

# SOFTWARE ENGINEERING (CEN5035)

## GatorX

(A web application to view and review places)

### Backend API documentation

#### Developers:

#### APIs:

1. User Management (Register, login, etc.)
2. View/Add places
3. View/Post/Edit Reviews

#### User Management API:

##### Description:

- This API allows the user to register by entering their credentials and getting added to the database.
- Once the user is registered, they can login into the website whenever, using their email address and password
- Registering and logging in is mandatory for any user trying to post any new reviews for the visible locations on the site

##### 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:

POSTlocalhost:8080/register

Send

ParamsAuthorizationHeaders (9)BodyPre-request ScriptTestsSettingsCookies

noneform-datax-www-form-urlencodedrawbinaryGraphQLText

```
1 {
2   ... "name": "Roshit M",
3   ... "email": "mr@ufl.edu",
4   ... "password": "secoursepwd",
5   ... "phone": "09873421"
6 }
```

BodyCookies (1)Headers (3)Test Results

Status: 200 OKTime: 10 msSize: 160 BSave Response

PrettyRawPreviewVisualizeJSON

```
1 {
2   "result": "User created in database"
3 }
```

POSTlocalhost:8080/login

Send

ParamsAuthorizationHeaders (9)BodyPre-request ScriptTestsSettingsCookies

noneform-datax-www-form-urlencodedrawbinaryGraphQLText

```
1 {
2   ... "email": "m@ufl.edu",
3   ... "password": "secoursepwd"
4 }
```

BodyCookies (1)Headers (4)Test Results

Status: 200 OKTime: 13 msSize: 372 BSave Response

PrettyRawPreviewVisualizeJSON

```
1 {
2   "result": "login success"
3 }
```

## API for Viewing/Adding places:

### Description:

- This API sends a get request whenever the user wishes to view all the places available to review.
- A post request is sent whenever the user wishes to add a new place to the database. The following are the required parameters for sending a post request to add a new place:  
*placename, location, type, avgrating.*

### Acceptance Criteria:

- The API should throw an error message whenever any of the above mentioned required parameters aren't specified.
- Whenever a get request is sent the API should send a response with the list of places along with the values of all the other place parameters.

### Sample Requests and Responses:

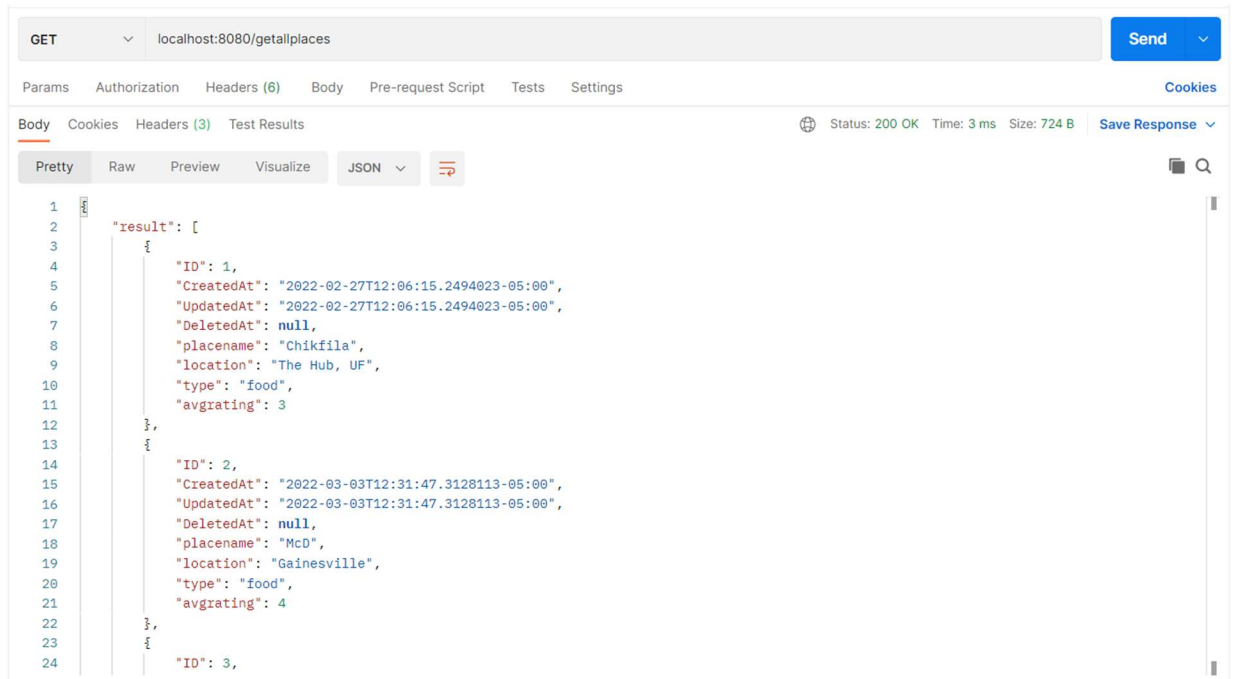
The screenshot displays a REST client interface with a POST request to `localhost:8080/postplace`. The request body is a JSON object with the following fields: `placename` (Dominos), `location` (Gainesville), `type` (restaurant), and `avgrating` (4). The response status is 200 OK, and the response body is a JSON object with the field `result` (Place created in database).

```
POST localhost:8080/postplace

{
  "placename": "Dominos",
  "location": "Gainesville",
  "type": "restaurant",
  "avgrating": 4
}
```

