****

**DOT OSAV Task order 693JK419F980013**

**Task 3: Transportation Data Discovery, Evaluation, Compilation, and Publication**

**Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Author** | **Date of Release** | **Version** | **Approved by** | **Changes** |
| Zhi Liu | 11/26/2019 | 1.0 |  | Original |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Table of Contents**

[**1 PROJECT TITLE 1**](#_heading=h.30j0zll)

[**2 INTRODUCTION 1**](#_heading=h.1fob9te)

[**3 Plan and milestone 2**](#_heading=h.3znysh7)

[**4 SYSTEM ARCHITECTURE 3**](#_heading=h.2et92p0)

[4.1 System Architecture Design 3](#_heading=h.tyjcwt)

[4.2 Hardware and Software 3](#_heading=h.3dy6vkm)

[4.2.1 Hardware 3](#_heading=h.1t3h5sf)

[4.2.2 Software 3](#_heading=h.4d34og8)

[**5 Methodologies 4**](#_heading=h.2s8eyo1)

[5.1 Data Discovery 4](#_heading=h.17dp8vu)

[5.1.1 Identify the required data from internet 4](#_heading=h.3rdcrjn)

[5.1.2 NTD agency information collection 4](#_heading=h.26in1rg)

[5.1.3 GTFS data download URLs identification 5](#_heading=h.lnxbz9)

[5.2 Evaluation 13](#_heading=h.35nkun2)

[5.3 Compilation 14](#_heading=h.1ksv4uv)

[5.4 Publication 15](#_heading=h.44sinio)

[**6 CONSIDERATIONS 15**](#_heading=h.2jxsxqh)

# PROJECT TITLE

Project Code: **DOT OSAV Task order 693JK419F980013**

Task 3 Name: Transportation Data Discovery, Evaluation, Compilation, and Publication

# INTRODUCTION

**Detailed Tasks:**

* Identify and collect data from the Internet
* The data collected will include, but not be not limited to:

1. transportation safety across all modes and intermodally;
2. the state of good repair of United States transportation infrastructure;
3. the extent, connectivity, and condition of the transportation system
4. economic efficiency across the entire transportation sector;
5. the effects of the transportation system on global and domestic economic competitiveness;
6. demographic, economic, and other variables influencing travel behavior, including choice of transportation mode and goods movement;
7. transportation-related variables that influence the domestic economy and global competitiveness;
8. economic costs and impacts for passenger travel and freight movement;
9. intermodal and multimodal passenger movement;
10. intermodal and multimodal freight movement; and
11. consequences of transportation for the human and natural environment;

* Identify, store, and track the transportation data authorities responsible for publishing the data
* Identify, store, and track the Uniform Resource Locators (URLs) at which valid transportation source data exist
* Evaluate the data collected
* Compile the data collected
* Publish the data collected

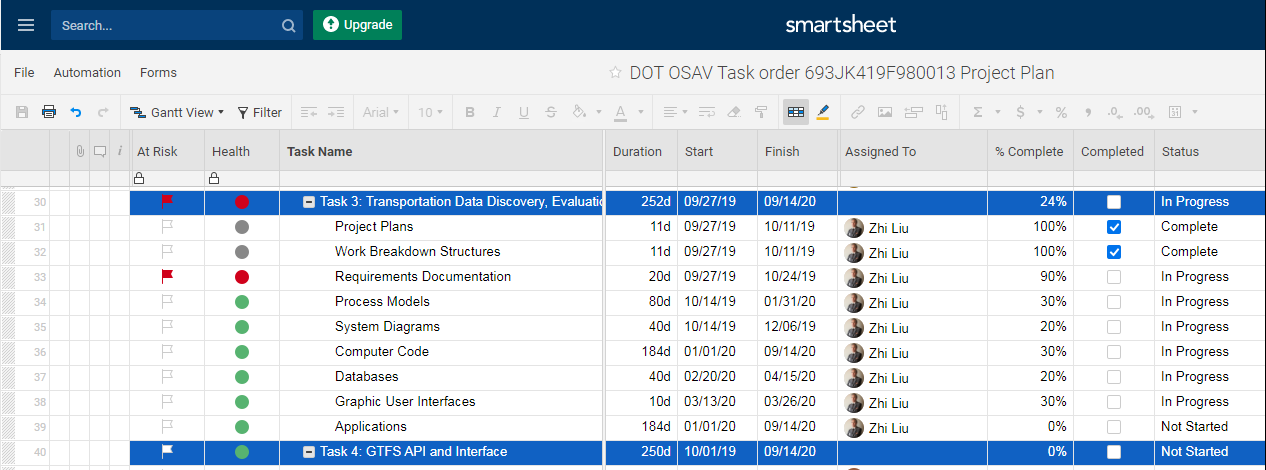
Provide metrics on the success or failure of data identification, collection, evaluation, compilation, and publication

