

Project 3

Redis and DynamoDB Local

GROUP 1

Luis Alverado

Marco Gabriel

Michael Carey

Rishub Goel

Nhan Mac

CPSC 449-01 13661

Fall Semester

Table of Contents

Install and configure the AWS CLI	3
Install and configure Amazon DynamoDB local	3
Install the AWS SDK for Python	4
New Database Schema For Dynamodb and Redis	5
Design a data model for each database	5
Dynamodb Schema	5
Class table:	5
User table:	5
Redis	5
Class waitlists, with key:	5
Student waitlists, with key:	5
Student Endpoint	6
GET: Available classes	6
POST: Enroll into an available class	7
PUT: Drop a class	8
Waitlist Endpoint	9
GET: View waiting List from Student	9
PUT: Remove from the waitlist	10
GET: View current waitlist from Instructor	11
Instructor Endpoint	12
Get Instructor Enrollment	12
Get Instructors Dropped Students	13
Instructor Drop Student	14
Registrar Endpoint	17
POST: Create Class	17
DELETE: Remove Class	18
PUT: Change instructor	19
POST: Create User	20

Install and configure the AWS CLI

Context: Installing AWS CLI

```
(.venv) student@tuffix-vm:~$ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
```

```
(.venv) student@tuffix-vm:~$ aws --version
aws-cli/2.13.32 Python/3.11.6 Linux/6.2.0-36-generic exe/x86_64.ubuntu.22 prompt/off
```

Context: Configuring dummy credentials for DynamoDB local

```
(.venv) student@tuffix-vm:~$ aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: json
(.venv) student@tuffix-vm:~$
```

Install and configure Amazon DynamoDB local

```
(.venv) student@tuffix-vm:~$ sudo apt install --yes openjdk-19-jre-headless
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  gir1.2-snapd-1 python3-debconf update-notifier-common
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  ca-certificates-java java-common
Suggested packages:
```

```
(.venv) student@tuffix-vm:~/Desktop$ java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar -sharedDb
Initializing DynamoDB Local with the following configuration:
Port:      8000
InMemory:   false
DbPath: null
SharedDb:   true
shouldDelayTransientStatuses: false
CorsParams: null
```

```
(.venv) student@tuffix-vm:~/Desktop/CPSC449-Project_3$ aws dynamodb list-tables --endpoint-url http://localhost:5500
{
  "TableNames": [
    "enrollment_class",
    "enrollment_user"
  ]
}
```

Install the AWS SDK for Python

```
(.venv) student@tuffix-vm:~/Desktop$ python -m pip install boto3
Collecting boto3
  Downloading boto3-1.28.82-py3-none-any.whl (135 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 135.8/135.8 KB 2.7 MB/s eta 0:00:00
Collecting s3transfer<0.8.0,>=0.7.0
  Downloading s3transfer-0.7.0-py3-none-any.whl (79 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.8/79.8 KB 3.1 MB/s eta 0:00:00
Collecting botocore<1.32.0,>=1.31.82
  Downloading botocore-1.31.82-py3-none-any.whl (11.3 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.3/11.3 MB 8.6 MB/s eta 0:00:00
Collecting jmespath<2.0.0,>=0.7.1
  Downloading jmespath-1.0.1-py3-none-any.whl (20 kB)
Collecting python-dateutil<3.0.0,>=2.1
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 247.7/247.7 KB 4.1 MB/s eta 0:00:00
Requirement already satisfied: urllib3<2.1,>=1.25.4 in /home/student/.venv/lib/pyth
Collecting six>=1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, jmespath, python-dateutil, botocore, s3transfer
Successfully installed boto3-1.28.82 botocore-1.31.82 jmespath-1.0.1 python-dateuti
```

New Database Schema For Dynamodb and Redis

Design a data model for each database

Context: There are two tables located in the DynamoDB database, the class table and the user table. For both the class and user table, the id is the partition key, with a key type of HASH. Both also have a Global Secondary Index (GSI) called index-id, that uses the id attribute.

Dynamodb Schema

Class table:

- id: int
- name: str
- course_code: str
- section_number: int
- current_enroll: int
- max_enroll: int
- department: str
- instructor_id: int
- enrolled: list (of student_ids)
- dropped: list (of student_ids)

User table:

- id: int,
- name: str,
- roles: list (of strings)

Redis

Class waitlists, with key:

- "class:{class_id}:waitlist"
- format {class_id : [(student_id, placement)]}

Student waitlists, with key:

- "Student:{student_id}:waitlists"
- format {student_id: [(class_id, placement)]}

Student Endpoint

GET: Available classes

Context: We test the endpoint to see if we are able to get available classes. We use student 1 to see their classes that they are able to enroll in.

