




# Restaurant Scheduler on Cloud

Group: 404NotFound



# Project Idea

"Where Should We Eat?" is a timeless, exhausting, and frustrating struggle, especially when you're going out with a group of people.

We want to build an application to help users select restaurants for group dining

# Microservices

- User Microservice
- Restaurant Microservice
- Schedule Microservice

# Databases

- User Database
- Restaurant Database
- Schedule Database

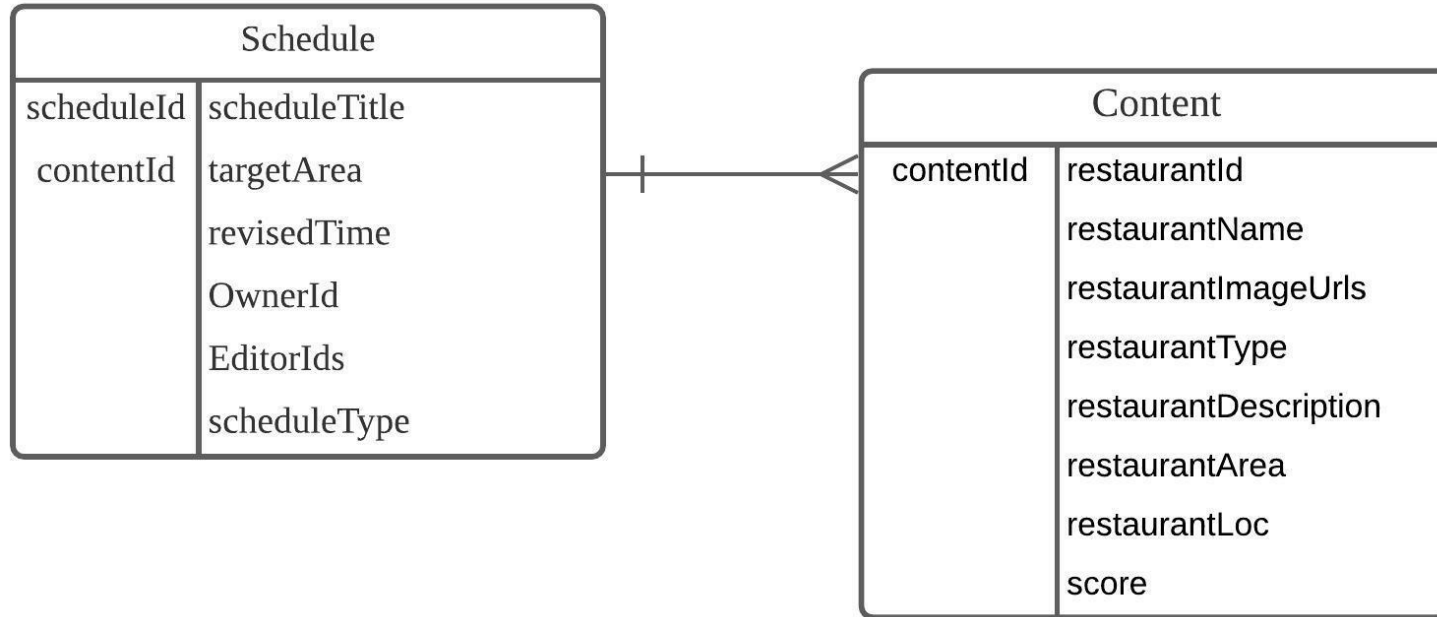
userID	userName	userEmail	userPWD	address	Preference	Links
1	Wenqing Zhong	wz2557@columbia.edu	12345	NY10027	{"1": "Chinese", "2": "Japanese", "3": "Korean"}	{"rel": "self", "href": "/users/1"}
2	Xiyuan Zhao	xz2994@columbia.edu	123	NY10025	{"1": "Chinese", "2": "Korean", "3": "Thai"}	{"rel": "self", "href": "/users/2"}
3	Moxin Xu	mx2237@hotmail.com	123	NY10025	{"1": "Chinese", "2": "Korean", "3": "Japanese"}	{"rel": "self", "href": "/users/3"}
4	Zhejiang Jin	zj2324@columbia.edu	123	NY10025	{"1": "American", "2": "Indian", "3": "Italian", "4": "Congo"}	{"rel": "self", "href": "/users/4"}
6	Aandy	11@qq.com	11	WA1111	{}	{}

