Zoomer- API

Location Microservice

PUT /location/user

Description: Add a user into the database with attributes longitude and latitude initialized as 0, the “street” attribute must be initially set as an empty string.

Body Parameters:

■ uid - a String that represents the uid in the location database

■ is_driver - the boolean value to define whether the user is a driver.

Expected Response:

■ “status”

- “OK”, if adding the user node successfully
- <string>, any other status if adding the user failed in some manner.

Example Response:

{

"status": "OK"

}

DELETE /location/user

Description: Delete a user in the database.

Body Parameters:

■ uid - a string that represents the uid in the location database

Expected Response:

■ “status”

- “OK”, if deleting the user node successfully

- <string>, any other status if deleting the user results in an error.

Example Response:

{

"status": "OK"

}

GET /location/:uid

Description: Get the current location for a certain user.

URL Parameters: uid- a String that represents the uid in the location database

Expected Response:

■ “status”

- “OK”, if the user’s location was retrieved successfully
- <string>, any other status if the user’s location was not retrieved successfully.

■ “data”: {“longitude”, ”latitude”} returns the longitude and latitude of the user if the status is “OK”.

Example Response:

{

"status": "OK",

"data": {

“longitude”: 79.3832,

“latitude”: 43.6532,

“street”: “Queens St. E”

}

}

PATCH /location/:uid

Description: Update the user's location information

URL Parameters: uid - a string that represents the uid in the location database

Body Parameters: longitude, latitude, street

Example Body:

{

"longitude": 79.0358,

"latitude": 42.0057,

"street": "Dundas St. W"

}

Expected Response:

■ "status"

- "OK", if the location was updated successfully
- <string>, any other status if the location was unable to be traced.

GET /location/nearbyDriver/:uid?radius=

Description: Get the drivers that are in a radius that the user defined from the user's current location.

URL Parameters:

■ uid - a String that represents user in the database

■ radius - Positive integer that represents the radius in km.

Expected Response:

■ "status"

- "OK", if the driver was successfully being traced
- <string>, There are no nearby drivers or errors occurred.

■ “data”

- List of uid for driver and their location

Example Response:


```
{
    "data": {
        “1”: {“longitude”: 56.235, “latitude”: 70.368,
        “street”: “Mississauga Rd.”},
        “2”: {“longitude”: 56.255, “latitude”: 70.168,
        “street”: “Burnhamthorpe Rd. W”}
    }
    "status": "OK"
}
```


PUT /location/road

Description: Add a road into the database. If the road name already exists in the database, update the rest of the info in the database with the road name

Body Parameters:

■ roadName- The name of the road

■ hasTraffic - A boolean value to define whether the road is on heavy traffic

Expected Response:

■ “status”

- “OK”, if the road is successfully added/updated
- <string>, if an error happened when adding / modifying the road information.

Example Response:


```
{
```

"status": "OK"

}

POST /location/hasRoute

Description: Create a connection from a road to another; making a relationship in Neo4j.

Body Parameters:

■ roadName1 - The name of the road

■ roadName2 - The road that can be reached from roadName1

■ hasTraffic - A boolean value to define whether the route is on heavy traffic

■ time - positive integer representing the time to travel from roadName1 to roadName2 in minutes

Expected Response:

● “status”

○ “OK”, if the road is successfully added

○ <string>, if an error happened when adding the route information.

Example Response:

{

"status": "OK"

}

DELETE /location/route

Description: Disconnect a road with another; remove the relationship in Neo4j.

Body Parameters:

■ roadName1 - The name of the road

■ roadName2 - The road that can be reached from RoadName1

Expected Response:

● “status”

○ “OK”, if the road is successfully added

○ <string>, if an error happened when adding / modifying the route information.

Example Response:

```
{
    "status": "OK"
}
```

GET /location/navigation/:driver?passengerUid=

Description: Get the navigation from the current driver’s road to the passenger’s road with minimal time.

URL Parameters:

■ driverUid - the driver’s uid

■ passengerUid - the passenger’s uid

Expected Response:

■ “status”

● “OK” - if it gets the time successfully

● <string> - if there is an error.

■ “data”

● A list of json which contains the routing of the drive, each step’s time and the total time of the trip with traffic information of the road.

Example Response:

```
{
    “status”: “OK”,
```

“data”:{“total_time”: 43,

“route”: [{“street”: “Bloor St W”, “time”: 0,

“is_traffic”: false},{“street”: “Spadina Crescent”, “time”: 3,

“is_traffic”: true}, {“street”: “Lake Shore Blvd W”, “time”: 7,

“is_traffic”: false}, {“street”: “Gardiner Expy”, “time”: 2,

“is_traffic”: false}, {“street”: “Mississauga Rd E”, “time”:

22, “is_traffic”: true}, {“street”: “Dundas St W”, “time”: 5,

“is_traffic”: true}, {“street”: “Mississauga Rd W”, “time”: 1,

“is_traffic”: true}, {“street”: “Outer Circle Rd”, “time”: 3,

“is_traffic”: true}]

}

}

TripInfo Microservice

POST /trip/request

Description: Send a request for the trip that a passenger requests and return the status.