# Plan and milestone

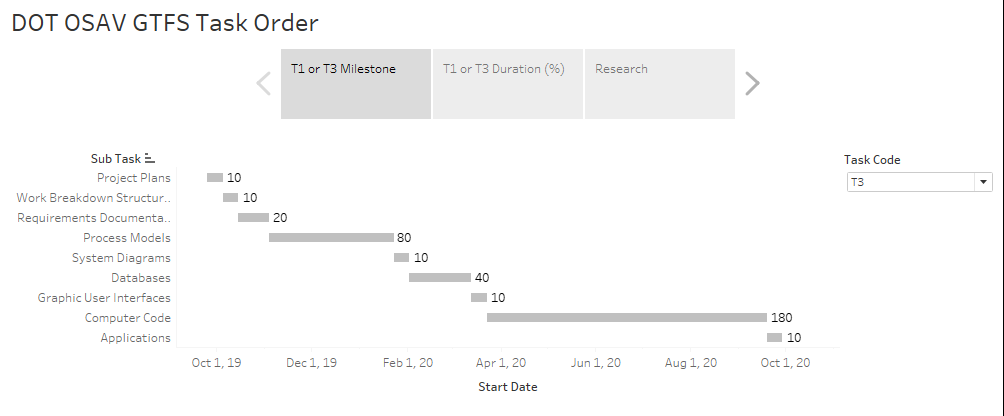
The working plan for Task 3 have been worked out and documented using Smartsheet (progress tracking) and Tableau (see the screenshots), and they can be accessed via the below links:

Smartsheet: <https://app.smartsheet.com/sheets/qjhqPpg3G3HWRcw7WwV3cm9cmjmG8jjPw3MQccC1?view=gantt>

Tableau: <https://public.tableau.com/profile/zhi.liu#!/vizhome/T1orT3Milestones/DOTOSAVGTFSTaskOrder>



