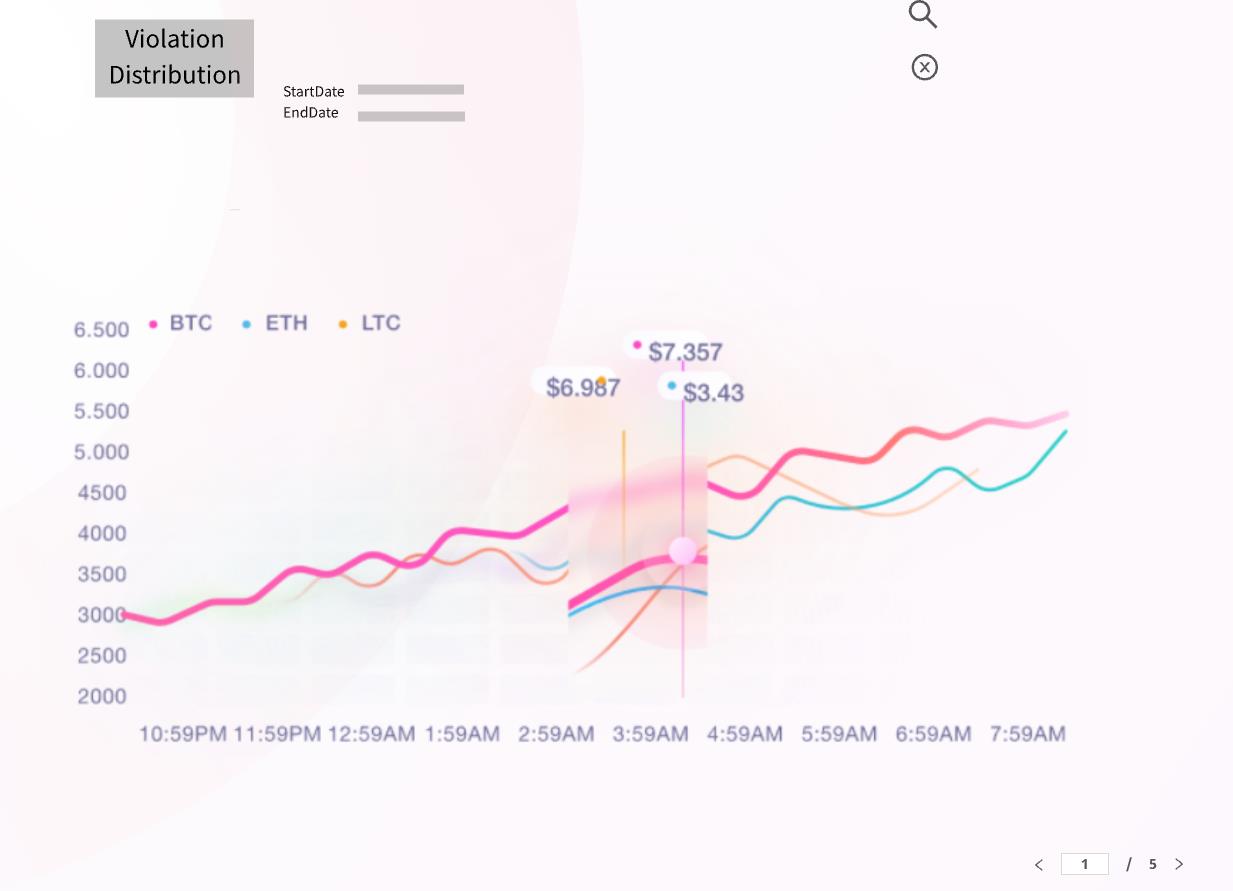
Keyword

Wireframe 3: Violation Keyword Page

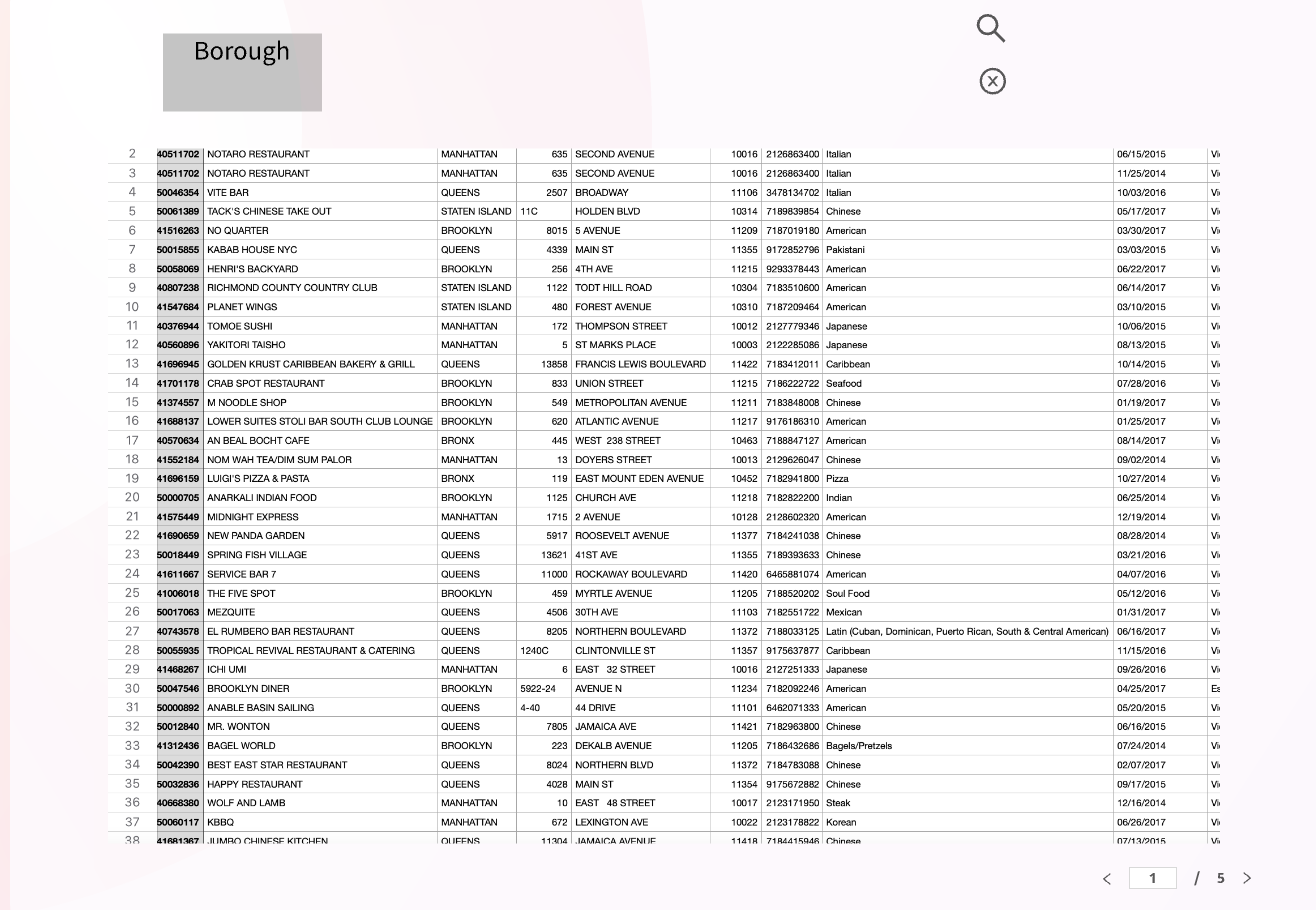


Animal Type

Wireframe 4: Animal Related Case Page



Wireframe 5: Violation Distribution Page



Wireframe 6: Inspection Details (Borough) Page