

Installation and run:

1- clone repository:

```
git clone https://github.com/MJDodangeh/car_traffic_management.git
```

Then enter the car traffic management folder and open cmd

2- create and activate virtualenv:

```
py -m virtualenv venv
```

```
.\venv\Scripts\activate
```

3- install packages:

```
pip install -r requirement.txt
```

4- Initialize database:

```
python manage.py makemigrations
```

5- migrate the database:

```
python manage.py migrate
```

6- run the application:

```
python manage.py runserver
```

توضیحات پروژه:

	Methods	URL
1.	POST,GET	map/
2.	POST	tollstation/
3.	POST	owner/
4.	POST	sedan/
5.	POST	lorry/
6.	GET	cars
7.	GET	ownerscar
8.	GET	cartollarea
9.	PUT	editcarlocation/<int:carid>/
10.	POST	paytoll/
11.	GET	tolllist
12.	GET	tollviolators
13.	GET	trafficviolations

شبکه ی راه ها:

شبکه ی راه ها در برنامه به صورت گراف ساخته میشود و داده های این گراف توسط دو جدول nodes , edges , در پایگاه داده ذخیره میشوند اما برای افزایش سرعت پاسخ به درخواست ها این گراف در لحظه اجرای برنامه از پایگاه داده به memory آورده شده و به صورت ساختمان داده گرافی لیست مجاورت (adjacency list) ساخته میشود زمانی هم که تغییری روی شبکه ی راه ها اتفاق بیفتد این ساختمان داده نیز آپدیت میشود کد های این بخش برنامه درون فایل maps/apps.py درون تابع ready که در لحظه اجرای برنامه فراخوانی میشود نوشته شده است.

۱-۱- ساخت شبکه ی راه ها:

Request URL: <http://localhost:8000/map/>

Request Method: POST

نحوه ارسال اطلاعات:

```
{
  "fromnode": "a",
  "tonode": "b",
  "name": "ab",
  "longitude": 200,
  "latitude": 15
}
```

نام node مبدا
نام node مقصد
نام edge (خیابان)
طول خیابان
عرض خیابان

۱-۲- دریافت اطلاعات شبکه ی راه ها:

Request URL: <http://localhost:8000/map>

Request Method: GET

پاسخ به صورت لیست مجاورت میباشد

ایستگاه های عوارض:

ایستگاه های عوارض به صورت یک node در گراف شبکه ی راه ها ساخته میشود و داده های آن شامل nodeid آن و اطلاعات موقعیت آن در جدول tollstation ذخیره میشوند

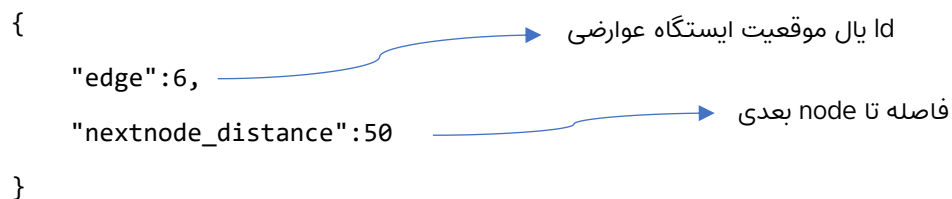
۲- ساخت ایستگاه عوارض:

Request URL: <http://localhost:8000/tollstation/>

Request Method: POST

نحوه ارسال اطلاعات:

```
{  
  "edge": 6,   
  "nextnode_distance": 50  
}
```



مالک ها:

اطلاعات مالک های خودرو شامل نام، کدملی و سن در جدول owner ذخیره میشود

۳- ایجاد یک owner در پایگاه داده:

Request URL: <http://localhost:8000/owner/>

Request Method: POST

نحوه ارسال اطلاعات:

```
{  
  "name": "sara",  
  "national_code": "0025",  
  "age": 19  
}
```

خودرو ها:

