EcoPRT REST APIs

All success REST calls will return a 200 status code. If they are not a success, it will return a 400 or 500 status code

Admin

All POST admin functions return a success or error message

POST

URI: /addNode
Method: POST
Description: Adds a node to the database. If it's an already existing name, it will reject it
Form Inputs:

Field	Type	Description
name	string	Human-readable name to identify the node (unique)
location	[float, float]	Geographic coordinates of the node, in decimal form (latitude, longitude)
type	Number	What type of node this is: ○ 0: pickup station ○ 1: charging station ○ 2: docking station ○ 3: not a station/just a node

Form Output: Success Message if worked, Error message if not

URI: /addVehicle
Method: POST
Description: Adds a vehicle to the database. If a vehicle name already exists, it will reject a new vehicle with that name
Form Inputs:

Field	Type	Description
name	string	Human-readable name to identify the vehicle (unique)

Form Output: Success Message if worked, Error message if not

URI: /addEdge
Method: POST
Description: Adds an edge to the database.
Form Inputs:

Field	Type	Description

startingNode	ObjectID	Starting node of the edge
endingNode	ObjectID	Ending node of the edge
distance	Number	Length of the edge between the two nodes Represented as miles
waypoints	Array of waypoint items defined in schema	

Form Output: Success Message if worked, Error message if not

URI: /addRide
Method: POST
Description: Adds a ride to the database.
Form Inputs:

Field	**Type**	**Description**
numberOfPassengers	Number	Number of passengers in the ride
pathChosen	Array of Edges	The path that the vehicle will follow for the ride
passengerLocation	Array of Number	Location of passenger when they request a ride Latitude and Longitude coordinates
distance	Number	Length of the entire ride (in miles)
vehicleID	ObjectID	Vehicle that will carry out the ride
userID	ObjectID	User that requested the ride
startingNode	ObjectID	Where the user will get picked up at
endingNode	ObjectID	Where the user will get dropped off at
currentTask	Number	Current task of 0 (which means it's been created)
requestTime (optional)	*Date*	*TIme the ride was requested by the user*
carArrivalTime (optional)	*Date*	*Time the vehicle arrives at the starting node*
pickupTime (optional)	*Date*	*Time the user gets into the vehicle and starts the ride*
dropOffTime (optional)	*Date*	*Time the user will arrive at destination*

Form Output: Success Message if worked, Error message if not

Sending commands to cars

URI: /wait
Method: POST
Description: Makes the car idle and has it waiting for a new command
Form Inputs:

Field	Type	Description
vehicleID	ObjectID	ID of the vehicle that the command will go to
~~command~~	~~Number~~	~~Number that corresponds to wait that will stop the car~~

Form Output: Success Message if successful, Error message if not.

URI: /drive
Method: POST
Description: Makes the able to drive and follow a certain path given
Form Inputs:

Field	Type	Description
vehicleID	ObjectID	Name of the vehicle that the command will go to
~~command~~	~~Number~~	~~Number that corresponds to drive that will let the vehicle continuing doing what it was doing~~
path	Array of Edges	The path the vehicle will drive once it's able to

Form Output: Success Message if successful, Error message if not.

GET

- Add queries
  - Like date ranges
  - Specific items
  - Parameters to specify get results

URI: /getNodes
Method: GET
Description: Returns a single node if a name or ID is given. Otherwise if no parameters are given, all the nodes in the database are returned
Form Inputs:

Field	Type	Description
name	String	Name of the node (optional)
nodeID	ObjectID	ID of the node (optional)

Form Output: Returns the specified node if the given parameter is given. Otherwise, an array of nodes is returned.

Returned in json format with the following field from the database for each node:

- Name of node

URI: /getVehicles
Method: GET
Description: Returns a single vehicle if a name or ID is given. Otherwise, if no parameters are given, all cars from the database is returned
Form Inputs:

Field	Type	Description
name	String	Name of the vehicle (optional)
vehicleID	ObjectID	ID of the vehicle (optional)

Form Output: Returns the specified vehicle from the database if given the parameter. Otherwise, an array of vehicles are returned.
Returned in json format with the following fields for each vehicle:

- Name of vehicle
- ID of vehicle

URI: /getEdges
Method: GET
Description: Returns a single Edges if a starting node and ending node given. You can also use an edgeID to get a specific Edge. Otherwise, if no parameters are given, it will return all edges from the database
Form Inputs:

Field	Type	Description
startingNode	ObjectID	First/Starting node for the edge (optional)
endingNode	ObjectID	Second/Ending node for the edge (optional)
edgeID	ObjectID	ID of the edge (optional)

Form Output: Returns the specified edge if given the parameters. Otherwise, an array of all edges will be given.
Returned in json format with the following fields for each node:

- startingNode:
    - Name of node
- endingNode
    - Name of node

URI: /getRides
Method: GET
Description: Returns a single ride if the ID is given. Can return a group of rides if you give a start date and end date that will return all rides for that range. You can also return a group of rides depending if the rides were completed or cancelled. Otherwise, if no parameters are given, all rides are returned.
Form Inputs:

Field	Type	Description

rideID	ObjectID	ID of the ride (optional)
startDate	Date	Start data for a date range (optional)
endDate	Date	End date for the date range (optional)
currentTask	Number	This will specify is the ride was completed or cancelled. 3 - completed 4 - cancelled (optional)

Form Output: Returns the specified edge if given an ID and nothing else. Returns array of rides for a specified date range if that is given. Returns array of rides of completed or cancelled if you use that parameter. Otherwise, an array of all rides are returned.
Returned in json format with the following fields from the database for each ride:

- rideID
- vehicleID
- userID
- numberOfPassengers
- requestTime
- carArrivalTime
- pickupTime
- dropOffTime
- currentTask
- pathChosen
    - For each edge
        - startingNode
            - Name of node
        - endingNode
            - Name of node

DELETE

- Maybe add parameters to specify deletion of certain items

URI: /deleteNode
Method: DELETE
Description: Deletes a single node based on the name or ID
Form Inputs:

Field	**Type**	**Description**
name	String	Name of the node
nodeID	ObjectID	ID of the node

Form Output: If successful, will return a success message and node will be removed from list. Error message if it didn't work.

URI: /deleteVehicle
Method: DELETE
Description: Deletes a single vehicle based on the name or ID

Form Inputs:

Field	Type	Description
name	String	Name of the vehicle
nodeID	ObjectID	ID of the vehicle

Form Output: If successful, will return a success message and vehicle will be removed from list. Error message if it didn't work.

URI: /deleteEdge
Method: DELETE
Description: Deletes an edge based on the ID or the starting and ending node
Form Inputs:

Field	Type	Description
edgeID	ObjectID	ID of the edge
startingNode	ObjectID	First node of the edge
endingNode	ObjectID	Last node of the edge

Form Output: If successful, will return a success message and edge will be removed from list. Error message if it didn't work.

URI: /deleteRide
Method: DELETE
Description: Deletes a single ride based on the ID. You can give a date range to delete multiple rides. You can also delete rides based on if they were completed or cancelled.
Form Inputs:

Field	Type	Description
rideID	ObjectID	ID of the ride
startDate	Date	Start data for a date range
endDate	Date	End date for the date range
currentTask	Number	This will specify is the ride was completed or cancelled. 3 - completed 4 - cancelled

Form Output: If successful, will return a success message and ride or rides will be removed from list. Error message if it didn't work.

User

POST

URI: /addUserRide
Method: POST

Description: Adds a user ride to the database.
Form Inputs:

Field	Type	Description
numberOfPassengers	Number	Number of passengers in the ride
pathChosen	Array of Edges	The path that the vehicle will follow for the ride
passengerLocation	Array of Number	Location of passenger when they request a ride Latitude and Longitude coordinates
distance	Number	Length of the entire ride (in miles)
vehicleID	ObjectID	Vehicle that will carry out the ride
userID	ObjectID	User that requested the ride
startingNode	ObjectID	Where the user will get picked up at
endingNode	ObjectID	Where the user will get dropped off at
currentTask	Number	Current task of 0 (which means it's been created)
requestTime (optional)	*Date*	*TIme the ride was requested by the user*
carArrivalTime (optional)	*Date*	*Time the vehicle arrives at the starting node*
pickupTime (optional)	*Date*	*Time the user gets into the vehicle and starts the ride*
dropOffTime (optional)	*Date*	*Time the user will arrive at destination*

Form Output:Success Message if worked, Error message if not

GET

URI: /getUserRides
Method: GET
Description: Returns a single ride if the ID is given. Can return a group of rides if you give a start date and end date that will return all rides for that range. You can also return a group of rides depending if the rides were completed or cancelled. Otherwise, if no parameters are given, all rides are returned.
Form Inputs:

Field	Type	Description
rideID	ObjectID	ID of the ride (optional)

startDate	Date	Start data for a date range (optional)
endDate	Date	End date for the date range (optional)
currentTask	Number	This will specify is the ride was completed or cancelled. 3 - completed 4 - cancelled (optional)

Form Output: Returns the specified edge if given an ID and nothing else. Returns array of rides for a specified date range if that is given. Returns array of rides of completed or cancelled if you use that parameter. Otherwise, an array of all rides are returned.
Returned in json format with the following fields from the database for each ride:

- rideID
- vehicleID
- userID
- numberOfPassengers
- requestTime
- carArrivalTime
- pickupTime
- dropOffTime
- currentTask
- pathChosen
    - For each edge
        - startingNode
            - Name of node
        - endingNode
            - Name of node