Status: 200 OK Time: 10 ms Size: 161 B

```
{
  "result": "Place created in database"
}
```



```
1  {
2    "result": [
3      {
4        "ID": 1,
5        "CreatedAt": "2022-02-27T12:06:15.2494023-05:00",
6        "UpdatedAt": "2022-02-27T12:06:15.2494023-05:00",
7        "DeletedAt": null,
8        "placename": "Chikfila",
9        "location": "The Hub, UF",
10       "type": "food",
11       "avgrating": 3
12     },
13     {
14       "ID": 2,
15       "CreatedAt": "2022-03-03T12:31:47.3128113-05:00",
16       "UpdatedAt": "2022-03-03T12:31:47.3128113-05:00",
17       "DeletedAt": null,
18       "placename": "McD",
19       "location": "Gainesville",
20       "type": "food",
21       "avgrating": 4
22     },
23     {
24       "ID": 3,
```

## API for Viewing, Posting or Editing reviews:

### Description:

- Logged in users can add a review to a corresponding place. The API receives a POST request with the place id of the corresponding place.
- Reviews from other users for all the places can be viewed
- Previously posted reviews can be edited.

### Acceptance Criteria:

- Whenever the user is not logged in and tries to write a review, the API should throw an error message saying: *"User not logged in"*
- The API should throw an error message when any of the following required fields are not filled, when trying to post a review: *reviewtitle, review, rating, placeid, reviewerid*
- The review id should be included in the URL when trying to edit an old review.
- Whenever a get request is sent, the API should send data having all the previously posted reviews that are stored on the database.
- When a delete review request is sent to the API along with the review id, the API should remove the corresponding review from the database.

## Sample Requests and Responses:

POST localhost:8080/postreview

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL Text

```
1 {
2   "reviewtitle": "good sandwiches",
3   "review": "good",
4   "rating": 3,
5   "placeid": 1,
6   "reviewerid": 1
7 }
```

Body Cookies (1) Headers (3) Test Results

Status: 200 OK Time: 10 ms Size: 162 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "result": "Review created in database"
3 }
```

GET localhost:8080/getallreviews

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Body Cookies (1) Headers (3) Test Results

Status: 200 OK Time: 4 ms Size: 344 B Save Response

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "ID": 2,
4     "CreatedAt": "2022-03-04T12:34:41.8455644-05:00",
5     "UpdatedAt": "2022-03-04T12:34:41.8455644-05:00",
6     "DeletedAt": null,
7     "reviewtitle": "good sandwiches",
8     "review": "good",
9     "rating": 3,
10    "placeid": 1,
11    "reviewerid": 3
12  }
13 ]
14 ]
15 ]
```