جدول car شامل نام، رنگ، نوع و موقعیت خودرو ها میباشد و موقعیت خودرو ها بدین شکل ذخیره میشود که شامل دو فیلد loc_nextnode_distance, loc_edge میباشد که loc_edge id یالی که خودرو روی آن قرار دارد میباشد و دیگری فاصله خودرو تا node بعدی در آن یال میباشد

خودرو های سواری:

اطلاعات خودرو های سواری شامل ownerid و id آن در جدول car در جدول sedan که زیر جدولی از جدول car میباشد ذخیره میشود

۴- ایجاد یک خودرو سواری در پایگاه داده:

Request URL: <http://localhost:8000/sedan/>

Request Method: POST

نحوه ارسال اطلاعات:

```
{
  "name": "samand",
  "owner": 1,
  "color": "black",
  "loc_edge": 8,
  "loc_nextnode_distance": 60
}
```

نام خودرو → "name": "samand",
Id مالک خودرو → "owner": 1,
رنگ خودرو → "color": "black",
Id یال موقعیت خودرو → "loc_edge": 8,
فاصله خودرو تا node بعدی → "loc_nextnode_distance": 60

خودرو های سنگین:

اطلاعات خودرو های سنگین شامل ownerid، وزن بار و id آن در جدول car در جدول lorry که زیر جدولی از جدول car میباشد ذخیره میشود

۵- ایجاد یک خودرو سنگین در پایگاه داده:

Request URL: <http://localhost:8000/lorry/>

Request Method: POST

نحوه ارسال اطلاعات:

```
{
  "name": "trail",
  "owner": 2,
  "color": "white",
  "loc_edge": 5,
  "loc_nextnode_distance": 130,
  "load_weight": 150
}
```

نام خودرو →
Id مالک خودرو →
رنگ خودرو →
Id یال موقعیت خودرو →
فاصله خودرو تا node بعدی →
وزن بار →

۶- دریافت اطلاعات خودروها بر اساس رنگ:

Request URL: <http://localhost:8000/cars>

Request Method: GET

<http://localhost:8000/cars?color=white&color=blue>

نحوه ارسال پارامترهای فیلتر:

رنگ های موردنظر

۷- دریافت اطلاعات خودروها بر اساس سن مالک:

Request URL: <http://localhost:8000/ownerscar>

Request Method: GET

نحوه ارسال پارامترهای فیلتر:

http://localhost:8000/ownerscar?owner_age=21&owner_age=23

سن مالک ۲۳ یا ۲۱

http://localhost:8000/ownerscar?owner_age_gte=30&owner_age_lte=40

بین ۳۰ و ۴۰

http://localhost:8000/ownerscar?owner_age_gte=70

سن مالک بزرگتر از ۷۰

http://localhost:8000/ownerscar?owner_age_lte=35

سن مالک کوچکتر از ۳۵

۸- دریافت اطلاعات خودروها بر اساس فاصله تا ایستگاه عوارض مدنظر:

این کار به این صورت انجام میشود که با الگوریتم دایکسترا کوتاه ترین مسیر تمامی خودرو ها تا عوارضی مدنظر محاسبه شده و اطلاعات آن هایی که کمتر از فاصله مدنظر بود ارسال میشوند

Request URL: <http://localhost:8000/cartollarea>

Request Method: GET

نحوه ارسال پارامترهای فیلتر:

<http://localhost:8000/cartollarea?tollstation=4&area=250>

خودروهایی که در فاصله ۲۵۰ متری از عوارضی ۴ قرار دارند

عوارض ها:

اطلاعات عوارض شامل id خودرو، id ایستگاه عوارض، مبلغ عوارض، زمان و اینکه آیا پرداخت شده است یا خیر در جدول toll ذخیره میشود عوارض در این جدول زمانی ایجاد میشود که موقعیت یک خودرو درون یالی باشد که مقصد آن یک ایستگاه عوارض است و مبلغ آن براساس نوع خودرو تعیین میگردد و زمانی که یک عوارض جدید ایجاد شد توسط Api ، paytoll باید وضعیت آن که آیا پرداخت شده یا نشده ارسال شود و اگر پرداخت نشده بود یک تخلف عوارض برای آن خودرو ثبت میگردد.