rid	name	owner	type	location	links
1	New_name_for_restaurant1	Manson Wudy	Pizza	10029	{"rel": "self", "href": "/restaurants/1"}
2	Massawa	Wooden Wuddy	Seafood	10027	{"rel": "self", "href": "/restaurants/2"}
3	12	Stephen Wang	Chinese	10025	{"rel": "self", "href": "/restaurants/3"}

contentId	restaurantId	restaurantName	restaurantImageUrls	restaurantType	restaurantDescription	restaurantArea	restaurantLoc	score
1	1	Thai Noodle	www.baidu.com	Thai	This restaurant sells Thai noodle	NY10027	50,127	70
2	2	Shanghai Cai	www.google.com	Shanghai	This is a tasty Shanghai Food	NY10001	30,50	80

scheduledId	contentId	scheduledTitle	revisedTime	targetArea	ownerId	editorIds
1	1	schedule1	2021-12-22 20:19:59	NY10027	1	2
1	2	schedule1	2021-12-22 20:24:27	NY10027	1	2
2	3	schedule2	2021-12-22 20:41:27	NY10014	3	1

# Schedule Database



# User Microservice

Url path(inserting/updating/deleting tuples):

-GET(/users/?name=M, /users/?name=Moxin&limit=1&offset=1)

-POST, DELETE(/delete\_user) {'uid':10}

Interface:

First page(sign in or sign up):

**Login from here:**

User Name:

Password:

Second page(chosen user/schedule and create):

Link: <http://18.222.171.216:5000/>

Please enter the user name:

Please enter the user id:

Please enter the user email:

Please enter the user password:

Please enter the user address:

Please enter the user preferences:

Recommend Just For You!

User



Create



**Create schedule:**

Schedule Title:

Target Area:

Restaurant Name:

Restaurant Type:

# Restaurant Microservice

Listing all the restaurants to choose

And inserting/searching/updating/deleting tuples of the restaurants:


E.g. Update the restaurants name based on restaurant id.

The screenshot shows a REST client interface with a POST request to `http://127.0.0.1:5000/hw1_db/restaurants/update/3/new_restaurant_name`. The response body is displayed in a 'Pretty' HTML format, showing a single line of JSON: `{ "Restaurant name updated successfully!" }`.

POST ⌵ `http://127.0.0.1:5000/hw1_db/restaurants/update/3/new_restaurant_name`

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

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize HTML ⌵ 

```
1 { "Restaurant name updated successfully!" }
```

# Schedule Microservice

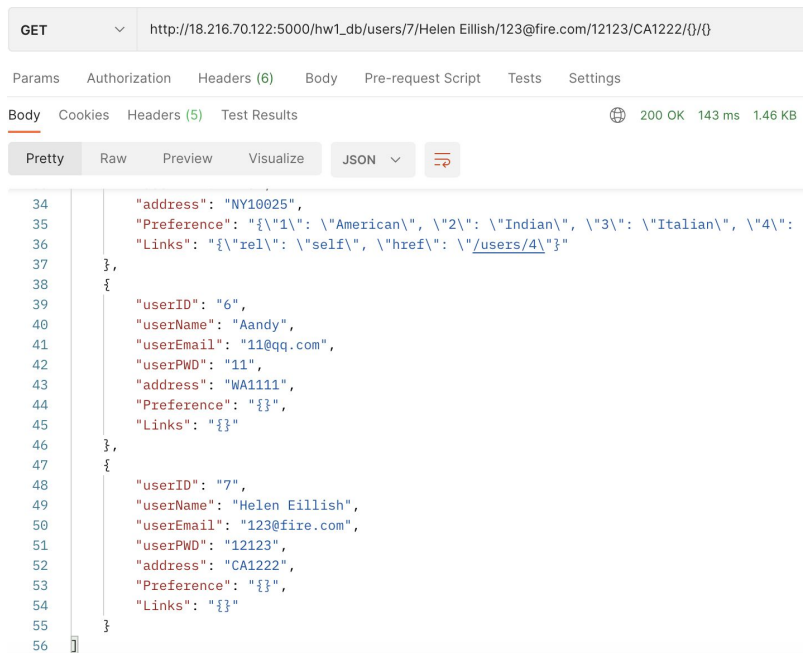
## Endpoints:

- **GET /schedule:** retrieve the schedule list by user id
- **POST /schedule:** create a preselected type schedule
- **GET /schedule/{scheduleId}:** retrieve a schedule
- **POST /schedule/{scheduleId}:** update a schedule
- **DELETE /schedule/{scheduleId}:** delete a schedule
- **GET schedule/{scheduleId}/restaurant/{restaurantId}:** retrieve a like info of a restaurant in the schedule
- **PUT schedule/{scheduleId}/restaurant/{restaurantId}:** initialize a like info of a restaurant in the schedule
- **DELETE schedule/{scheduleId}/restaurant/{restaurantId}:** delete a like info of a restaurant in the schedule
- **GET /schedule/{scheduleId}/submit:** change the stage of the schedule from to COMPLETED
- **GET /schedule/{scheduleId}/finish:** change the stage of the schedule from EDITING to COMPLETED



# What we accomplished

- Microservices that support CRUD operations for users



The screenshot shows a REST client interface with a GET request to `http://18.216.70.122:5000/hw1_db/users/7/Helen Eillish/123@fire.com/12123/CA1222/()/{}()`. The response is a JSON array of three user objects, displayed in 'Pretty' view. The first object is for user ID 5, the second for user ID 6, and the third for user ID 7. Each object contains fields for `address`, `Preference`, `Links`, `userID`, `userName`, `userEmail`, and `userPWD`.

```
34   "address": "NY10025",
35   "Preference": "{\"1\": \"American\", \"2\": \"Indian\", \"3\": \"Italian\", \"4\":",
36   "Links": "{\"rel\": \"self\", \"href\": \"/users/4/\"}"
37 },
38 {
39   "userID": "6",
40   "userName": "Aandy",
41   "userEmail": "11@qq.com",
42   "userPWD": "11",
43   "address": "WA1111",
44   "Preference": "{}",
45   "Links": "{}"
46 },
47 {
48   "userID": "7",
49   "userName": "Helen Eillish",
50   "userEmail": "123@fire.com",
51   "userPWD": "12123",
52   "address": "CA1222",
53   "Preference": "{}",
54   "Links": "{}"
55 }
56 ]
```

# What we accomplished

- Microservices that support CRUD operations for restaurants

```
GET http://127.0.0.1:5000/hw1_db/restaurants/

Params Authorization Headers (6) Body Pre-request Script Tests Settings
Body Cookies Headers (5) Test Results Status: 200 OK
Pretty Raw Preview Visualize JSON

1 {
2   {
3     "rid": "1",
4     "name": "New_name_for_restaurant1",
5     "owner": "Manson Wudy",
6     "type": "Pizza",
7     "location": "10029",
8     "links": {"rel": "self", "href": "/restaurants/1"}
9   },
10  {
11    "rid": "2",
12    "name": "Massawa",
13    "owner": "Wooden Wuddy",
14    "type": "Seafood",
15    "location": "10027",
16    "links": {"rel": "self", "href": "/restaurants/2"}
17  },
18  {
19    "rid": "3",
20    "name": "12",
21    "owner": "Stephen Wang",
22    "type": "Chinese",
23    "location": "10025",
24    "links": {"rel": "self", "href": "/restaurants/3"}
25  },
26 }
```

```
GET http://127.0.0.1:5000/hw1_db/restaurants/search/rid/3

Params Authorization Headers (6) Body Pre-request Script Tests Settings
Body Cookies Headers (5) Test Results
Pretty Raw Preview Visualize JSON

1 {
2   {
3     "rid": "3",
4     "name": "12",
5     "owner": "Stephen Wang",
6     "type": "Chinese",
7     "location": "10025",
8     "links": {"rel": "self", "href": "/restaurants/3"}
9   }
10 }
```

