MyTaxi Hands On

Start the application by building and running the application:

Build the application in 'server-applicant-test-1' folder:

```
mvn clean package
```

Run the jar in 'target' folder:

```
java -jar mytaxi_server_applicant_test-0.0.1-SNAPSHOT.jar
```

Visit

```
http://127.0.0.1:8080
```

Please refer to this resource configuration that I have configured for Security :

```
    @Override
    public void configure(HttpSecurity http) throws Exception {

        http
          .authorizeRequests()
          .antMatchers(HttpMethod.POST,
"/v1/cars/manufacturer").access("hasIpAddress('127.0.0.1')")
          .antMatchers(HttpMethod.GET,
"/v1/cars/manufacturers").access("hasIpAddress('127.0.0.1')")
          .antMatchers(HttpMethod.PUT,
"/v1/drivers/{\\d+}/activate").access("hasIpAddress('127.0.0.1')")
          .antMatchers(HttpMethod.PUT,
"/v1/drivers/{\\d+}/deactivate").access("hasIpAddress('127.0.0.1')")
          .antMatchers(HttpMethod.GET, "/v1/drivers").access("hasIpAddress('127.0.0.1')")
          .antMatchers(HttpMethod.GET,
"/v1/drivers/online").access("hasIpAddress('127.0.0.1')")
          .antMatchers(HttpMethod.GET,
"/v1/drivers/offline").access("hasIpAddress('127.0.0.1')")
          .antMatchers(HttpMethod.POST, "/v1/drivers").permitAll()
          .and()
          .authorizeRequests()
          .antMatchers("/v1/**").authenticated();

    }
```

As you can see above, all the URIs that should not be accessed by Drivers (in other words are meant for ADMIN role only) can be accessed ONLY from the localhost.

Following API has a `permitAll()` which means any driver can create a driver account without any authentication.

```
POST http://127.0.0.1:8080/v1/drivers
```

Before any driver can add a car, we need to have a few car manufacturers/types that we (myTaxi) support or have explicit tie-up with. I decided that the Manufacturer should be its own Entity for the following reasons :

Use Case :

As a Driver, one of the use cases could be that I might want to get all the car manufacturers myTaxi supports or promotes before I select a Car.

Efficiency :

If the Manufacturer was just a String , we might have to fetch all the Cars in our database and filter through them. There could be a million cars in the database and this can be an expensive operation. Of course, we can use a Cache to eliminate the I/O but even then you will need to filter through a million cars with every GET request.

But if Manufacturer is its own Entity, we can get all Cars manufactured by a particular company in one I/O operation very quickly by leveraging the hash-index on the Primary Key of the Manufacturer table.

```
POST http://127.0.0.1:8080/v1/cars/manufacturer
```

Request: (engineType is NOT a required field. Default value is GAS).

```
{
  "engineType": "HYBRID",
  "manufacturer": "Tesla",
  "model": "Model S",
  "type": "LUXURY"
}
```

Response: This id will be used by our potential Drivers to select a Car.

```
{
  "id": 1,
  "manufacturer": "Tesla",
  "model": "Model S",
  "type": "LUXURY",
  "engineType": "HYBRID"
}
```

Following are the possible values for the enums used in the above request.

```
public enum CarEngineType {
      GAS, ELECTRIC, HYBRID;
}
```

```
public enum CarType
{
      CONVERTIBLE, COUPE, ELECTRIC, HATCHBACK, HYBRID, LUXURY, SEDAN, SUV, WAGON;
}
```

Our potential drivers can create a Driver account without any authentication.
* 192.168.0.101 is the IP of myTaxi server in my network :)

```
POST http://192.168.0.101:8080/v1/drivers
{
  "password": "driver09pw",
  "username": "driver09"
}
```

Important Points :

* Never store Password in Plain Text. NEVER.
* Password should never be part of the Serialized data.

Thus I have used

```
driverDO.setPassword(new BCryptPasswordEncoder().encode(driverDO.getPassword()));
```

```
 @JsonProperty(access=Access.WRITE_ONLY)
 @NotNull(message = "Password can not be null!")
 private String password;
```

Response for the above POST API to create Driver:

```
{
  "id": 9,
  "username": "driver09",
  "cars": [],
  "deleted": false
}
```

As you can see, the password is not serialized. :)

Now, as a driver and a proud owner of Tesla S, I want to add the car to my Driver account.

