

SOFTWARE ENGINEERING (CEN5035)

RideShare

Backend API documentation

Developers:

1. Kalyan GVS
2. Dheeraj Kumar M

API's Developed:

- 1) signup
- 2) login
- 3) user
- 4) delete
- 5) PostaRide
- 6) SearchaRide

1) signup :

Description:

- Method: POST
- This API allows the new users to register on the application.
- Once the user is registered, they can login into the website whenever, using their email address and password.

Acceptance Criteria:

- For registering, the following fields are required to be entered by the user: *Name, Email, Password, Gender, City, DateofBirth, PhoneNumber*.
- If any/all of these fields weren't entered, the API throws an error message.
- Once all the inputs are properly entered by the user, all these details are updated to the *users* table in the database.
- If an user tries registering again using the same email, the API throws an error message: "*User already registered*"

Sample Requests and Responses:

The screenshot displays a REST client interface with a POST request to `localhost:8080/signup`. The request body is a JSON object containing user registration details. The response is a JSON object indicating a successful registration with a new user ID and timestamp.

Request:

```
POST localhost:8080/signup

{
  "email": "hell@gmail.com",
  "password": "hello123",
  "gender": "male",
  "city": "tampa",
  "dob": "12/12/2012",
  "name": "kalyan",
  "phonenumber": "352-352-352"
}
```

Response:

```
{
  "Value": {
    "ID": 12,
    "CreatedAt": "2022-03-03T23:42:14.625449-05:00",
    "UpdatedAt": "2022-03-03T23:42:14.625449-05:00",
    "DeletedAt": null,
    "Name": "kalyan",
    "Password": "$2a$10$D/QSnm9RGirQUEa0zfAKWugc1sFiWXxAIPgydx3a0.7z8XI.iIIne",
    "Email": "hell@gmail.com",
    "DOB": "12/12/2012",
    "Gender": "male",
    "PhoneNumber": "352-352-352",
    "City": "tampa"
  },
  "Error": null,
  "RowsAffected": 1
}
```

2) login:

Description:

- Method: POST.
- This API allows the already existing users to login to the application.
- Valid credentials must be provided for a successful login.
- Logging in is mandatory for any user to use the functionalities of application like *postaride* and *searchrides*.

Acceptance Criteria:

- For registering, the following fields are required to be entered by the user: *Name, Email, Phone and Password*.
- If any/all of these fields weren't entered, the API throws an error message.
- Once all the inputs are properly entered by the user, the API throws a confirmation message: "*User added to the database*"
- If the same user tries registering again using the same credentials, the API throws the message: "*User already exists*"
- Similarly for login, the API should throw an error message, if the user entered either a wrong email or password.

Sample Requests and Responses:

The screenshot displays a REST client interface with a POST request to `localhost:8080/login`. The request body is a JSON object with the following structure:

```
1 {
2   "email": "hello@gmail.com",
3   "password": "hello123"
4 }
```

The response status is 200 OK, with a time of 403 ms and a size of 792 B. The response body is a JSON object with the following structure:

```
1 {
2   "message": "logged in",
3   "status": false,
4   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NDY0NTUzNDQsImZcyI6IjYifQ.f_B1m1kR8URdn7bRRHg4gHz9I-Egx955ujpdN0JMiz0",
5   "user": {
6     "ID": 6,
7     "CreatedAt": "2022-03-02T13:18:24.70836-05:00",
8     "UpdatedAt": "2022-03-04T10:19:21.641591-05:00",
9     "DeletedAt": null,
10    "Name": "kalyan",
11    "Password": "$2a$10$f37q8yW36RSkGhHgpWpmbewbLGFw4Z87mGnBiDz3DPVZtbnwoVPru",
12    "Email": "hello@gmail.com",
13    "DOB": "12/12/2012",
14    "Gender": "mae",
15    "PhoneNumber": "352-352-352",
16    "City": "tampa"
17  }
18 }
```

3)user:

Description:

- Method: GET
- Once the user is logged in successfully an active JSONWebToken is generated. This API retrieves the user details such as *Name*, *Email*, *Password(encrypted format)*, *Gender*, *City*, *DateofBirth*, *PhoneNumber* using the web token.

