NYC Restaurant Inspection Interactive System

Group 8: Chenyang Li, Jianyu Yue, Ariel Zhou, Xinyan Wang , and Chenning Zhang

01

Background and definition of the data analytics problem

Background

NYC government has a history of storing the restaurant inspection data in its application programming interface. The inspection includes the details of violations and offenders.

Problem

However, not all the inspections data accurately reflect the actual behavior of the restaurants. Moreover, it is hard for restaurant owners to appeal or communicate with government associates on such non-interactive data, thus dampening the overall government management efficiency and generating unnecessary operating costs.

Data Analysis Case

Therefore, we plan to develop an online interactive web platform for the government to allow the restaurants' owners to search their violation history and be able to write their complaints on certain violation records (s). Then, the platform will restructure the dataset to enable the government representatives to see which violation was appealed and why, thus increasing their operation efficiency.

02 Data Source Specification and Procurement Details

- Data source is from NYC OpenData, about New York City Restaurant Inspection Results
  - API: *https://data.cityofnewyork.us/resource/43nn-pn8j.json*
- This dataset provides restaurant inspections, violations, grades, and adjudication information
- Real-time dataset, could be traced by inspection date and record date
  - Including 395,499 Rows (Restaurant Citation), 26 Columns
  - Columns including restaurants details, violation description, inspection date, inspection time, etc.
  - Adding case number (ID) for each case to make process easier
- User (Restaurants) could send complaints after they detected unreasonable violations
- User (Government) could review any complaints from restaurants

03

Implemented design choices & Technologies

Step 1: Access data from the DOHMH New York City Restaurant Inspection Results API.

- The data will be retrieved in JSON format and stored as a Python list of dictionaries for later use.

Step 2: Create a MongoDB database collection

- Since our data will be retrieved in JSON format, we will utilize MongoDB to store and manage the data.
- The current json dataset is 317.8MB and the horizontal scalability of MongoDB will allow us to work with large volumes of data presented in the Open Data API. The embedded replicas will also ensure the consistency of data.
- # of MongoDB connections made to a server: **~ 850000**

Step 3: Deploy a REST API web service using Python Flask

- We will develop a REST API and implement a frontend for the API to allow for user input.
- Restaurants will be able to retrieve inspection results, identify the record to dispute and send in comments directly using the frontend HTTP website
- The input data will be stored as a new key value pair, e.g. { Review: user input }, within the corresponding record document in original inspection data collection
- Maximum API requests per minute per user session: **500**

*Note: We can utilize Google App Engine in the future to support the volume of at least hundreds of simultaneously interacting users.

Technologies: API Data wrangling & Mongodb

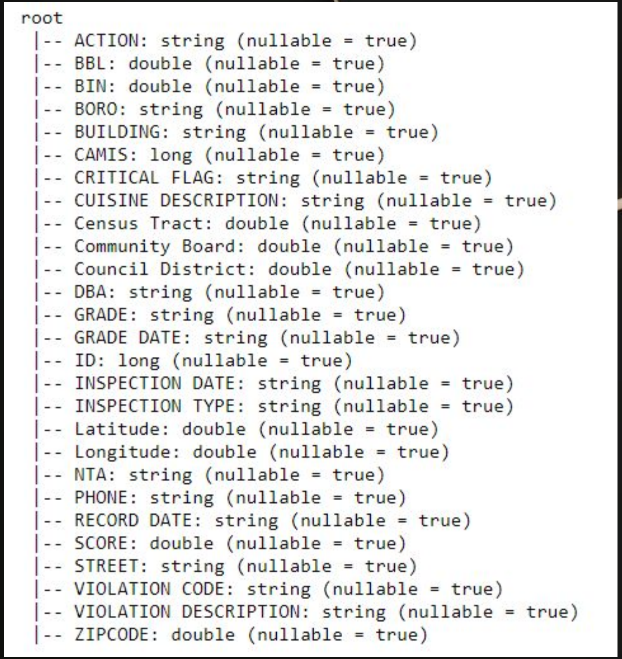
Json:

- 27 items
- 11 numerical items and 16 categorical items
- “ID” uniquely identify each record
- Length: 395499

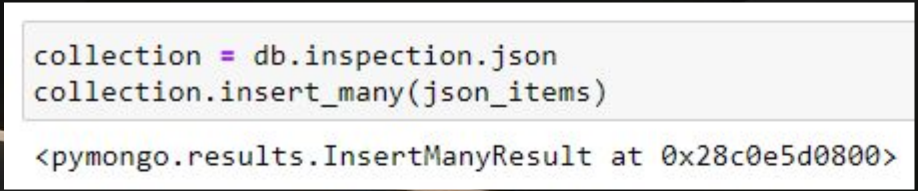
Mongodb:

- Json data is stored into variable “collection”.

*



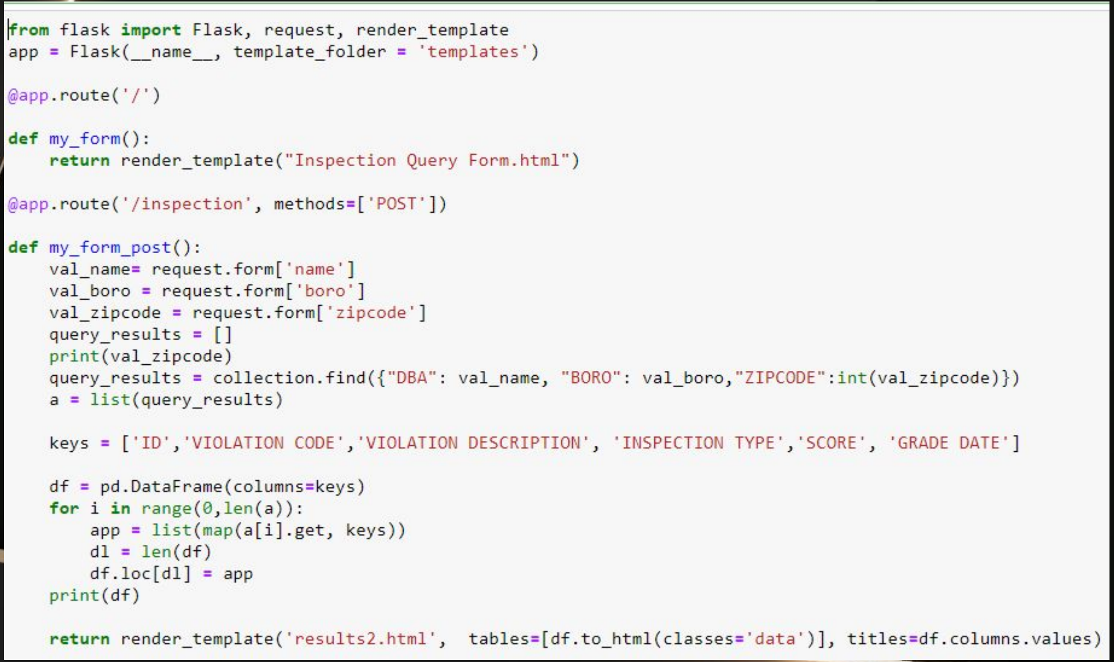
```
root
 |-- ACTION: string (nullable = true)
 |-- BBL: double (nullable = true)
 |-- BIN: double (nullable = true)
 |-- BORO: string (nullable = true)
 |-- BUILDING: string (nullable = true)
 |-- CAMIS: long (nullable = true)
 |-- CRITICAL FLAG: string (nullable = true)
 |-- CUISINE DESCRIPTION: string (nullable = true)
 |-- Census Tract: double (nullable = true)
 |-- Community Board: double (nullable = true)
 |-- Council District: double (nullable = true)
 |-- DBA: string (nullable = true)
 |-- GRADE: string (nullable = true)
 |-- GRADE DATE: string (nullable = true)
 |-- ID: long (nullable = true)
 |-- INSPECTION DATE: string (nullable = true)
 |-- INSPECTION TYPE: string (nullable = true)
 |-- Latitude: double (nullable = true)
 |-- Longitude: double (nullable = true)
 |-- NTA: string (nullable = true)
 |-- PHONE: string (nullable = true)
 |-- RECORD DATE: string (nullable = true)
 |-- SCORE: double (nullable = true)
 |-- STREET: string (nullable = true)
 |-- VIOLATION CODE: string (nullable = true)
 |-- VIOLATION DESCRIPTION: string (nullable = true)
 |-- ZIPCODE: double (nullable = true)
```