Before I can add a resource, I need to verify my credentials and get an access_token.

```
curl -X POST --user 'web:secret' -d
'grant_type=password&username=driver09&password=driver09pw'
http://192.168.0.101:8080/oauth/token
```

Following is the response I got

```
{
"Access_token":"a8912c59-6f8d-49e3-a01f-06f1dd5a247f",
"Token_type":"bearer",
"Refresh_token":"1b6156d3-386d-4137-a23e-51ea54da9fa6",
"Expires_in":3599,
"scope":"read write"
}
```

I need to use the above access token to select/add the car.

Request: seatCount is NOT required. The default value is 4.

```
POST
http://192.168.0.101:8080/v1/drivers/9/cars?access_token=a8912c59-6f8d-49e3-a01f-06f1dd5a247f

{
    "carManufacturerId" : 1,
    "licensePlate" : "IN KA 1234",
    "seatCount" : 2
}
```

However the response from the above API is

```
{
   "timestamp": 1519561741661,
   "status": 403,
   "error": "Forbidden",
   "exception": "com.mytaxi.exception.DriverNotOnExpectedOnlineStatusException",
   "message": "Driver does not match required online status",
   "path": "/v1/drivers/9/cars"
}
```

This is because the OnlineStatus of the Driver is Offline. We need to activate the driver account.
Only ADMIN can activate the driver account. So we need to use the localhost.

```
PUT http://127.0.0.1:8080/v1/drivers/9/activate
```

This will set the OnlineStatus of Driver to ONLINE. This can be done only by the ADMIN because we need to verify the Driver's license, Driver's driving history and other forms of background verifications.

Now try the above API again. It should give you a 201 created.
Let us verify by accessing the GET driver info API.

```
GET
http://192.168.0.101:8080/v1/drivers/9?access_token=a8912c59-6f8d-49e3-a01f-06f1dd5a247f
```

Response : (current* tells the client that this is the car that the driver is currently driving.)

```
{
   "id": 9,
   "username": "driver09",
   "cars": [
       {
           "id": 1,
           "manufacturer": {
               "id": 1,
               "manufacturer": "Tesla",
               "model": "Model S",
               "type": "LUXURY",
               "engineType": "HYBRID"
           },
           "licensePlate": "IN KA 1234",
           "seatCount": 2,
           "current": true
       }
   ],
   "deleted": false
}
```


What if, the access token has expired?

```
http://192.168.0.101:8080/v1/drivers/9?access_token=a8912c59-6f8d-49e3-a01f-06f1dd5a247f
```


```
{
    "error": "invalid_token",
    "error_description": "Invalid access token: 503f4b90-b326-4f35-aa6b-51f1a5c5ac4b"
}
```


In this case, we need to re-authenticate and fetch a new access_token.

What if I, as this driver, try to access a resource that does not belong to me ?

The resource (driverId=8) does not belong to me.

```
http://192.168.0.101:8080/v1/drivers/8?access_token=249bd36f-fa10-4733-8f3a-b7edb01f149b
```

Response :


```
{
   "timestamp": 1519570068539,
   "status": 403,
   "error": "Forbidden",
   "exception": "com.mytaxi.exception.UserNotOwnerOfResourceException",
   "message": "User is not the owner of resource.. ",
   "path": "/v1/drivers/8"
}
```


If the driver tries to select one of the many cars he owns in the database, the application will deselect all the other cars.

Example:

```
GET
http://192.168.0.101:8080/v1/drivers/8?access_token=538e4e44-e92e-4636-914a-8ec24ab64c72
```


```
{
   "id": 8,
   "username": "driver08",
   "coordinate": {
       "latitude": 55.954,
       "longitude": 9.5
   },
   "cars": [
       {
           "id": 2,
           "manufacturer": {
               "id": 2,
               "manufacturer": "Aston Martin",
               "model": "Vanquish",
               "type": "COUPE",
               "engineType": "GAS"
```

```
            },
            "licensePlate": "US NY 0007",
            "seatCount": 4,
            "current": false
        },
        {
            "id": 3,
            "manufacturer": {
                "id": 3,
                "manufacturer": "Ferrari",
                "model": "LaFerrari",
                "type": "CONVERTIBLE",
                "engineType": "GAS"
            },
            "licensePlate": "US NJ 1234",
            "seatCount": 4,
            "current": false
        },
        {
            "id": 4,
            "manufacturer": {
                "id": 1,
                "manufacturer": "Tesla",
                "model": "Model S",
                "type": "LUXURY",
                "engineType": "HYBRID"
            },
            "licensePlate": "US CA 1111",
            "seatCount": 4,
            "current": true
        }
    ],
    "deleted": false
}
```