Sample Requests and Responses:

The screenshot displays a REST client interface with a request and response view. The request is a GET to `localhost:8080/auth/user`. The response is a JSON object containing user details.

Request:

- Method: GET
- URL: `localhost:8080/auth/user`
- Params: x-access-token (Value: `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJvc2VyS...`)

Response:

```
1  {
2    "ID": 6,
3    "CreatedAt": "2022-03-02T13:18:24.70836-05:00",
4    "UpdatedAt": "2022-03-04T10:19:21.641591-05:00",
5    "DeletedAt": null,
6    "Name": "kalyan",
7    "Password": "$2a$10$f37q8yW36RSkGhHgpWpmbewBLGFw4Z87mGnBiDz3DPVZtbnwoVPru",
8    "Email": "hello@gmail.com",
9    "DOB": "12/12/2012",
10   "Gender": "mae",
11   "PhoneNumber": "352-352-352",
12   "City": "tampa"
13 }
```

4)delete:

Description:

- Method: Delete
- This API is used to delete all the data of the existing user from the database records.
- This action is performed using JSONWebToken.

Sample Requests and Responses:

The screenshot displays a REST client interface with a DELETE request to `localhost:8080/delete`. The interface includes tabs for Params, Authorization, Headers (6), Body, Pre-request Script, Tests, and Settings. The Authorization tab is active, showing a dropdown for 'Inherit auth...' and a note: 'The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)'. Below this, a message states: 'This request is using No Auth from collection [Rideshare](#).' The bottom section shows the response body with a 'Pretty' tab selected, displaying a JSON array of two objects. The status bar at the bottom indicates a 200 OK status, 306 ms time, and 262 B size.

DELETE `localhost:8080/delete` **Send**

Params **Authorization** Headers (6) Body Pre-request Script Tests Settings

Type **Inherit auth...**

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

This request is using No Auth from collection [Rideshare](#).

Body Cookies Headers (4) Test Results **Status: 200 OK** Time: 306 ms Size: 262 B [Save Response](#)

Pretty Raw Preview Visualize **Text**

```
1 {"message": "logged out success"}
2 {"message": "http: named cookie not present"}
3
```

5)postaride:

Description:

- Method: POST
- This API is used to “post a ride” details by the users.
- Users must be logged in to use this PostARide functionality.

Acceptance Criteria:

- For Posting a ride the following fields must be provided by the user: *name, startLocation, fromCity, toCity, destLocation, priceperSeat, duration, seatsAvailable, petsAllowed, carModel, carType, addlNotes, phoneNumber*.
- Once all the inputs are properly entered by the user, the API throws a confirmation message: *“Ride has been Successfully Posted.”*

Sample Requests and Responses:

The screenshot displays a REST client interface with a POST request to `localhost:8081/postaride`. The request body is a JSON object containing ride details. The response is a 200 OK status with a message indicating the ride was successfully posted.

Request:

```
POST localhost:8081/postaride
```

Body (JSON):

```
{
  "Name": "kalyan",
  "StartLocation": "Gainesville",
  "FromCity": "Gainesville",
  "ToCity": "Ocla",
  "DestLocation": "Starbucks",
  "Price": "40",
  "StartTime": "2022-02-01 08:10:50",
  "EndTime": "2022-02-05 08:10:22",
  "RideDuration": "1 hour",
  "SeatsAvailable": 4,
  "PetsAllowed": "No",
  "carModel": "Toyota",
  "carType": "SUV",
  "AddlNotes": "Please be present on time",
  "PhoneNumber": "352-352-3523"
}
```

Response:

```
{
  "message": "Ride has been successfully posted"
}
```

Status: 200 OK Time: 35 ms Size: 165 B

6)searchrides:

Description:

- Method: POST
- Using this API, users can get the available rides.
- To search for available rides *fromCity* , *toCity* and *startTime* must be provided.

Acceptance Criteria:

- The API should throw an error message whenever any of the above mentioned required parameters aren't specified.
- Whenever a request is sent the API should send a response with the list of available rides in the specified routes with details : *name*, *startLocation*, *fromCity*, *toCity*, *destLocation*, *priceperSeat*, *duration*, *seatsAvailable*, *petsAllowed*, *carModel*, *carType*, *addlNotes*, *phoneNumber*.

