Brent Nolan
Cloud Only

Summary:
For all requests, there needs to be an authentication sent. Depending on the account different actions will be possible or prohibited. This is described below for each request.

POST /account
This will create a new account entity, assign it the attributes specified in the body, auto-generate an id for the account. There are no required fields.

Example body: {"name":"User 1", "address":"111 User Lane", "phone_number":"111-111-1111","d_license":"A1111", "dl_state":"Alabama"}

GET /account
This is available to the admin accounts only. This will return a JSON list of accounts.

GET /account/{account id}
This will return a JSON object for the account corresponding to that account id. Only admin accounts, and the user that is tied to that account (and supplies the authorization) can get and view the account information.

PATCH/account/{account id}
This will modify the account based on the attributes specified in the body. However, the id cannot be updated, and users will get a 403 error if they attempt to update the id. Again, only admin accounts, and the user that is tied to that account (and supplies the authorization) can update the account information.

Example body: {"name":"User 1 - Updated", "address":"111 User Road - Updated"}
This example will modify the account entity corresponding to the {account id} by changing the name to User 1 – Updated, and the address to 111 User Road - Updated.

PUT/account/{account id}
This will modify all valid account attributes. But, since there is a Patch function, the user must provide all values in the PUT request body. Again, only admin accounts, and the user that is tied to that account (and supplies the authorization) can update the account information.

Example body: {"name":"User 1", "address":"111 User Lane", "phone_number":"111-111-1111","d_license":"A1111", "dl_state":"Alabama"}

DELETE/account/{account id}
This will delete the corresponding account. It will first check to see if the account is currently renting a car, and if it is it will remove that account from the car, clearing the start_date and end_date fields, and then will delete the account entity.

POST/car
This will create a new car entity. It will assign the car attributes based on the body, and it will auto-generate an id and current_acount, start_date, and end_date to None. Only admin accounts can create

new cars.

Example body: {"make":"Make 1", "model":"Model 1", "color":"Black", "v_license":"VA1111", "v_state":"Alabama", "year":2001}

GET /car
This will return a JSON list of cars. Every user can do this

GET /car/{car id}
This will return a JSON object for the car corresponding to that car id. Every user can do this

PATCH/car/{car id}
This will modify the car based on the body. Only admin accounts can edit cars.

Example body: {"make":"Make 1 - Updated", "model":"Model 1 - Updated"}
This will modify the car with the corresponding id to the updated values.

DELETE/car/{car id}
This will delete the corresponding car. Only admin accounts can delete cars.

PUT/car/{car id}/rental
This will update the corresponding car's arrival_date and current_account to the values provided in the body. If there is already a account in the car it will return a 403 forbidden error.

{"account_id":"{{Account id}}", "a_date":"1/1/2001"}

This will place the id for Account into the current_account field and will put the string “1/1/2001” into the arrival date field for the car id that is in the url.

PATCH/car/{car id}/rental
This will update the corresponding car's rental information to values provided in the body.

{"end_date":"2/22/2011"}
This will change the end date of the car id provided in the url to “2/22/2011”

GET/rental/(.*)
This will get the car that is being rented by the account id provided in the URL. Only the account who rented it, or an admin can access this.