Endpoint: <http://0.0.0.0:5000/students/1/classes>



▼ Classes:	
▼ 0:	
id:	2
name:	"Web Back-End Engineering"
course_code:	"449"
section_number:	2
current_enroll:	24
max_enroll:	30
department:	"CPSC"
▼ instructor:	
id:	502
name:	"Vickie Rangel"
current_waitlist:	0
max waitlist:	15

POST: Enroll into an available class

Context: We test the endpoint to check if a student is enrolled. We use student 1 to enroll them into class 2. If they are not enrolled we get a response back stating the student was enrolled successfully else we get a different response as in screenshot 2.

Endpoint: <http://0.0.0.0:5000/students/1/classes/2/enroll>

Code	Details
200	Response body <pre>{ "message": "Student successfully enrolled in class" }</pre>

400 <i>Undocumented</i>	Error: Bad Request Response body <pre>{ "detail": "Student is already enrolled in this class or currently on waitlist" }</pre>
----------------------------	--

PUT: Drop a class

Context: We test the end point to check if the student is able to be dropped from the class. We use student 1 to get dropped from class 1. If the student is not enrolled in a class we get a different response such as the second screenshot.

Endpoint: <http://0.0.0.0:5000/students/1/classes/2/drop/>

Code	Details
200	Response body <pre>{ "message": "Student successfully dropped class" }</pre>

Code	Details
400 <i>Undocumented</i>	Error: Bad Request Response body <pre>{ "detail": "Student is not enrolled in the class" }</pre>

Waitlist Endpoint

GET: View waiting List from Student

Context: We test the endpoint to fetch the students' waitlist. We use student 1 to see their current waitlist. If a student is not on a waitlist we get a response that states the student is not on any waitlists.

Endpoint: <http://0.0.0.0:5000/waitlist/students/1>

Code	Details
200	Response body <pre>{ "waitlists": [{ "class_id": 4, "waitlist_position": 3 }, { "class_id": 8, "waitlist_position": 1 }, { "class_id": 13, "waitlist_position": 6 }] }</pre>

Code	Details
400 <i>Undocumented</i>	Error: Bad Request Response body <pre>{ "detail": "Student is not on a waitlist" }</pre>

PUT: Remove from the waitlist

Context: We test this endpoint to check if we are able to remove a student from the waitlist. We use student 1 and class 8 to remove them from the waitlist. If it is successful we get a message stating that it was able to remove the student. If the student is not on any waitlist it will give us a response as the second screenshot.

Endpoint: <http://0.0.0.0:5000/waitlist/students/1/classes/8/drop>

Code	Details
200	Response body <pre>{ "message": "Student removed from the waiting list" }</pre>

Code	Details
404 <i>Undocumented</i>	Error: Not Found Response body <pre>{ "detail": "Class not found" }</pre>

GET: View current waitlist from Instructor

Context: We test this endpoint to see if we are able to get the waitlists based of an instructor. We use instructor 508 to retrieve its class 8's waitlist. If the instructor has no waitlist we get a response just like in the second screenshot.

Endpoint: <http://0.0.0.0:5000/waitlist/instructors/508/classes/8>

Code	Details
200	<div>Response body</div> <pre>{ "waitlist": [{ "student": { "id": 1, "name": "James Smith" }, "waitlist_position": 1 }] }</pre>

Code	Details
400 <i>Undocumented</i>	<div>Error: Bad Request</div> <div>Response body</div> <pre>{ "detail": "Class does not have a waitlist" }</pre>

Instructor Endpoint

Get Instructor Enrollment

Context: I will try to get the enrollment of Instructor id 501 with class id 1.

Instructor

GET /instructors/{instructor_id}/classes/{class_id}/enrollment Get Instructor Enrollment

Parameters

Name	Description
Instructor_id * required integer (path)	501
class_id * required integer (path)	1

Execute Clear

Result: We ended up getting 10 students enrolled in the class.