# What we accomplished

- Microservices that support CRUD operations for schedule

18.220.208.27:5003/schedules

POST 18.220.208.27:5003/schedules

Params Authorization Headers (8) **Body** Pre-request Script

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary

```
1 {
2   "scheduledId": "4",
3   "targetArea": "NY999",
4   "ownerId": "3",
5   "editorId": "1"
6 }
7
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

1 1

18.220.208.27:5003/schedules/2

GET 18.220.208.27:5003/schedules/2

Params Authorization Headers (8) **Body** Pre-

Query Params

KEY
Key

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "scheduledId": "2",
3   "targetArea": "NY001",
4   "ownerId": "5",
5   "editorId": "3"
6 }
7
8
```

# What we accomplished

- API Gateway

The screenshot shows the Amazon API Gateway console for the API 'e6156 (p1snq6g5ge)'. The left sidebar contains navigation links: APIs, Custom Domain Names, VPC Links, Resources (selected), Stages, Authorizers, Gateway Responses, Models, Resource Policy, Documentation, Dashboard, Settings, Usage Plans, API Keys, Client Certificates, and Settings. The main panel displays the configuration for the resource '/schedules - GET - Method Test'. It includes a 'Method Execution' tab, a 'Path' section with a note about path parameters, 'Query Strings' with a parameter 'param1=value1&param2=value2', 'Headers' with a note about header syntax, 'Stage Variables' with a note about stage variables, 'Client Certificate', and 'Response Headers'. The bottom of the console shows a footer with 'Feedback', 'English (US)', and copyright information.

The screenshot shows a web browser displaying the response body of the /schedules - GET - Method Test resource. The response is a JSON array of objects, each representing a scheduled task. The objects contain fields for 'scheduledId', 'targetArea', 'ownerId', and 'editorId'. The response is as follows:

```
[{"scheduledId": "3", "targetArea": "NY10040", "ownerId": "2", "editorId": "1"}, {"scheduledId": "5", "targetArea": "NY999", "ownerId": "3", "editorId": "4"}, {"scheduledId": "1", "targetArea": "NY10030", "ownerId": "4", "editorId": "1"}, {"scheduledId": "2", "targetArea": "NY999", "ownerId": "5", "editorId": "4"}]
```