As you can see, the “current” for “US CA 1111” is true.

```
POST
http://192.168.0.101:8080/v1/drivers/8/cars?access_token=538e4e44-e92e-4636-914a-8ec24ab
64c72
```

```
{
    "carManufacturerId" : 2,
    "licensePlate" : "US NY 0007",
    "seatCount" : 4
}
```

```
{
   "id": 8,
   "username": "driver08",
   "coordinate": {
       "latitude": 55.954,
       "longitude": 9.5
   },
   "cars": [
       {
           "id": 2,
           "manufacturer": {
               "id": 2,
               "manufacturer": "Aston Martin",
               "model": "Vanquish",
               "type": "COUPE",
               "engineType": "GAS"
           },
           "licensePlate": "US NY 0007",
           "seatCount": 4,
           "current": true
       },
       {
           "id": 3,
           "manufacturer": {
               "id": 3,
               "manufacturer": "Ferrari",
               "model": "LaFerrari",
               "type": "CONVERTIBLE",
               "engineType": "GAS"
           },
           "licensePlate": "US NJ 1234",
           "seatCount": 4,
           "current": false
       },
       {
           "id": 4,
           "manufacturer": {
               "id": 1,
               "manufacturer": "Tesla",
```

```
                "model": "Model S",
                "type": "LUXURY",
                "engineType": "HYBRID"
            },
            "licensePlate": "US CA 1111",
            "seatCount": 4,
            "current": false
        }
    ],
    "deleted": false
}
```


As you can see , the car with id=2 has current as true. This allows clients to switch between cars or add a new car using the same API.

What happens if a Driver tries to select a car that is owned by another driver ?

```
POST
http://192.168.0.101:8080/v1/drivers/6/cars?access_token=0dd55b27-e0dd-48d3-b88f-ae949f6dd4b0
{
    "carManufacturerId" : 1,
    "licensePlate" : "IN KA 1234",
    "seatCount" : 2
}
```

The driver with id=6 is trying to select a car that is owned by driver09.

Response:


```
{
    "timestamp": 1519573988796,
    "status": 400,
    "error": "Bad Request",
    "exception": "com.mytaxi.exception.CarAlreadyInUseException",
    "message": "This car is already in use ...",
    "path": "/v1/drivers/6/cars"
}
```


Additional exceptions thrown by the application :

DriverAccountDeletedException : If the account has been deleted by the driver (or by the ADMIN for bad ratings), the client will not be able to call any API for this driver.

DriverNotOwnerOfCarException : If the Driver tries to deselect a car that belongs to another driver.

Search (only allowed for ADMIN roles as we cannot expose Driver Info to the rest of the world):

```
GET http://127.0.0.1:8080/v1/drivers/online?type=LUXURY&manufacturer=Ferrari
```

Response:


```
{
  "driver08": [
    {
      "id": 3,
      "manufacturer": {
        "id": 3,
        "manufacturer": "Ferrari",
        "model": "LaFerrari",
        "type": "CONVERTIBLE",
        "engineType": "GAS"
      },
      "licensePlate": "US NJ 1234",
      "seatCount": 4,
      "current": false
    }
  ],
  "driver11": [
    {
      "id": 8,
      "manufacturer": {
        "id": 4,
        "manufacturer": "Ferrari",
        "model": "Superamerica",
        "type": "LUXURY",
        "engineType": "GAS"
      },
      "licensePlate": "BO LP 4321",
      "seatCount": 0,
      "current": false
    }
  ]
}
```


By default, the search applies the Logical AND amongst the search filters. If you want OR, then you can do the following :

```
GET
http://127.0.0.1:8080/v1/drivers/online?type=LUXURY&engineType=HYBRID&andCriteria=false
```