*Figure 1 Plan and breakdown sub-tasks tracking for Task 3 in Smartsheet*

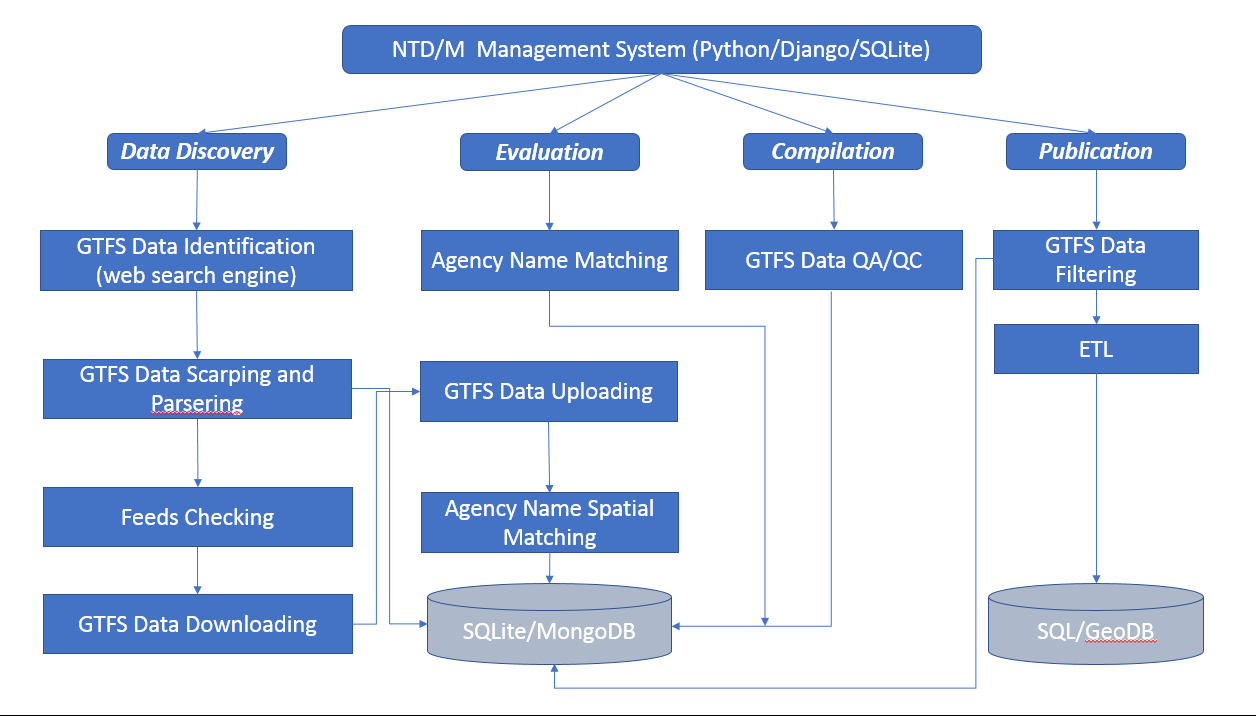


*Figure 2 Breakdown sub-tasks and milestone for Task 3 in Tableau*

# SYSTEM ARCHITECTURE

## System Architecture Design

After extensive research, the draft version system architecture design has completed to reach the goals (modifications will be made on a regular basis if needed).



*Figure 3 Draft version of system architecture*

## Hardware and Software

### Hardware

* DELL Laptop or Desktop (at least 16 GB RAM, Disk Size: 1 TB)

### Software

* Python (free)
* Django (free)
* SQLite (free) or MS SQL (for dev)
* MongoDB (free)
* PyCharm (community: free; need to install professional)
* Jupyter Notebook (free)
* Eclipse (Java, Tomcat)
* Tableau
* ESRI Suit (ArcGIS Pro etc.)

# Methodologies

## Data Discovery

### 5.1.1 Identify the required data from internet

After extensive research, we found that the Python and third party packages (web scraping, search engines, machine learning etc.) have been selected as tools for the data sources identification and collection.

PyCharm and Jupyter Notebook have been selected as IDEs, Django was selected as a framework for web-based interface.

Identified major data sources:

<https://www.transit.dot.gov/ntd/ntd-data>

<https://www.nationaltransitdatabase.org/>

<http://osav-usdot.opendata.arcgis.com/datasets?q=national+transit+map&sort_by=relevance>

<https://transitfeeds.com/>

<http://transit.land/feed-registry/>

<http://www.gtfs-data-exchange.com/> (was shut down, but there are still have some useful information for registered agencies)

GTFS data information form Google:

<https://developers.google.com/transit/gtfs/reference/>

### 5.1.2 NTD agency information collection

Programmed Python module collected more than 2,300 agencies’ information from the <https://www.nationaltransitdatabase.org/>, the information covered the following fields (screenshot of an example: Figure 4):

Physical address, websites; Service Area Detail: Service Area Sq Miles, Service Area Pop, UZA Name, Tribal Area Name, Population, Density, Square Miles;  Agency Detail:  NTD ID, Reporter Status, Reporter Type, Organization Type, FY End Date, Service Type.