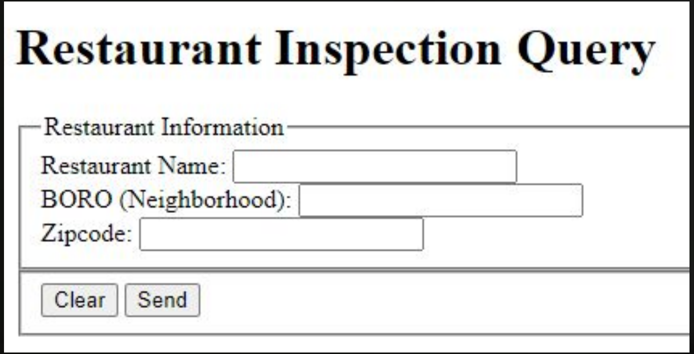
```
collection = db.inspection.json
collection.insert_many(json_items)
```

```
<pymongo.results.InsertManyResult at 0x28c0e5d0800>
```

Technologies: Flask and html



```
from flask import Flask, request, render_template
app = Flask(__name__, template_folder = 'templates')

@app.route('/')

def my_form():
    return render_template("Inspection Query Form.html")

@app.route('/inspection', methods=['POST'])

def my_form_post():
    val_name= request.form['name']
    val_boro = request.form['boro']
    val_zipcode = request.form['zipcode']
    query_results = []
    print(val_zipcode)
    query_results = collection.find({"DBA": val_name, "BORO": val_boro,"ZIPCODE":int(val_zipcode)})
    a = list(query_results)

    keys = ['ID','VIOLATION CODE','VIOLATION DESCRIPTION', 'INSPECTION TYPE','SCORE', 'GRADE DATE']

    df = pd.DataFrame(columns=keys)
    for i in range(0,len(a)):
        app = list(map(a[i].get, keys))
        dl = len(df)
        df.loc[dl] = app
    print(df)

    return render_template('results2.html',  tables=[df.to_html(classes='data')], titles=df.columns.values)
```