Notice the “andCriteria” query parameter.


```
{
  "driver10": [
    {
      "id": 7,
      "manufacturer": {
        "id": 7,
        "manufacturer": "Toyota",
        "model": "Prius",
        "type": "HYBRID",
        "engineType": "ELECTRIC"
      },
      "licensePlate": "PE CZ 9999",
      "seatCount": 0,
      "current": true
    },
    {
      "id": 6,
      "manufacturer": {
        "id": 2,
        "manufacturer": "Aston Martin",
        "model": "Vanquish",
        "type": "COUPE",
        "engineType": "GAS"
      },
      "licensePlate": "PE CZ 0001",
      "seatCount": 0,
      "current": false
    }
  ],
  "driver09": [
    {
      "id": 1,
      "manufacturer": {
        "id": 1,
        "manufacturer": "Tesla",
        "model": "Model S",
        "type": "LUXURY",
        "engineType": "HYBRID"
      },
      "licensePlate": "IN KA 1234",
```

```
      "seatCount": 2,
      "current": true
    }
  ],
  "driver08": [
    {
      "id": 2,
      "manufacturer": {
        "id": 2,
        "manufacturer": "Aston Martin",
        "model": "Vanquish",
        "type": "COUPE",
        "engineType": "GAS"
      },
      "licensePlate": "US NY 0007",
      "seatCount": 4,
      "current": true
    },
    {
      "id": 3,
      "manufacturer": {
        "id": 3,
        "manufacturer": "Ferrari",
        "model": "LaFerrari",
        "type": "CONVERTIBLE",
        "engineType": "GAS"
      },
      "licensePlate": "US NJ 1234",
      "seatCount": 4,
      "current": false
    },
    {
      "id": 4,
      "manufacturer": {
        "id": 1,
        "manufacturer": "Tesla",
        "model": "Model S",
        "type": "LUXURY",
        "engineType": "HYBRID"
      },
      "licensePlate": "US CA 1111",
      "seatCount": 4,
      "current": false
    }
  ],
```

```
  "driver02": [
    {
      "id": 5,
      "manufacturer": {
        "id": 7,
        "manufacturer": "Toyota",
        "model": "Prius",
        "type": "HYBRID",
        "engineType": "ELECTRIC"
      },
      "licensePlate": "GE HM 1111",
      "seatCount": 4,
      "current": true
    }
  ],
  "driver11": [
    {
      "id": 9,
      "manufacturer": {
        "id": 1,
        "manufacturer": "Tesla",
        "model": "Model S",
        "type": "LUXURY",
        "engineType": "HYBRID"
      },
      "licensePlate": "ME ME 1010",
      "seatCount": 0,
      "current": true
    },
    {
      "id": 8,
      "manufacturer": {
        "id": 4,
        "manufacturer": "Ferrari",
        "model": "Superamerica",
        "type": "LUXURY",
        "engineType": "GAS"
      },
      "licensePlate": "BO LP 4321",
      "seatCount": 0,
      "current": false
    }
  ]
}
```

Database Schema :

Default value of engineType is GAS.

max_passengers_allowed is provided by the ADMIN when adding carmanufacturers.

We might need this property when doing car_pool service. We do not want the Drivers to pick up as many passengers as possible since this may affect the safety of our customers. The limit can be set as per the size of the car. Currently, the service does not validate it against this max value when the driver selects seat_count.

```
CREATE TABLE carmanufacturer
  (
     id                    BIGINT GENERATED BY DEFAULT AS IDENTITY,
     date_created          TIMESTAMP NOT NULL,
     engine_type           VARCHAR(32) DEFAULT 'GAS' NOT NULL,
     manufacturer          VARCHAR(255) NOT NULL,
     max_passengers_allowed INT DEFAULT 6,
     model                 VARCHAR(255) NOT NULL,
     TYPE                  VARCHAR(255) NOT NULL,
     PRIMARY KEY (id)
  )
```

```
CREATE TABLE car
 (
                        id      BIGINT generated BY DEFAULT AS identity,
                        current boolean DEFAULT FALSE,
                        date_created timestamp NOT NULL,
                        license_plate   VARCHAR(255) NOT NULL,
                        seat_count      TINYINT(1) DEFAULT 4,
                        manufacturer_id BIGINT NOT NULL,
                        driver_id       BIGINT NOT NULL,
                        PRIMARY KEY (id)
)
```

```
CREATE TABLE driver
             (
                        id         BIGINT generated BY DEFAULT AS identity,
                        coordinate BINARY(255),
                        date_coordinate_updated timestamp,
                        date_created timestamp NOT NULL,
                        deleted       boolean NOT NULL,
                        online_status VARCHAR(255) NOT NULL,
```

```
                    password      VARCHAR(60) NOT NULL,
                    username      VARCHAR(255) NOT NULL,
                    PRIMARY KEY (id)
            )
```