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5000/instructors/501/classes/1/enrollment' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:5000/instructors/501/classes/1/enrollment
```

Server response

Code	Details
200	<p>Response body</p> <pre>{ "Enrolled": [{ "id": 1, "name": "James Smith" }, { "id": 2, "name": "Mary Cooke" }, { "id": 3, "name": "John Johnson" }, { "id": 4, "name": "Patricia Key" }, { "id": 5, "name": "Robert Williams" }, { "id": 6, "name": "Linda Kidd" }] }</pre> <p>Response headers</p>

Get Instructors Dropped Students

Context: Instructor_id 501 class id 1 has 4 students present to drop.

GET

/instructors/{instructor_id}/classes/{class_id}/drop

Get Instructor Dropped

⌵

Parameters

Cancel

Name	Description
Instructor_id * required integer (path)	<input type="text" value="501"/>
class_id * required integer (path)	<input type="text" value="1"/>

Execute

Clear

Result: When we make an API call to retrieve the list of dropped students we get student id's 11-14.

Curl

```
curl -X 'GET' \
  'http://localhost:5000/instructors/501/classes/1/drop' \
  -H 'accept: application/json'
```

Request URL

http://localhost:5000/instructors/501/classes/1/drop

Server response

Code

Details

200

Response body

```
{
  "Dropped": [
    {
      "id": 11,
      "name": "David Miller"
    },
    {
      "id": 12,
      "name": "Jennifer Murillo"
    },
    {
      "id": 13,
      "name": "Richard Davis"
    },
    {
      "id": 14,
      "name": "Maria Bolton"
    }
  ]
}
```

Download

Response headers

```
content-length: 146
content-type: application/json
date: Sat, 25 Nov 2023 05:54:25 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	No links

Instructor Drop Student

Context: Testing Endpoint using Instructor id = 1, class_id = 1, and student_id = 1 (James Smith).

POST /instructors/{instructor_id}/classes/{class_id}/students/{student_id}/drop Instructor Drop Class

Parameters

Name	Description
Instructor_id * required integer (path)	501
class_id * required integer (path)	1
student_id * required integer (path)	1

Execute

Result: In our endpoint, we get a message saying that “the student was successfully dropped”

Server response

Code	Details
200	<p>Response body</p> <pre>{ "Message": "Student successfully dropped" }</pre> <p>Response headers</p> <pre>content-length: 42 content-type: application/json date: Sat, 25 Nov 2023 06:16:10 GMT server: uvicorn</pre>

Responses

Code	Description	Links
200	Successful Response	No links

Media type: application/json

Controls Accept header.

Example Value | Schema

Context: Now, we will backtrack to the instructor enrollment list and dropped list to verify if the student was successfully dropped from the class and to ensure the lists get updated properly.

Test - Instructor Enrollment List Endpoint: We can see that id: 1 does not exist in the enrollment list of class_id:1 and instructor_id:501 anymore.

Code	Details
200	<div>Response body</div> <pre>{ "Enrolled": [{ "id": 2, "name": "Mary Cooke" }, { "id": 3, "name": "John Johnson" }, { "id": 4, "name": "Patricia Key" }, { "id": 5, "name": "Robert Williams" }, { "id": 6, "name": "Linda Kidd" }, { "id": 7, "name": "Michael Brown" }] }</pre> <div>Download</div>

Test - Instructor Dropped List Endpoint: We will now look at the dropped list endpoint and now we can see student id 1 “James Smith” in the list.

```
curl -X 'GET' \
  'http://localhost:5000/instructors/501/classes/1/drop' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:5000/instructors/501/classes/1/drop
```

Server response

Code	Details
200	<div>Response body</div> <pre>{ "Dropped": [{ "id": 11, "name": "David Miller" }, { "id": 12, "name": "Jennifer Murillo" }, { "id": 13, "name": "Richard Davis" }, { "id": 14, "name": "Maria Bolton" }, { "id": 1, "name": "James Smith" }] }</pre> <div>Download</div>

Context: If the Instructor decides to remove a student again we have a constraint set up to ensure it notifies the instructor that the student no longer exists in the class.