Restaurant Inspection Query

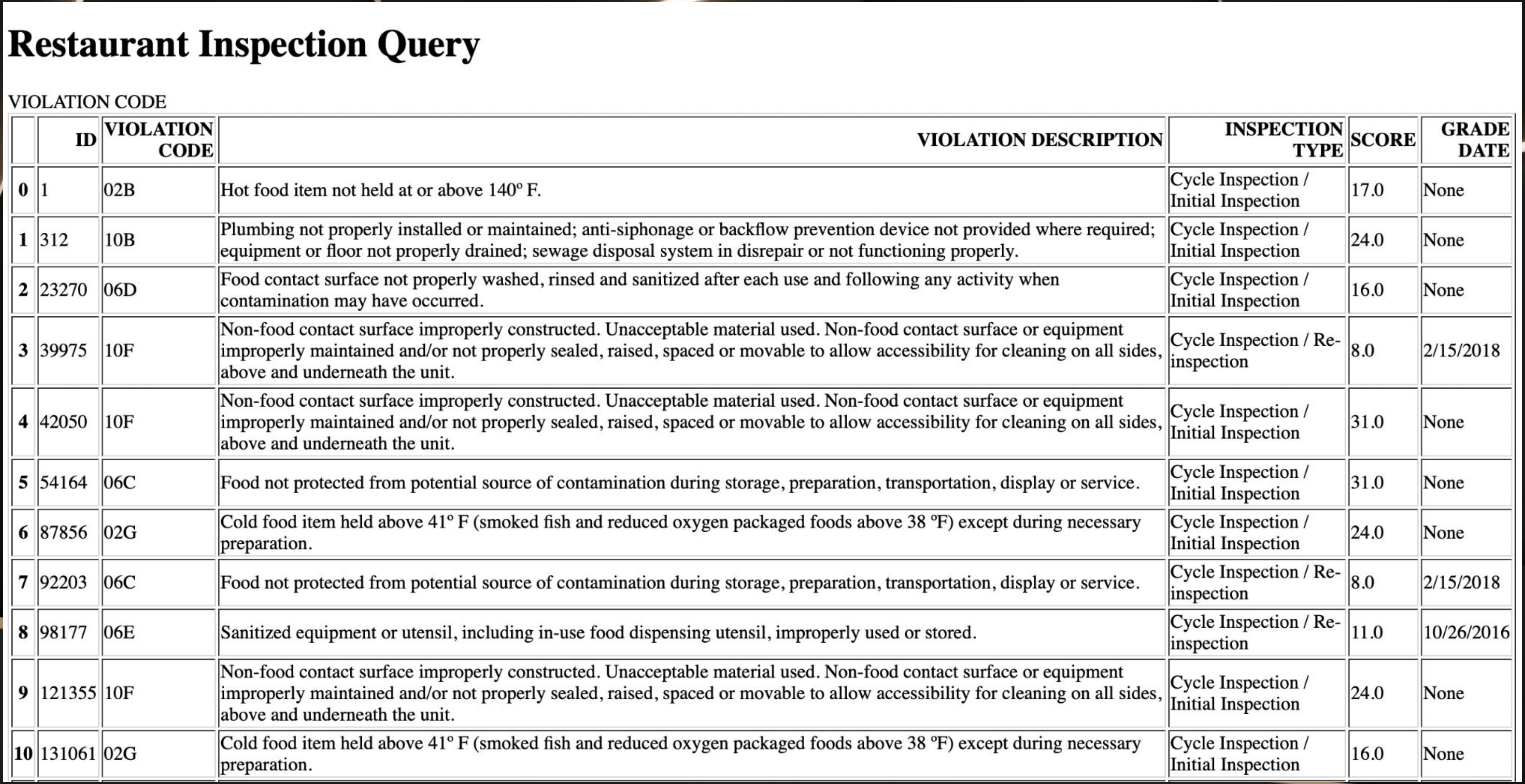
Restaurant Information

Restaurant Name:

BORO (Neighborhood):

Zipcode:

Technologies: Flask and html (cont.)