Machine generated alternative text:
National Transit Database 
States 
National Statistics 
Glossary 
Help 
Birmingham-Jefferson County Transit Authority (MAX) 
Agency Information 
2121 Rev. Abraham Woods Jr. Blvd. 
5th Floor 
P.o. Box 10212 
Birmingham, AL 35203 
Web Site 
Service Area Detail 
Service Area Sq Miles: 
Service Area Pop: 
UZA Name: 
Tribal Area Name: 
Population: 
Density: 
Square Miles: 
152 
442,804 
Birmingham, 
749,495 
1,414/mi2 
530 
AL 
Agency Detail 
NTD ID: 
Reporter Status: 
Reporter Type: 
Organization Type: 
FY End Date: 
Service Types: 
40042 
Active 
Full Reporter 
Independent Public 
Agency or Authority of 
Transit Service 
9/30/2017 
Demand Response DO; 
Demand Response PT; 
aus DO; Bus PT; 

*Figure 4 NTD agency information*

### 5.1.3 GTFS data download URLs identification

Google search engine, Python, Beautiful Soup and other packages have been used for the GTFS data web scraping. After the NTD agencies’ websites data collected, the following script was used to scrape the possible GTFS download URLs:

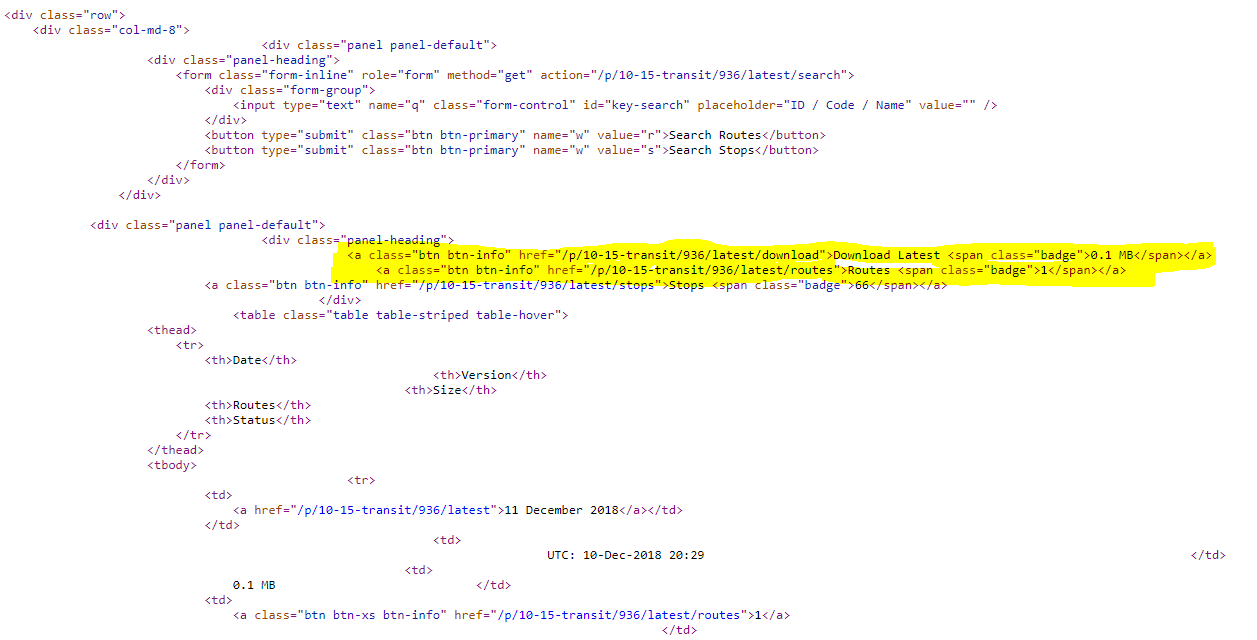
import requests  
from bs4 import BeautifulSoup  
import csv  
import re  
from datetime import datetime  
  