Sample Requests and Responses:

The screenshot shows a REST client interface with a POST request to `localhost:8080/searchrides`. The request body is a JSON object with the following fields:

```
{  "FromCity": "Gainesville",  "ToCity": "Ocla",  "StartTime": "2022-02-03 08:10:50"}
```

The response status is 200 OK. The response body is a JSON array of ride objects:

```
[  {    "ID": 85,    "CreatedAt": "2022-03-04T16:50:15.048376-05:00",    "UpdatedAt": "2022-03-04T16:50:15.048376-05:00",    "DeletedAt": null,    "Name": "kalyan",    "startLocation": "Gainesville",    "fromCity": "Gainesville",    "toCity": "Ocla",    "destLocation": "Starbucks",    "priceperseat": 0,    "Duration": 0,    "seatsAvailable": 4,    "petsAllowed": "No",    "carModel": "Toyota",    "carType": "SUV",    "addlNotes": "Please be present on time",    "PhoneNumber": "352-352-3523",    "UserId": 6,    "ToStartTime": "2022-03-04T03:10:50-05:00",    "ToEndTime": "2022-02-05T03:10:22-05:00"},    {    "ID": 84,    "CreatedAt": "2022-03-04T16:50:12.616449-05:00",    "UpdatedAt": "2022-03-04T16:50:12.616449-05:00",    "DeletedAt": null,    "Name": "kalyan",    "startLocation": "Gainesville",    "fromCity": "Gainesville",    "toCity": "Ocla",    "destLocation": "Starbucks",    "priceperseat": 0,    "Duration": 0,    "seatsAvailable": 4,    "petsAllowed": "No",    "carModel": "Toyota",    "carType": "SUV",    "addlNotes": "Please be present on time",    "PhoneNumber": "352-352-3523",    "UserId": 6,    "ToStartTime": "2022-02-03T03:10:50-05:00",    "ToEndTime": "2022-02-05T03:10:22-05:00"},    {    "ID": 81,    "CreatedAt": "2022-03-04T16:35:28.067715-05:00",    "UpdatedAt": "2022-03-04T16:35:28.067715-05:00",    "DeletedAt": null,    "Name": "kalyan",    "startLocation": "Gainesville",    "fromCity": "Gainesville",    "toCity": "Ocla",    "destLocation": "Starbucks",    "priceperseat": 0,    "Duration": 0,    "seatsAvailable": 4,    "petsAllowed": "No",    "carModel": "Toyota",    "carType": "SUV",    "addlNotes": "Please be present on time",    "PhoneNumber": "352-352-3523",    "UserId": 6,    "ToStartTime": "2022-02-03T03:10:50-05:00",    "ToEndTime": "2022-02-05T03:10:22-05:00"},    {    "ID": 82,    "CreatedAt": "2022-03-04T16:37:22.294977-05:00",    "UpdatedAt": "2022-03-04T16:37:22.294977-05:00",    "DeletedAt": null,    "Name": "kalyan",    "startLocation": "Gainesville",    "fromCity": "Gainesville",    "toCity": "Ocla",    "destLocation": "Starbucks",    "priceperseat": 0,    "Duration": 0,    "seatsAvailable": 4,    "petsAllowed": "No",    "carModel": "Toyota",    "carType": "SUV",    "addlNotes": "Please be present on time",    "PhoneNumber": "352-352-3523",    "UserId": 6,    "ToStartTime": "2022-02-03T03:10:50-05:00",    "ToEndTime": "2022-02-05T03:10:22-05:00"},    {    "ID": 83,    "CreatedAt": "2022-03-04T16:37:23.009893-05:00",    "UpdatedAt": "2022-03-04T16:37:23.009893-05:00",    "DeletedAt": null,    "Name": "kalyan",    "startLocation": "Gainesville",    "fromCity": "Gainesville",    "toCity": "Ocla",    "destLocation": "Starbucks",    "priceperseat": 0,    "Duration": 0,    "seatsAvailable": 4,    "petsAllowed": "No",    "carModel": "Toyota",    "carType": "SUV",    "addlNotes": "Please be present on time",    "PhoneNumber": "352-352-3523",    "UserId": 6,    "ToStartTime": "2022-02-03T03:10:50-05:00",    "ToEndTime": "2022-02-05T03:10:22-05:00"}  ]
```