## Sprint 3:

### GetPlaceByIdView:

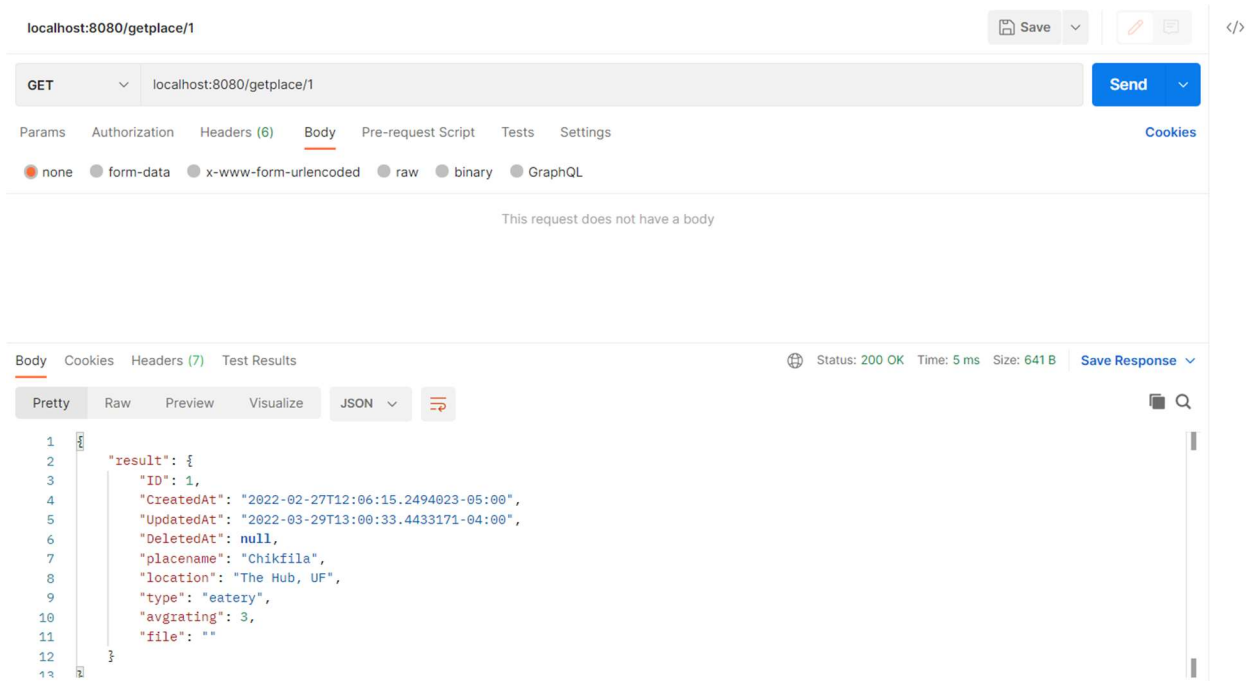
#### Description:

- Logged in users can retrieve a place corresponding to an ID. The API receives a GET request with the place id of the corresponding place.

- Place details like location, name, etc could be viewed.

#### Acceptance Criteria:

- The place id should be included in the URL when trying to get the place details.
- Whenever a get request is sent, the API should send data having all the place details from the database.



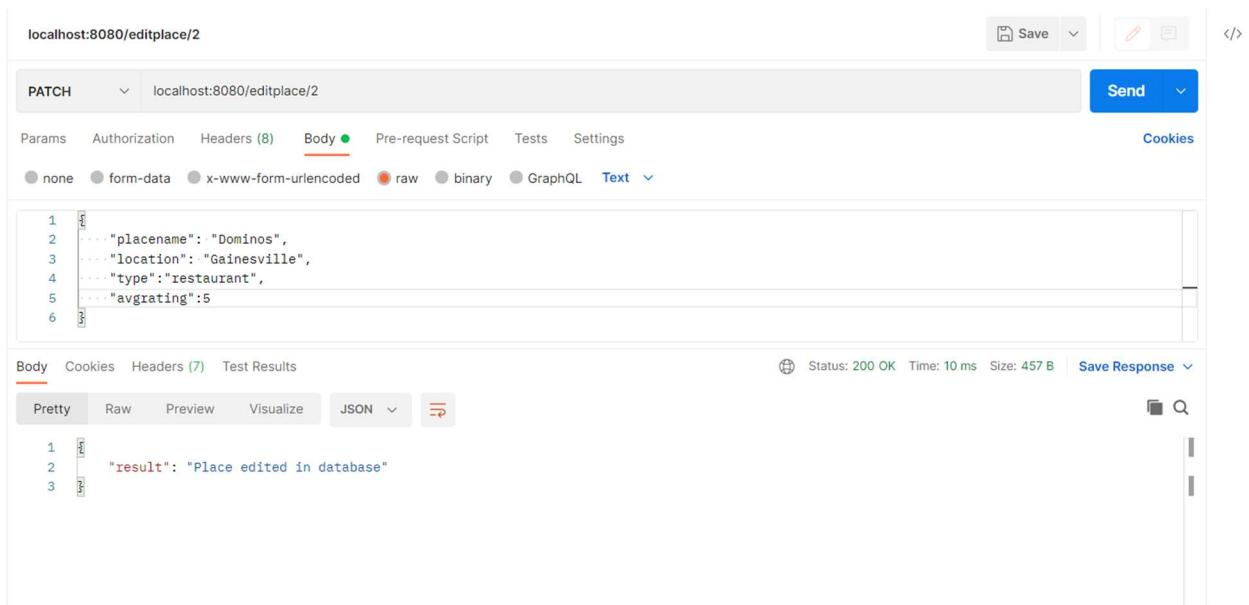
#### EditplaceView:

##### Description:

- Logged in users can edit the place details corresponding to a place ID. The API receives a PATCH request with the place id of the corresponding place.
- A patch request with a format similar to the create place request gets sent as a json file with changes made to whichever fields the user desires.
- All the previously created places in the database can be edited.

#### Acceptance Criteria:

- Whenever the user is not logged in and tries to edit a place, the API should throw an error message saying: *"User not logged in"*
- The API should throw an error message when any of the following required fields are not filled, when trying to edit a place: *placename, location, type, avgrating*.
- The place id should be included in the URL when trying to edit a previously created place.
- Whenever a patch request is sent, the API should send back an appropriate acknowledgement message.