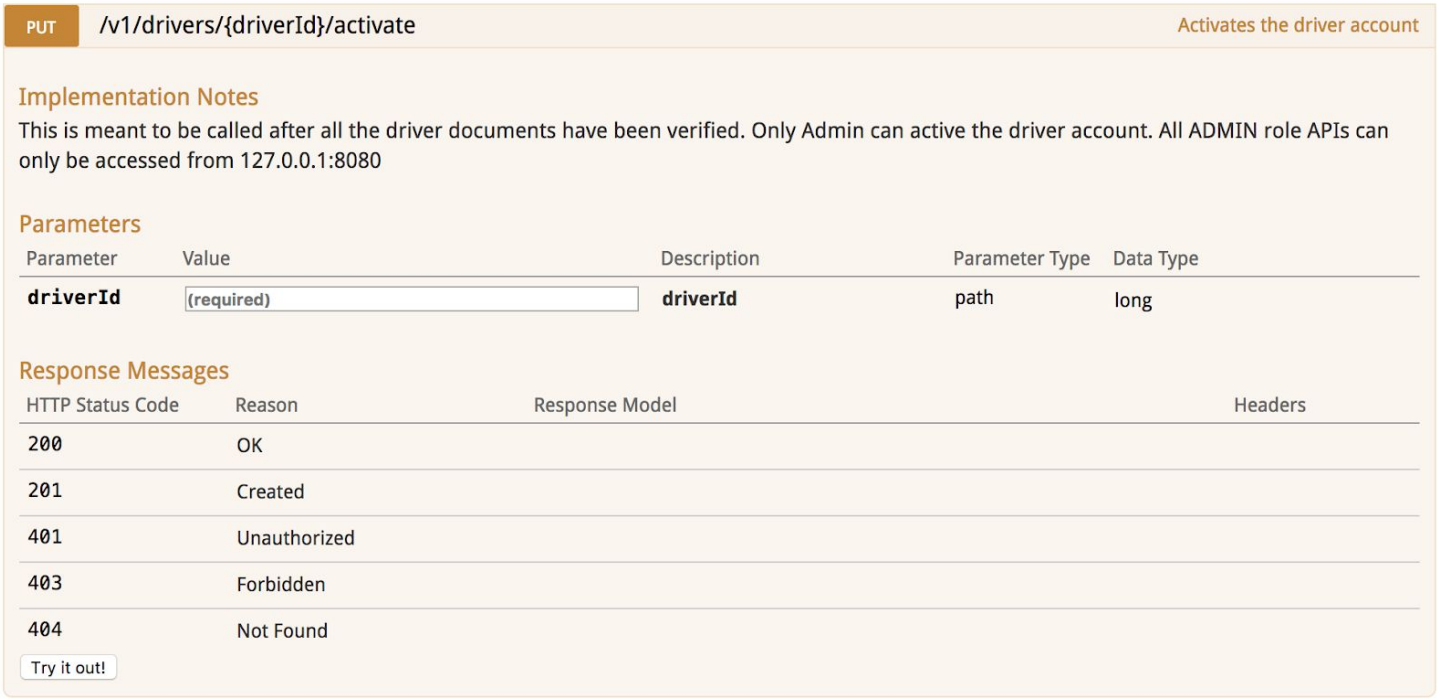
Following constraints are set on the above tables.

```
alter table car add constraint uc_car unique (license_plate)
```

```
alter table carmanufacturer add constraint uc_manufacturer unique (manufacturer, model)

alter table driver add constraint uc_username unique (username)

alter table car add constraint FK83mhba6k00fg2ctksjunys6b3 foreign key (manufacturer_id)
references carmanufacturer

alter table car add constraint FKt075681k23ii3uvdxjvvmmpm foreign key (driver_id)
references driver
```

Swagger-UI:

I have used `@ApiOperation` and `@ApiModelProperty(hidden=true)` wherever applicable so that clients can understand the API, request and response Models.



Unit Tests :

I have written unit tests for controllers using Mockito.

Scope for Improvement:

In terms of Efficiency:

```
@Override
public List<CarDO> getAllCars() {
	return Lists.newArrayList(carRepository.findAll());
}
```

The above operation is very expensive. MyTaxi, which possibly has over a million cars in its database, will require a better and more efficient code. We could use ElasticSearch that indexes documents by different search criteria like manufacturer, model, engine-type etc. Storm could be used to process the records and create documents that are then inserted into ElasticSearch.

In terms of Security:

Ideally we should have Auth tokens cached in a different server, say a Couchbase based solution.
In terms of implementation, we should have USER and ADMIN roles and use @PreAuthorize on APIs. Thus even ADMIN roles also have to go through the same authentication process as our Drivers.

For any more questions or feedback, please don't hesitate to reach me on rohan1239@gmail.com

- Rohan Doshi.