تخلفات:

دو نوع تخلف میتواند توسط خودرو ها رخ دهد. تخلف نوع اول تخلف عوارضی است که در صورت پرداخت نشدن یک عوارض توسط خودرو یک تخلف عوارضی برای آن خودرو ثبت میگردد. اطلاعات این نوع تخلف شامل id خودرو و زمان تخلف درون جدول tollviolation ثبت میگردد

تخلف نوع دوم تخلف تردد خودروهای سنگین در خیابان هایی با عرض کمتر از عرض مجاز میباشد که این تخلف زمانی ثبت میشود که موقعیت یک خودرو سنگین درون یالی با عرض کمتر از عرض مجاز باشد و اطلاعات این تخلف شامل id خودرو و زمان تخلف درون جدول trafficviolation ثبت میگردد



۹- تغییر موقعیت یک خودرو:

Request URL: <http://localhost:8000/editcarlocation/<int:carid>/>

Request Method: PUT

نحوه ارسال اطلاعات:

دریافت موقعیت جدید خودرو با id ارسال شده در متغیر carid

```
{  
  "loc_edge": 3,  id یال موقعیت جدید خودرو  
  "loc_nextnode_distance": 50  فاصله خودرو تا node بعدی  
}
```






زمانی که موقعیت یک خودرو تغییر میکند اگر موقعیت جدید آن درون یالی بود که مقصد آن یک ایستگاه عوارض است یک عوارض برای آن خودرو ثبت میگردد و همچنین اگر خودرو سنگین باشد و موقعیت جدید آن درون یالی بود که عرض آن از عرض مجاز کمتر است یک تخلف تردد برای آن خودرو ثبت میگردد

۱۰- تعیین وضعیت پرداخت یک عوارض:

Request URL: <http://localhost:8000/paytoll/>

Request Method: POST

نحوه ارسال اطلاعات:

```
{  
  "toll_id": 5,  id عوارض در جدول toll  
  "ispaid": 0  تعیین وضعیت آن عوارض: اگر 1 ارسال شود  
   یعنی آن عوارض پرداخت شده و اگر 0 ارسال  
   شود یعنی آن عوارض پرداخت نشده و یک تخلف  
   عوارض ثبت میشود  
}
```

۱۱- مشاهده لیست عوارض یک خودرو یا مالک در مدت زمان معین:

Request URL: <http://localhost:8000/tolllist>

Request Method: GET

نحوه ارسال پارامترهای فیلتر:

[http://localhost:8000/tolllist?start_time=2023-02-01 11:15&end_time=2023-02-01 11:30&carid=6](http://localhost:8000/tolllist?start_time=2023-02-01%2011:15&end_time=2023-02-01%2011:30&carid=6)

عوارض ثبت شده برای خودرو با id=6 و مدت زمان start_time تا end_time

[http://localhost:8000/tolllist?start_time=2023-02-01 11:15&end_time=2023-02-01 11:30&ownerid=4](http://localhost:8000/tolllist?start_time=2023-02-01%2011:15&end_time=2023-02-01%2011:30&ownerid=4)

عوارض ثبت شده برای خودروهای مالک با id=4 و مدت زمان start_time تا end_time

۱۲- دریافت لیست مالک هایی که تخلف عوارضی داشته اند بر اساس میزان تخلف:

Request URL: <http://localhost:8000/tollviolators>

Request Method: GET

۱۳- دریافت لیست خودروهای سنگین که در خیابان هایی با عرض کمتر از بیست متر تردد داشته اند:

Request URL: <http://localhost:8000/trafficviolations>

Request Method: GET