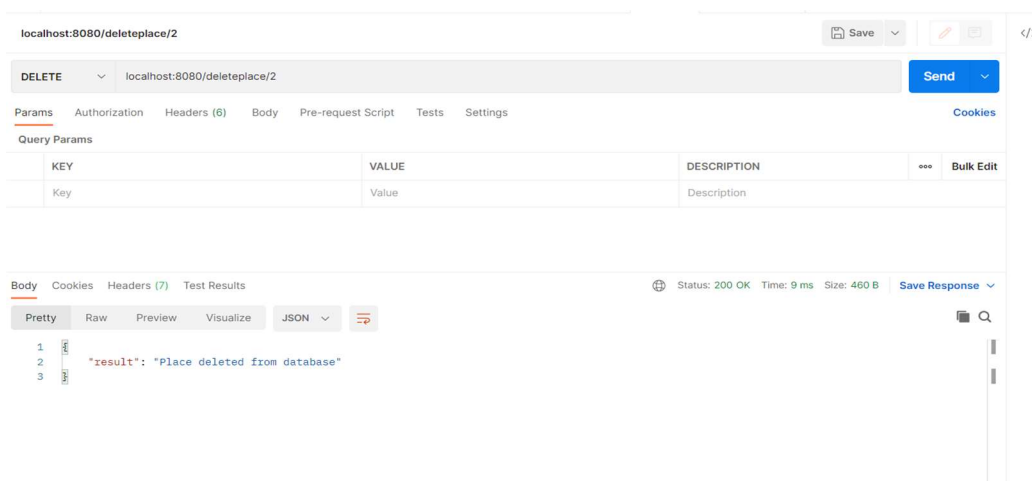
## DeleteplaceView:

### Description:

- Logged in users can delete an existing place from the database corresponding to a place ID. The API receives a DELETE request with the place id of the corresponding place.
- Any of the previously created places in the database can be deleted by appending the place id to the APIs url.

### Acceptance Criteria:

- The API should throw an error message when the place id in the url is not valid.
- The place id should be included in the URL when trying to delete a previously created place.
- Whenever a delete request is sent, the API should send back an appropriate acknowledgement message.



## GetreviewsbyuserView:

### Description:

- Logged in users can retrieve all reviews for all places. The API receives a GET request and send all the reviews data stored in the database
- Review details like location, name, place name, title and rating could be viewed.
- All the previously posted reviews in the database can be viewed.

### Acceptance Criteria:

- Whenever the user is not logged in and tries to edit a place, the API should throw an error message saying: *"User not logged in"*
- Whenever a get request is sent, the API should send back an appropriate acknowledgement message with all the review details.

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** localhost:8080/getuserreviews
- Status:** 200 OK
- Time:** 4 ms
- Size:** 649 B

The response body is displayed in JSON format:

```
1  {
2    "result": [
3      {
4        "ID": 3,
5        "CreatedAt": "2022-04-01T21:23:05.512441-04:00",
6        "UpdatedAt": "2022-04-01T21:23:05.512441-04:00",
7        "DeletedAt": null,
8        "reviewtitle": "good sandwiches",
9        "review": "good",
10       "rating": 3,
11       "placeid": 1,
12       "reviewerid": 3,
13       "file": ""
14     }
15   ]
16 }
```

## GetreviewsbyplaceView:

### Description:

- Logged in users can retrieve all reviews for a particular place corresponding to an ID. The API receives a GET request along with the place id and sends all the reviews data stored in the database for that specific place.
- Review details like location, name, place name, title and rating could be viewed.
- All the previously posted reviews for the corresponding place id in the database can be viewed.



## Acceptance Criteria:

- Whenever the user is not logged in and tries to edit a place, the API should throw an error message saying: *"User not logged in"*
- The API should send a relevant error message when the place id appended to the URL is not valid.
- Whenever a get request is sent, the API should send back an appropriate acknowledgement message with all the review details.

localhost:8080/getplacereviews/1 Save </>

GET localhost:8080/getplacereviews/1 Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies (1) Headers (7) Test Results Status: 200 OK Time: 7 ms Size: 907 B Save Response

Pretty Raw Preview Visualize JSON

```
2  "result": [  
3    {  
4      "ID": 2,  
5      "CreatedAt": "2022-03-04T18:08:25.3898836-05:00",  
6      "UpdatedAt": "2022-03-29T13:00:59.9728626-04:00",  
7      "DeletedAt": null,  
8      "reviewtitle": "good sandwiches",  
9      "review": " The food here is good especially sandwiches",  
10     "rating": 3,  
11     "placeid": 1,  
12     "reviewerid": 2,  
13     "file": ""  
14   }  
15 ]
```