quotes = []  
  
with open('ntd\_agency\_website.csv') as csvfile:  
 readCSV = csv.reader(csvfile, delimiter=',')  
 for row in readCSV:  
 url1 = "https://www.google.com/search?q=site: " + row[0] + " GTFS" # change to whatever your url is  
 page = requests.get(url1)  
 soup = BeautifulSoup(page.content, 'html.parser')  
 # print(soup.prettify())  
 tb = soup.find('body', class\_='')  
 for link in tb.find\_all('div'):  
 quote = {}  
 name = link.find('a')  
 if name:  
 extractedString = re.findall(r'(https?://\S+)', str(name))  
 # extractedString = re.split('"', extractedString)  
 if extractedString:  
 # print(extractedString)  
 extractedString = re.split('&amp;', str(extractedString))  
 print(extractedString[0].replace("['", ""))  
 quote['agency\_name'] = row[0]  
 quote['GTFS\_website'] = extractedString[0].replace("['", "")  
 quotes.append(quote)  
  
res = []  
[res.append(x) for x in quotes if x not in res]  
  
datetimestr = datetime.now().strftime("%x").replace("/", "") + datetime.now().strftime("%X").replace(":", "")  
filename = 'ntd\_agency\_GTFS\_' + datetimestr + '.csv'  
with open(filename, 'wb') as f:  
 w = csv.DictWriter(f, ['agency\_name', 'GTFS\_website'])  
 w.writeheader()  
 for quote in res:  
 w.writerow(quote)  
print("Done")

***Limitation:*** Google only allow the users to do the site search with 100 websites per day per machine, otherwise must buy a API key with different plans.

There are three scenarios in the web scraping:

***Scenario 1: Static webpage (Transitfeeds website as example)***

In this scenario, the website contents are in normal HTML format, and the GTFS download links are in a table or div with clear flags (see the screenshot in Figure 5: Highlighted in yellow) in the page source:



*Figure 5 Page source codes for a static page*

***Scenario 2: Dynamic website (Transitland as example)***

Most of page contents are not shown in the page source, they are dynamically fed by js or other methods. The chrome driver, selenium and other packages are used to scrape the GTFS links:

import requests  
from bs4 import BeautifulSoup  
import csv  
import re  
from datetime import datetime  
from selenium import webdriver  
  
  
quotes = []  
  
with open('ntd\_agency\_info\_transitland\_url.csv') as csvfile:  
 readCSV = csv.reader(csvfile, delimiter=',')  
 for row in readCSV:  
 url1 = row[0] # change to whatever your url is  
 driver = webdriver.Chrome()  
 driver.get(url1)  
 page = driver.page\_source  
 soup = BeautifulSoup(page, 'html.parser')  
 # print(soup)  
 tb = soup.find('ul', class\_='list-unstyled')  
 ## print(len(tb))  
 if tb:  
 quote ={}  
 name = tb.find\_all('a')  
 if name:  
 extractedString = re.split('"', str(name))  
 print(row[0], extractedString[1])  
 if extractedString is not None:  
 #extractedString = re.split('/', str(extractedString[1]))  
 if extractedString[1]:  
 quote['agency\_name'] = row[0]  
 quote['agency\_gtfs\_url'] = extractedString[1]  
 quotes.append(quote)  
 dirver.quit()  
datetimestr = datetime.now().strftime("%x").replace("/", "") + datetime.now().strftime("%X").replace(":", "")  
filename = 'ntd\_agency\_gtfs\_transitland\_url\_' + datetimestr + '.csv'  
with open(filename, 'w') as f:  
 w = csv.DictWriter(f, ['agency\_name', 'agency\_gtfs\_url'])  
 w.writeheader()  
 for quote in quotes:  
 w.writerow(quote)  
print("Done")

***Scenario 3: Need user account credentials***

There are a few agencies require users to register to download the GTFS data, due the complex of this scenario, we didn’t collect the GTFS data from these agencies until further actions are taken if needed.