# What we accomplished

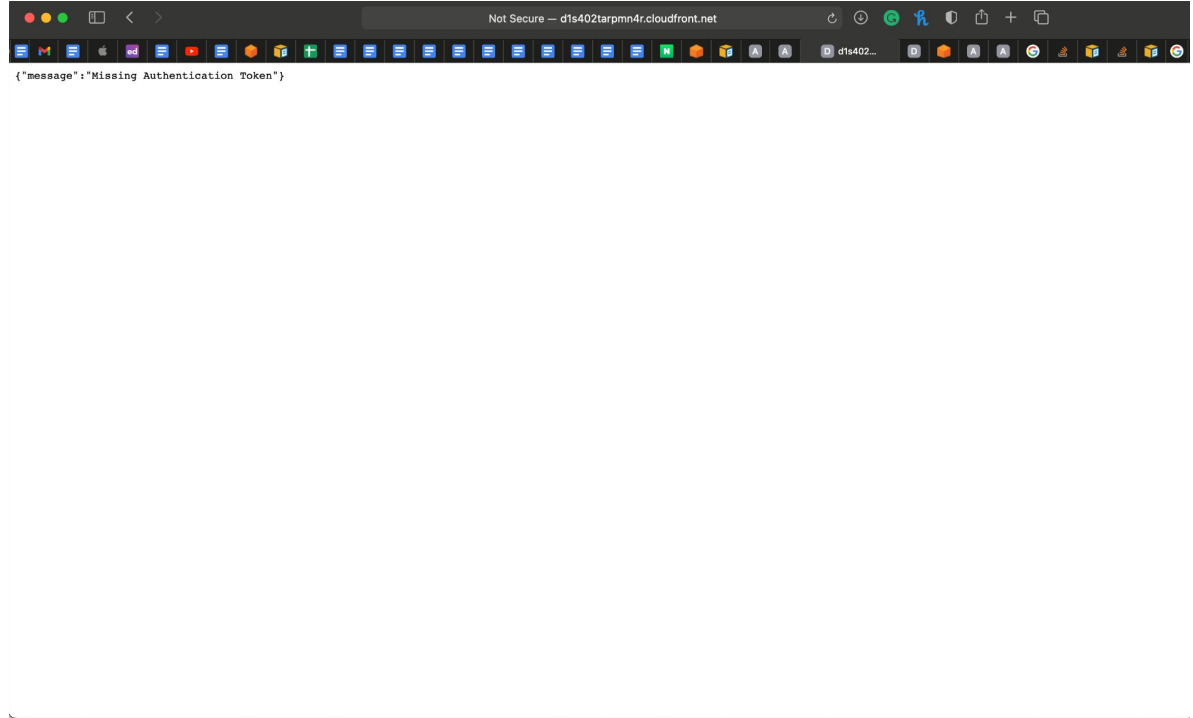
- S3 on CloudFront

A screenshot of a web browser window. The address bar shows a URL starting with 'ui404notfound.s3-website.us-east-2.amazonaws.com'. The page content displays a heading 'Here are our restaurants:)' followed by a table with three columns: 'Restaurants Name', 'Type', and 'Location'. The table lists three restaurants: 'ELysian Fields Cafe' (Pizza, 10029), 'Massawa' (Seafood, 10027), and 'Szechuan Garden' (Chinese, 10025).

Restaurants Name	Type	Location
ELysian Fields Cafe	Pizza	10029
Massawa	Seafood	10027
Szechuan Garden	Chinese	10025

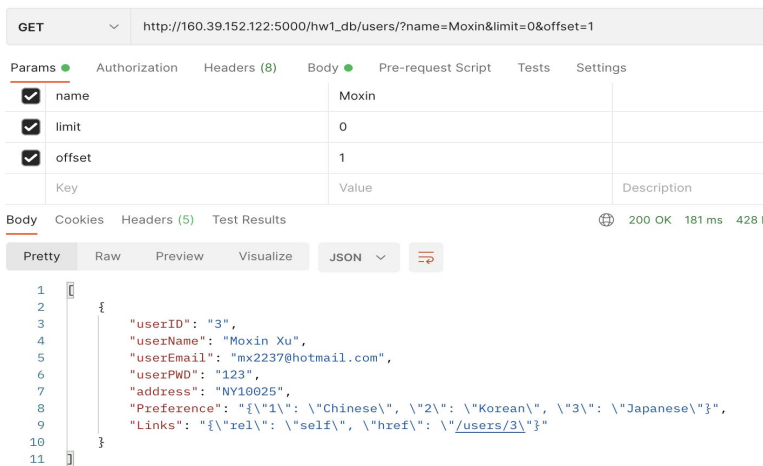
# What we accomplished

- Problem we faced



# What we accomplished

- Pagination, Query support



The screenshot shows a REST client interface with a GET request to `http://160.39.152.122:5000/hw1_db/users/?name=Moxin&limit=0&offset=1`. The Params tab is active, showing three parameters: `name` (value: Moxin), `limit` (value: 0), and `offset` (value: 1). The Body tab shows a JSON response with user details.

Key	Value	Description
name	Moxin	
limit	0	
offset	1	

Body: Cookies Headers (5) Test Results 200 OK 181 ms 428 B

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "userID": "3",
4     "userName": "Moxin Xu",
5     "userEmail": "mx2237@hotmail.com",
6     "userPWD": "123",
7     "address": "NY10025",
8     "Preference": "{\"1\": \"Chinese\", \"2\": \"Korean\", \"3\": \"Japanese\"}",
9     "Links": "{\"rel\": \"self\", \"href\": \"/users/3\"}"
10  }
11 }
```

```
@app.route('/hw1_db/users/', methods=['GET'])
def get_users_by_name():
    name, limit, offset = request.args.get('name'), request.args.get('limit'), request.args.get('offset')
    if limit is None and offset is None:
        limit, offset = '0', '100'
    print(name, limit, offset)
    res = IMDBUsersResource.get_by_name(name, limit, offset)
    if res == False:
        rsp = Response('Find 0 result.', status=404, content_type="application/json")
    else:
        rsp = Response(json.dumps(res), status=200, content_type="application/json")
    return rsp
```