Vehicle Tracking API Documentation

APIs Flow:

1. A user Login as an admin user. (LoginAsync API)

2. The admin user registers the vehicles and gets a password for each vehicle.
3. (RegisterVehicleAsync API)

4. Navigation devices use their vehicle registration number and password to login as a navigation device. (VehicleLoginAsync API)

5. Navigation devices send vehicles location to the server. (TrackAsync API)

6. Admin users can report the current position of each vehicle. (GetVehicleCurrentLocationAsync API)

7. Admin users can report the route of each vehicle in a specific. (GetVehicleRouteAsync API).

Application’s Response Format:

```
ResponseDto{
  "success": bool,
  "result": object,
  "error": ErrorDto
}

ErrorDto{
  "statusCode":int,
  "statusType": "string",
  "errorMessage":"string"
}
```

- All APIs response by this format.
- when the success parameter sets to True, the result parameter shows the result of API
- when the success parameter sets to false, Error parameter adds additional information about Error(error number, error name, error message)

LoginAsync:

```
Controller: BackOffice
Url: api/v1/BackOffice/LoginAsync
Methos: POST

InputDto{
  "username": "string",
  "password": "string"
}

ResponseDto{
  "success": true,
  "result": {
    "accessToken": "string",
    "refreshToken": "string"
  },
  "error": null
}
```

- Use this API for users to log in.
- The access token is a JSON Web Token(JWT).
- For Authorization, add an access token to header like this: **Bearer {accessToken}"**
- for authorization in swagger, press Authorize button and add token like above format.
- Access token expires after 5 minutes.
- use refresh token to get a new token by calling RefreshToken API.

RefreshTokenAsync:

```
Controller: BackOffice
Url: api/v1/BackOffice/RefreshTokenAsync
Methos: POST

InputDto{
  "accessToken": "string",
  "refreshToken": "string"
}

ResponseDto{
  "success": true,
  "result": {
    "accessToken": "string",
    "refreshToken": "string"
  },
  "error": null
}
```

- Use this API to get a new token by expired access token and refresh token.
- refresh token changes by each call and can be used in only one refresh token request.

AddUserAsync:

```
Controller: BackOffice
Url: api/v1/BackOffice/AddUserAsync
Methos: POST

InputDto{
   "firstName": "string",
  "lastName": "string",
  "username": "string",
  "roleIdList": [
    int
  ],
  "countryCode": "string",
  "mobileNumber": "string",
  "email": "string",
  "password": "string",
  "confirmPassword": "string"
}

```

```
ResponseDto{
"success": true,
  "result": {
    "userId": 4
  },
  "error": null
}
```

- Use this API for adding new users.
- Only Admin users can access to this API.
- country code, mobile number, and email are optional.
- retrieve RoleIds from **GetRoleListAsync** API.
- The system automatically add an admin user for the first run time, you can use this user to define other users :

  username: Admin
  password: QAZwsx123

GetUserListAsync:

```
Controller: BackOffice
Url: api/v1/BackOffice/GetUserListAsync
Methos: POST

InputDto{
  "orderByDateDescending": bool,
  "limit": int,
  "offset": int
}

ResponseDto{
```

```
"success": true,
  "result": {
    "Collection<UserDto>"
  },
  "error": null
}
UserDto{
 "userId": int,
 "firstName": "string",
 "lastName": "string",
 "username": "string",
 "email": "string",
 "mobileNumber": "string"
}

```

- Use this API for getting the user list.
- Only Admin users can access to this API.
- when orderByDateDescending parameter sets to true, the user list ordered descending by creation time.
- Only Admin users can access to this API.
- all parameters are optional.
- orderByDateDescending default value: True,
- Limit default value: 10,
- Offset default value: 0

GetRoleListAsync:

```
Controller: BackOffice
Url: api/v1/BackOffice/GetRoleListAsync
```

```
Methos: GET

ResponseDto{
"success": true,
  "result": {
    "Collection<RoleDto>"
  },
  "error": null
}
RoleDto{
 "roleId": int,
 "roleName": "string"
}

```

- Use this API for getting the role list.
- Only Admin users can access to this API.