After we did the existing GTFS providers websites, we found that there are many cases are existing from simple to very complex.

*Case 1: Direct download URLs without sub-directory*

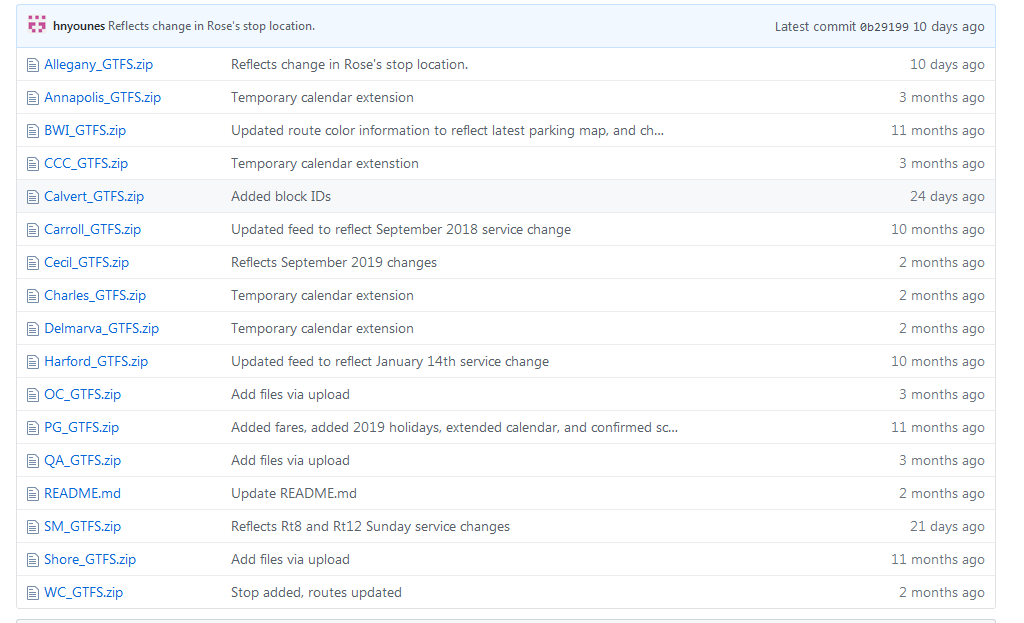
Example:

<http://www.bft.org/gtfs/google_transit.zip>

There are about 800 download URLs which can be downloaded directly.

*Case 2: Direct download URLs with sub-directory*

This case can be converted into Case 1 after the web scraping, see the screenshot (Figure 6 and Figure 7) below:



*Figure 6 Multiple download from one URL*



*Figure 7 Sub-directory download*

*Case 3: Some websites require registration (Figure 9)*



*Figure 8 Need registration to download GTFS data*

*Case 4: Some websites only provide PDF format (Figure 9)*

Machine generated alternative text:
@ https://wwwspokanetribe.com/userfiles/The_Moccasin Express_Route_2015.pdf 
The Moccasin express 
PO 50k 100 
West end Route 
A utomatic Zoom 
Leave 
Rez Rd. 
Cluster 
5:15am 
6:45am 
8:15am 
9:45am 
12:15p 
1:45pm 
3:15pm 
4:45pm 
6:15pm 
7:30pm 
509—458 _ 6549 
Wellpinit, WA. 99040 
Ford Route 
Leave 
Leave 
Arrive at 
Wellpinit 
7:30am 
10:30am 
2:30pm 
5:30pm 
7:OOpm 
8:00pm 
Connect w/ 
Bus 1 
To Spokane 
6:15am 
9:15am 
1:15pm 
4:15pm 
NEEDED 
Leave 
Wellpinit 
4:45am 
6:15am 
7:45am 
9:15am 
11:45a 
1:15pm 
2:45pm 
4:15pm 
5:45pm 
7:OOpm 
Leave 
New 
House 
Center 
5:30am 
7:OOam 
8:30am 
IO:OOa 
12:30p 
2:00 pm 
3:30pm 
5:OOpm 
6:30pm 
7:45pm 
Leave 
Leave 
Wellpinit West End 
Kokanee Boardman 
5:OOam 
6:30am 
8:OOam 
9:30am 
12:OOpm 
1:30pm 
3:00pm 
4:30pm 
6:00pm 
7:OOpm 
5:30am 
7:OOam 
8:30am 
IO:OOam 
12:30pm 
2:00 pm 
3:30pm 
5:OOpm 
6:30pm 
7:30pm 
5:40am 
7:10am 
8:40am 
10:10a 
12:40p 
2:10pm 
3:40pm 
5:10pm 
6:40pm 
7:55pm 
5:45am 
7:15am 
8:45am 
10:20a 
12:45p 
2:15pm 
3 :45pm 
5:15pm 
6:45pm 
8:00pm 
Arrive at 
Wellpinit 
7:30am 
10:30a 
2:30pm 
5:30pm 
7:OOpm 
8:15pm 
Connect 
w/ Bus 1 
Spokane 
6:15am 
9:15am 
1:15pm 
4:15pm 
* The 7pm routes from Wellpinit will ONLY run on an as needed basis. That being that there is passengers returning 
from Spokane on the last trip. 

*Figure 9 Routes and stops in PDF format*

*Other Cases: Agencies have websites with different online tools to show routes etc. but no directly download links*



*Figure 10 Agency's website with different apps*

## Evaluation

There are about 2,300 agencies having GTFS data in the USA. About 400 agencies have registered and provided GTFS download links to NTD. There are more than 600 agencies have used other transit websites (such as transitfeed, transitland etc.) to feed their GTFS data for public users.

After we downloaded all GTFS data from different sources, we found that only some agencies’ name matched the names from the NTD and with NTD\_ID. The data are from transitfeed, transitland and individual websites didn’t have NTD\_ID with them, agencies’ name is only the clue for matching the standard NTD agencies’ name. We developed a module to find the agency name similarity (machine learning package: FuzzyWuzzy), there were about 20% of them were matched the standard NTD (similarity ratio =1.00), visual checks have been conducted for the similarity ratio over 0.85, there were about 10% were fixed.

The GTFS data usually have 24 files, some agencies only provide 8 files in GTFS format. The non-matched GTFS data (large percent) with different names and make the name matching more complex and difficulty.