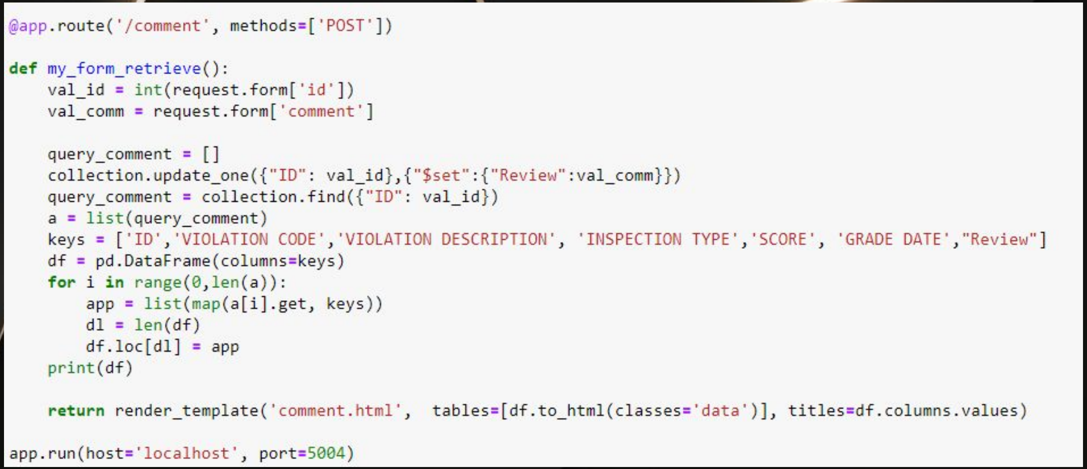


Restaurant Inspection Query

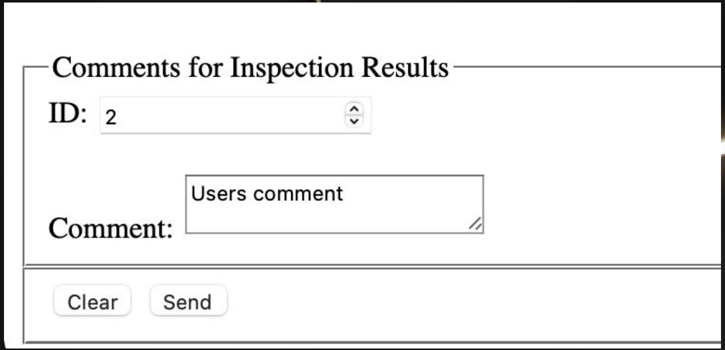
VIOLATION CODE

	ID	VIOLATION CODE	VIOLATION DESCRIPTION	INSPECTION TYPE	SCORE	GRADE DATE
0	1	02B	Hot food item not held at or above 140º F.	Cycle Inspection / Initial Inspection	17.0	None
1	312	10B	Plumbing not properly installed or maintained; anti-siphonage or backflow prevention device not provided where required; equipment or floor not properly drained; sewage disposal system in disrepair or not functioning properly.	Cycle Inspection / Initial Inspection	24.0	None
2	23270	06D	Food contact surface not properly washed, rinsed and sanitized after each use and following any activity when contamination may have occurred.	Cycle Inspection / Initial Inspection	16.0	None
3	39975	10F	Non-food contact surface improperly constructed. Unacceptable material used. Non-food contact surface or equipment improperly maintained and/or not properly sealed, raised, spaced or movable to allow accessibility for cleaning on all sides, above and underneath the unit.	Cycle Inspection / Re-inspection	8.0	2/15/2018
4	42050	10F	Non-food contact surface improperly constructed. Unacceptable material used. Non-food contact surface or equipment improperly maintained and/or not properly sealed, raised, spaced or movable to allow accessibility for cleaning on all sides, above and underneath the unit.	Cycle Inspection / Initial Inspection	31.0	None
5	54164	06C	Food not protected from potential source of contamination during storage, preparation, transportation, display or service.	Cycle Inspection / Initial Inspection	31.0	None
6	87856	02G	Cold food item held above 41º F (smoked fish and reduced oxygen packaged foods above 38 ºF) except during necessary preparation.	Cycle Inspection / Initial Inspection	24.0	None
7	92203	06C	Food not protected from potential source of contamination during storage, preparation, transportation, display or service.	Cycle Inspection / Re-inspection	8.0	2/15/2018
8	98177	06E	Sanitized equipment or utensil, including in-use food dispensing utensil, improperly used or stored.	Cycle Inspection / Re-inspection	11.0	10/26/2016
9	121355	10F	Non-food contact surface improperly constructed. Unacceptable material used. Non-food contact surface or equipment improperly maintained and/or not properly sealed, raised, spaced or movable to allow accessibility for cleaning on all sides, above and underneath the unit.	Cycle Inspection / Initial Inspection	24.0	None
10	131061	02G	Cold food item held above 41º F (smoked fish and reduced oxygen packaged foods above 38 ºF) except during necessary preparation.	Cycle Inspection / Initial Inspection	16.0	None

Technologies: Flask and html (cont.)