Request URL

http://localhost:5000/instructors/501/classes/1/students/1/drop

Server response

Code	Details
404	Error: Not Found

Undocumented

Response body

```
{  
  "detail": "Student not enrolled in this class"  
}
```

Download

Response headers

```
content-length: 47  
content-type: application/json  
date: Sat, 25 Nov 2023 06:22:23 GMT  
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	No links

Registrar Endpoint

POST: Create Class

Context: Creating a swift class using the Create Class Registrar Endpoint.

POST /registrar/classes/ Create Class

Parameters

Cancel

Reset

No parameters

Request body required

application/json

```
{  "id": 223,  "name": "Intro to Swift Programming (IOS)",  "course_code": "CPSC-223W",  "section number": 1,  "current_enroll": 0,  "max_enroll": 1,  "department_id": 1,  "instructor_id": 1}
```

Servers

These operation-level options override the global server options.

/

Execute

Clear

Test: Checking if class exists by using the Instructor Enrollment Endpoint. We should get a 200 status code with an empty "Enrolled" list.

Name

Description

Instructor_id required

integer

(path)

1

class_id required

integer

(path)

223

Execute

Clear

Responses

Curl

```
curl -X 'GET' \  'http://localhost:5000/instructors/1/classes/223/enrollment' \  -H 'accept: application/json'
```

Request URL

```
http://localhost:5000/instructors/1/classes/223/enrollment
```

Server response

Code

Details

200

Response body

```
{  "Enrolled": []}
```

Download

Response headers

Test - Add a new student to a new class.

student_id * required
integer
(path)

class_id * required
integer
(path)
Servers
These operation-level options override the global server options.

Result: Looking at the Instructor Enrolled List Endpoint now.

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5000/instructors/501/classes/223/enrollment' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:5000/instructors/501/classes/223/enrollment
```

Server response

Code	Details
200	<p>Response body</p> <pre>{ "Enrolled": [{ "id": 601, "name": "Luis Alvarado" }] }</pre> <input type="button" value="Download"/>

DELETE: Remove Class

Context: We test this endpoint if we can properly delete a class. We use class 2 to remove. If it is able to be removed we get back a response stating it was able to be removed. If the class does not exist we get a response back like in the second screenshot.

Endpoint: <http://0.0.0.0:5000/registrar/classes/2>

Code	Details
200	<p>Response body</p> <pre>{ "message": "Class removed successfully" }</pre>

Code	Details
404 <i>Undocumented</i>	Error: Not Found Response body <pre>{ "detail": "Class not found" }</pre>

PUT: Change instructor

Context: We test this endpoint to make sure that we are able to change the instructor to a new class. We use class 3 and instructor 505 to be changed to the new class. If it is successful we would get a response stating that the instructor was changed successfully. If the instructor was already changed we should get a response back just like in the second screenshot.

Endpoint: <http://0.0.0.0:5000/registrar/classes/3/instructors/505>

Code	Details
200	Response body <pre>{ "message": "Instructor changed successfully"}</pre>

Code	Details
400 <i>Undocumented</i>	Error: Bad Request Response body <pre>{ "detail": "Instructor already assigned to class"}</pre>

POST: Create User

Context: Creating a new student named “Luis Alvarado”

POST

/registrar/create_user

Create User

^

Parameters

Cancel

Reset

No parameters

Request body required

application/json

```
{  "name": "Luis Alvarado",  "roles": [    "Student"  ]}
```

Execute

Clear

Response: We successfully created a new student with incremented id.

```
}]
```

Request URL

http://localhost:5000/registrar/create_user

Server response

Code	Details
200	<div><div>Response body</div><div><pre>{ "Message": "user created successfully. Luis Alvarado Assigned id = 601"}</pre></div><div>Download</div></div> <div><div>Response headers</div><div><pre>content-length: 72 content-type: application/json date: Sun, 26 Nov 2023 04:45:05 GMT server: uvicorn</pre></div></div>

Responses

Code	Description	Links
200	Successful Response	No links

Media type

application/json

Controls Accept header.

Test - Enroll Student to an existing class.

Context: Added student 601 onto class id:1 and Instructor id 501. When calling we end up seeing student 601 in the enrolled list.

server response

Code

Details

200

Response body

```
{
  "id": 5,
  "name": "Robert Williams"
},
{
  "id": 6,
  "name": "Linda Kidd"
},
{
  "id": 7,
  "name": "Michael Brown"
},
{
  "id": 8,
  "name": "Barbara Duarte"
},
{
  "id": 9,
  "name": "William Jones"
},
{
  "id": 601,
  "name": "Luis Alvarado"
}
]
```

 Download

Response headers

```
content-length: 299
content-type: application/json
date: Sun, 26 Nov 2023 04:49:33 GMT
server: uvicorn
```

Responses