***Questions:***

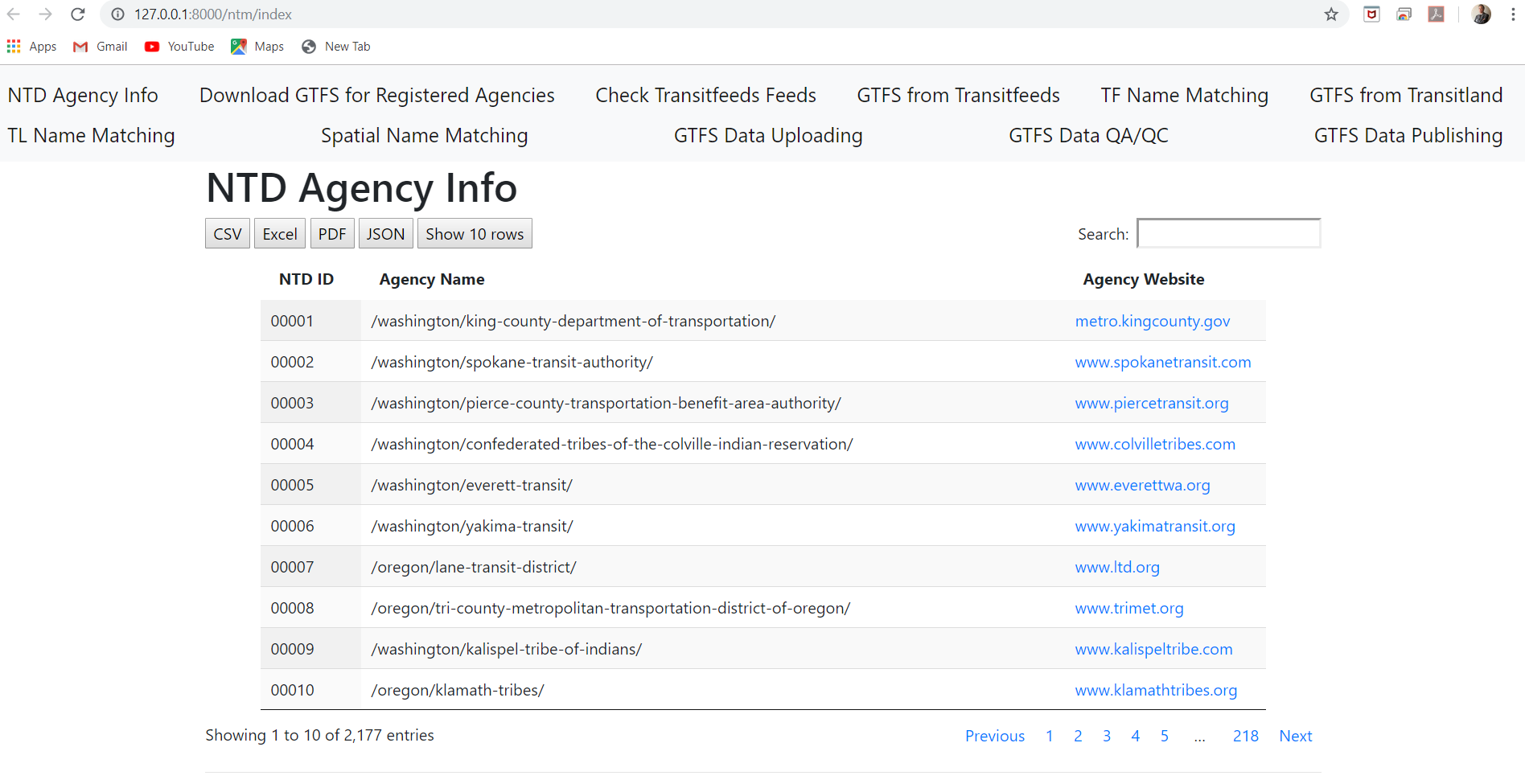
1. Where is the registered agencies’ data come from (from NTD FACE)?
2. BTS NTM shown the GTFS data are from open data, do they include the agencies from other sources (like transitfeeds, transitland etc.)? If we just considering all open data for publishing, will be much easier to handle the data and don’t need to match the NTD\_ID, just use the agencies’ as is.

***Next step:***

Since each route and stop have GIS information, we will develop a module to fix those data. The cluster analysis or CNN will be employed in Python programming, we will be used the standard NTD polygons as reference, if a classified cluster fall in known NTD area, then the NTD\_ID will be assigned to the classified cluster.

## Compilation

In order to process the GTFS data automatically, we design a web-based interface to merge all modules together and provide the developed app to the internal users to manage the GTFS data. Python, Django were used for the front end and SQLite was used for the backend, Django is a light and powerful web framework for this task. The draft of design interface (modules may change based on requests from the client) as shown in Figure 11.



*Figure 11 Draft version of interface*

***Next step:***

To obtain the SQL GTFS data schema and migrate them into SQLite DB, then develop a module to upload the zipped GTFS data into SQLite database.

## Publication

Not started yet…

# CONSIDERATIONS