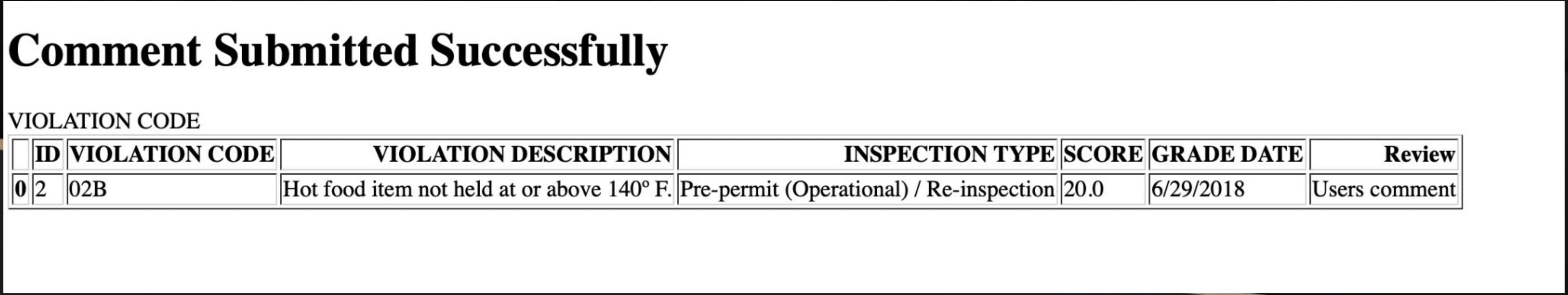
```
@app.route('/comment', methods=['POST'])

def my_form_retrieve():
    val_id = int(request.form['id'])
    val_comm = request.form['comment']

    query_comment = []
    collection.update_one({"ID": val_id},{"$set":{"Review":val_comm}})
    query_comment = collection.find({"ID": val_id})
    a = list(query_comment)
    keys = ['ID','VIOLATION CODE','VIOLATION DESCRIPTION', 'INSPECTION TYPE','SCORE', 'GRADE DATE',"Review"]
    df = pd.DataFrame(columns=keys)
    for i in range(0,len(a)):
        app = list(map(a[i].get, keys))
        dl = len(df)
        df.loc[dl] = app
    print(df)

    return render_template('comment.html',  tables=[df.to_html(classes='data')], titles=df.columns.values)

app.run(host='localhost', port=5004)
```



Comments for Inspection Results

ID: 2

Comment: Users comment



Comment Submitted Successfully

VIOLATION CODE

	ID	VIOLATION CODE	VIOLATION DESCRIPTION	INSPECTION TYPE	SCORE	GRADE DATE	Review
0	2	02B	Hot food item not held at or above 140º F.	Pre-permit (Operational) / Re-inspection	20.0	6/29/2018	Users comment

04

Applicable data governance policies & cost implications

We are using the NYC open data, which is a publicly available data.

Our data does not involve any personal information (PI) or personally identifiable information (PII).

There are no cost implications for our developer version of API services.

If we find out our dataset is too large, and need cloud environment to process and storage, we will use Google BigQuery. Our monthly cost would be $10.

We may use Google App Engine to build scalable web, mobile back ends, and let our customers use our application. The cost estimated to be $150 per month. We also estimate to have a total of $5100 labor and utilities cost each month.

	Cost per month (estimate)
Labor costs	$5000
Google Cloud BigQuery	$10
Website service rental	$20
Google App Engine	$150
Utilities	$100
TOTAL	$5280

*Our cost implications may change depending on the current situation

05

Assessment of Success Based on the Proposed Metrics

- Since we build an interactive online web platform with a search function. We have to use some metrics to measure the success of the website.
    - Retention rate means a number to percentage of users who still use the website in a certain time period.
    - According to the study by Mixpanel, the range of average retention rate for most website is 6% to 20% in eight weeks depends on the industry.
- Search bar plays a significant role in this system. Our main success criteria are retrieval of time, and the relevance of the result.
    - Click through rate (CTR): Users who clicked on a search result to the total number of the users who viewed search results. CTR measures the relevance of the result. Average search network CTR is 3.7%.
    - Latency is one of the main metrics that is used to measure the retrieval of the time. Based on a research written by Jaime Teevan , the average service side delay range is from 100 to 400 millisecond.

06

Conclusion, Limitation and Future Recommendations

- Data security problem need to be solved.
  -Verification system should be built

- Time-sensitive
  -We need to build a system that can automatically inform restaurant owns when a new inspection is completed, and inform government when a new complaint is submitted.

- Follow up system
  -to follow up with more questions or details

THANKS!