Body Parameters:

■ uid - the passenger’s UID.

■ radius - Positive integer that represents the radius in km.

Expected Response:

■ “status”

- “OK” if the request is being made (there is a driver nearby)
- <string>, any other status if the request fails or there is no driver in the radius.

■ “data”

- A list of driver’s uid. (e.g. [“1”,”2”,”3”,”4”,”5”])

POST /trip/confirm

Description: Adding trip info into the database

Body Parameters:

■ driver - driver's uid string

■ passenger - Passenger's uid string

■ startTime - The trip start time, counting when the driver accepts the trip, in unix timestamp format

Expected Response:

■ "status"

- "OK", if trip is created and added correctly to the database
- <string>, or any other status if the trip was unable to be created.

■ "data" - an object

- Includes the _id that created by the MongoDB

PATCH /trip/:_id

Description: Adding extra information when the trip is done.

URL Parameters: _id - the trip id for the trip.

Body Parameters:

■ distance - The distance of the trip.

■ endTime - The end time of the trip, in unix timestamp format.

■ timeElapsed - Total time of the trip.

■ discount - the discount for the trip, if there is no discount, the value should be 0.

■ totalCost - The total cost before the discount

■ driverPayout - the amount earned by the driver in the trip, The driver earns 65% of the cost before the discount.

Expected Response:

■ “status”

- “OK”, if the trip was updated
- <string>, any other status if the trip cannot be updated.

GET /trip/passenger/:uid

Description: Get all the trips that the certain passenger has.

URL Parameters:

■ uid- a String that represents a user in the database

Expected Response:

■ “status”

- “OK”, if the user exists.
- <string>, any other status if the user does not exist in the database or other errors.

■ “data”

- trips - Includes a list of detailed trip info

Example Response:


```
{
    "data": {
        “trips”:[{‘_id’:1, ‘distance’: 7, ‘totalCost’:14.59,
        ‘discount’:0, ‘startTime’:1615855049, ‘endTime’: 1615855949,
        ‘timeElapsed’: ‘00:15:00’, ‘driver’: ‘acbd’}]
        }
    "status": "OK"
}
```

GET /trip/driver/:uid

Description: Get all the trips that the certain driver has.

URL Parameters:

■ uid - a String that represents the user in the database

Expected Response:

■ “status”

- “OK”, if the driver is able to trace the history
- <string>, any other status if the trace is failed.

■ “data” - The trips info that the curtain driver has

Example Response:

```
{
    "data": {
        “trips”:[{‘_id’:1, ‘distance’: 7, ‘driverPayout’:7.36,
        ‘startTime’:1615855049, ‘endTime’: 1615855949, ‘timeElapsed’:
        ‘00:15:00’, ‘passenger’: ‘cdad’}]
    }
    "status": "OK"
}
```

GET /trip/driverTime/:_id

Description: Get the estimated time it will take for the driver to arrive at the passenger. This time is obtained from the navigation endpoint in location microservice.

URL Parameters:

■ _id - A string stands for the trip.

Expected Response:

■ “status”

- “OK”, if the driver’s distance successfully gets from the database
- <string>, any other status if an error occurred.

■ “data”

- “arrival_time” - positive integer representing the estimated time it will take for the driver to arrive to the passenger

Example Response:


```
{
    "data": {
        "arrival_time": 2
    }
    "status": "OK"
}
```


User Microservice

GET /user/:uid

Description: Get user’s basic information

URL Parameters:

■ uid - a String that represents the user in the database

Expected Response:

■ “status”

- “OK”, if the user’s info is being successfully being retrieved
- <string>, any other status if the user was not able to get traced.

■ “data”

- name
- email

- rides
- isDriver
- availableCoupons
- redeemedCoupons

POST /user/register

Description: Register a user into the system.

Body Parameters:

■ name - The name of the user

■ email - The email of the user

■ password - The password that is set by the user

Expected Response:

■ “status”

- “OK”, if the user was successfully registered
- <string>, any other status if the User was unable to be registered.

POST /user/login

Description: login a user into the system

Body Parameters:

■ email - The email of the user

■ password - The password that is set by the user

Expected Response:

■ “status”

- “OK”, if the user was successfully logged in
- <string>, any other status if the User was unable to be logged in.

PATCH /user/:uid

Description: Editing part of user's information

URL Parameters:

■ Uid - a String that represents a user in the database

Body Parameters: At least one of email, password, rides, is_driver, availableCoupons or redeemedCoupons

Expected Response:

■ "status"

- "OK", if the User is modified based on the provided parameters
- <string>, any other status if the User was not able to be modified.