RegisterVehicleAsync:

```
Controller: BackOffice
Url: api/v1/BackOffice/RegisterVehicleAsync
Methos: POST

InputDto{
  "vehicleRegistrationNumber": "string"
}

ResponseDto{
```

```
"success": true,
  "result": {
    "vehicleRegistrationNumber": "string",
    "password": "string"
  },
  "error": null
}
```

- By this API an admin user can register a vehicle(GPS Navigator).
- Only Admin users can access to this API.
- vehicleRegistrationNumber**is a unique name for each vehicle**.
- by registering a vehicle, the system generates a password for that vehicle.
- **Vehicles(GPS Navigator) for authorizations must use their vehicleRegistrationNumber and password.**

GetVehicleNewPasswordAsync:

```
Controller: BackOffice
Url: api/v1/BackOffice/GetVehicleNewPasswordAsync
Methos: POST

InputDto{
  "vehicleRegistrationNumber": "string"
}

ResponseDto{
"success": true,
  "result": {
    "vehicleRegistrationNumber": "string",
    "password": "string"
  },
  "error": null
}
```

- By this API an admin user can request a new password for a vehicle.
- Only Admin users can access to this API.
- **After requesting a new password, the old password changed and the vehicle must use the new password for authorization.**

GetVehicleListAsync:

```
Controller: BackOffice
Url: api/v1/BackOffice/GetVehicleListAsync
Methos: POST

InputDto{
  "orderByDateDescending": bool,
  "limit": int,
  "offset": int
}

ResponseDto{
"success": true,
  "result": {
    "Collection<VehicleDto>"
  },
  "error": null
}
VehicleDto{
 "vehicleId": int,
 "vehicleRegistrationNumber": "string"
}

```

- Use this API for getting the vehicle list.
- Only Admin users can access to this API.
- when orderByDateDescending parameter sets to true, the user list ordered descending by creation time.
- Only Admin users can access to this API.
- all parameters are optional.
- orderByDateDescending default value: True,
- Limit default value: 10,
- Offset default value: 0

GetVehicleCurrentLocationAsync:

```
Controller: BackOffice
Url: api/v1/BackOffice/GetVehicleNewPasswordAsync
Methos: POST

InputDto{
  "vehicleRegistrationNumber": "string"
}

ResponseDto{
"success": true,
  "result": {
    "GetVehicleCurrentLocationAsync",
  },
  "error": null
}
LocationDto{
  "point": PointDto,
  "detail": DetailDto
}
PointDto{
  "latitude": double,
  "longitudes": double
}
DetailDto{
"addressText": "string",
      "road": "string",
      "neighbourhood": "string",
      "suburb": "string",
      "city": "string",
      "county": "Chamorshi",
      "state": "Maharashtra",
      "postcode": "string",
      "country": "India",
      "countryCode": "in"
}



```

- Use this API for retrieving the current location of a vehicle.
- Only Admin users can access to this API.
- **PointDto stores latitude and longitudes of the current location.**

- DetailDto adds Additional information about the location, this info provides by https://locationiq.com/
- LocationIq is a third party that provides location-based APIs like google, since Google APIs are not accessible in Iran, I've decided to use this their services, their API structure is almost the same to google Geocoding API.

GetVehicleRouteAsync:

```
Controller: BackOffice
Url: api/v1/BackOffice/GetVehicleRouteAsync
Methos: POST

InputDto{
    "vehicleRegistrationNumber": "string",
  "startDateTimeOffset": dateTimeOffset,
  "endDateTimeOffset": dateTimeOffset
}

ResponseDto{
"success": true,
  "result": {
    "Colection<LocationDto>",
  },
  "error": null
}
LocationDto{
  "latitude": double
  "longitudes": double,
  "dateTimeOffset": string
}

```

- Use this API for retrieving the vehicle's route during a specific time.
- Only Admin users can access to this API.

VehicleLoginAsync:

```
Controller: GpsNavigator
Url: api/v1/GpsNavigator/VehicleLoginAsync

InputDto{
  "vehicleRegistrationNumber": "string",
  "password": "string"
}

ResponseDto{
  "success": true,
  "result": {
    "accessToken": "string",
    "refreshToken": "string"
  },
  "error": null
}
```

- Use this API for vehicle login.
- The access token is a JSON Web Token(JWT).
- For Authorization, add an access token to header like this: **Bearer {accessToken}"**

- for authorization in swagger, press Authorize button and add token like above format.
- Access token expires after 5 minutes.
- use refresh token to get a new token by calling VehicleRefreshTokenAsync API.

VehicleRefreshTokenAsync:

```
Controller: GpsNavigator
Url: api/v1/GpsNavigator/VehicleRefreshTokenAsync
Methos: POST

InputDto{
  "accessToken": "string",
  "refreshToken": "string"
}

ResponseDto{
  "success": true,
  "result": {
    "accessToken": "string",
    "refreshToken": "string"
  },
  "error": null
}
```

- Use this API to get a new token by expired access token and refresh token.
- refresh token changes by each call and can be used in only one refresh token request.

TrackAsync:

```
Controller: GpsNavigator
Url: api/v1/GpsNavigator/TrackAsync
Methos: POST

InputDto{
    "latitude": double,
   "longitudes": double
}

ResponseDto{
  "success": true,
  "result": bool,
  "error": null
}
```

- Use this API in device service for send location.
- Only Vehicles can access to this API.
- The vehicleRegistrationNumber will be found from Token.