Web Backend Project 4

Semester	Fall 2023
Course Number	CPSC 449
Section	01
Project Number	4
Group Number	7
S.No	**Team Members**
1	Daniel Truong
2	Gaurav Warad
3	Nathan Storm
4	Ornella Irene Dsouza
5	Sagar Verma

Project requirements

https://docs.google.com/document/d/1BumTQHLYAym13K714sldMR8uGtvsmBWjMHdfvhjaA8I/edit

Task 1: Create enrollment notification service

Objective:

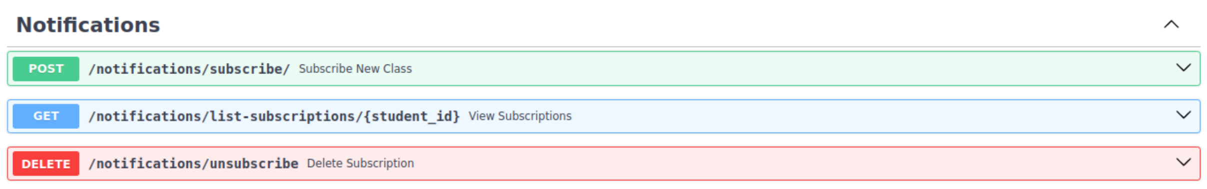
The objective of this task is to create an enrollment notification service that allows student to subscribe to a course, view all of their subscriptions, unsubscribe from a course

For our implementation, we'll use Redis to store the subscription with key-value design as following:

Key: student{student_id}:sub{class_id}

Set value: [webhook_url, email_id, class_id]

Here're the three APIs for enrollment notification services



Notifications

POST /notifications/subscribe/ Subscribe New Class

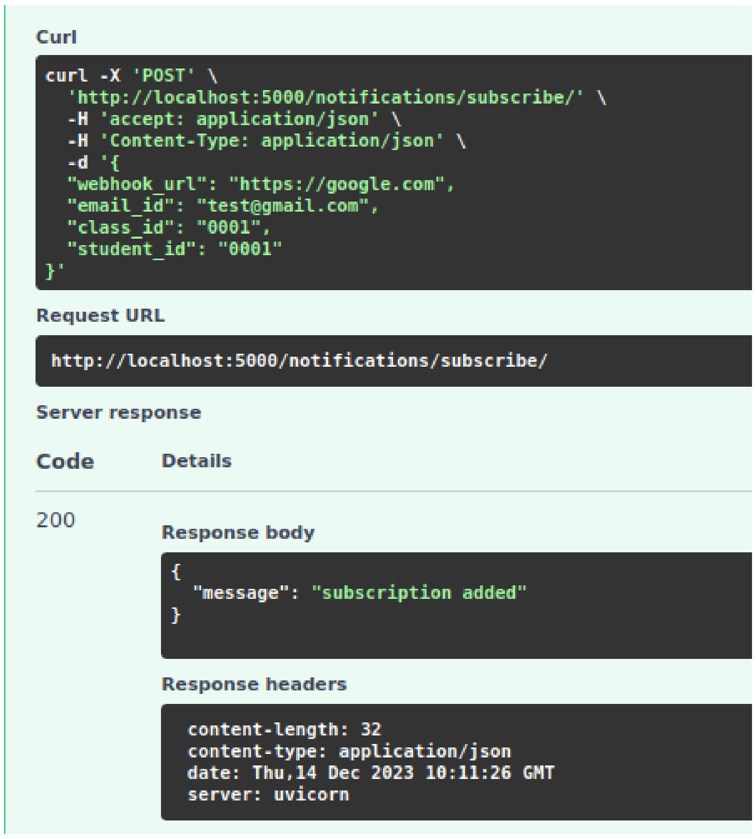
GET /notifications/list-subscriptions/{student_id} View Subscriptions

DELETE /notifications/unsubscribe Delete Subscription

POST /notification/subscribe

This API allows students to subscribe to a class giving student_id, email_id, class_id. It should also first check if student_id or class_id exist. It should also cover case where student already subscribe to the class

Here’s screenshot to see student subscribed successfully



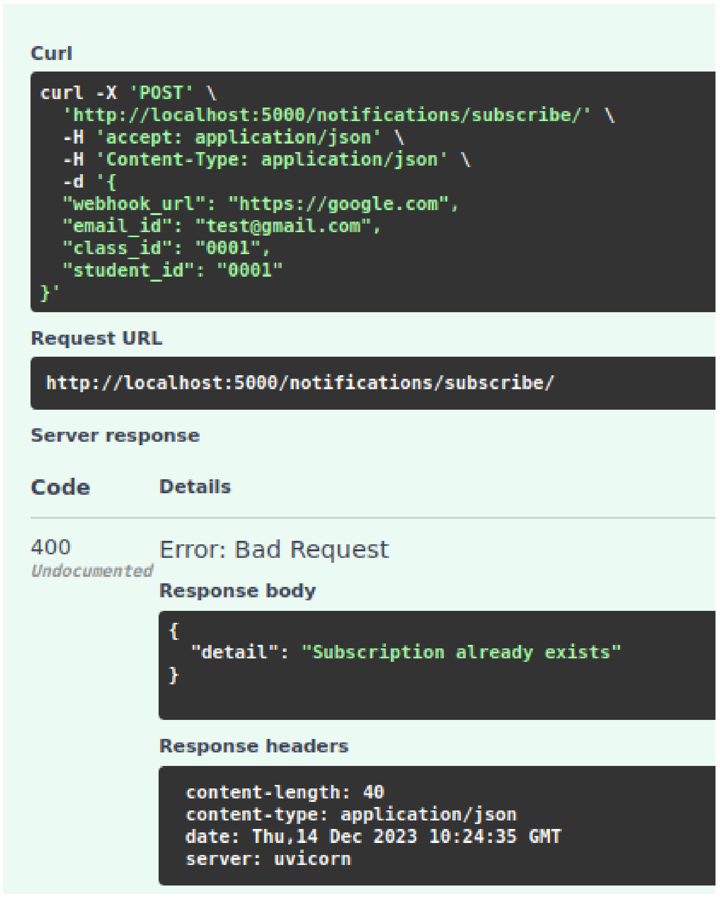
```
curl -X 'POST' \
  'http://localhost:5000/notifications/subscribe/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "webhook_url": "https://google.com",
  "email_id": "test@gmail.com",
  "class_id": "0001",
  "student_id": "0001"
}'
```

```
http://localhost:5000/notifications/subscribe/
```

```
{
  "message": "subscription added"
}
```

```
 content-length: 32
 content-type: application/json
 date: Thu,14 Dec 2023 10:11:26 GMT
 server: uvicorn
```

Here’s screenshot of case when student already subscribed to the class


Curl

```
curl -X 'POST' \
  'http://localhost:5000/notifications/subscribe/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "webhook_url": "https://google.com",
  "email_id": "test@gmail.com",
  "class_id": "0001",
  "student_id": "0001"
}'
```

Request URL

```
http://localhost:5000/notifications/subscribe/
```

Server response

Code	Details
400 *Undocumented*	Error: Bad Request

Response body

```
{
  "detail": "Subscription already exists"
}
```

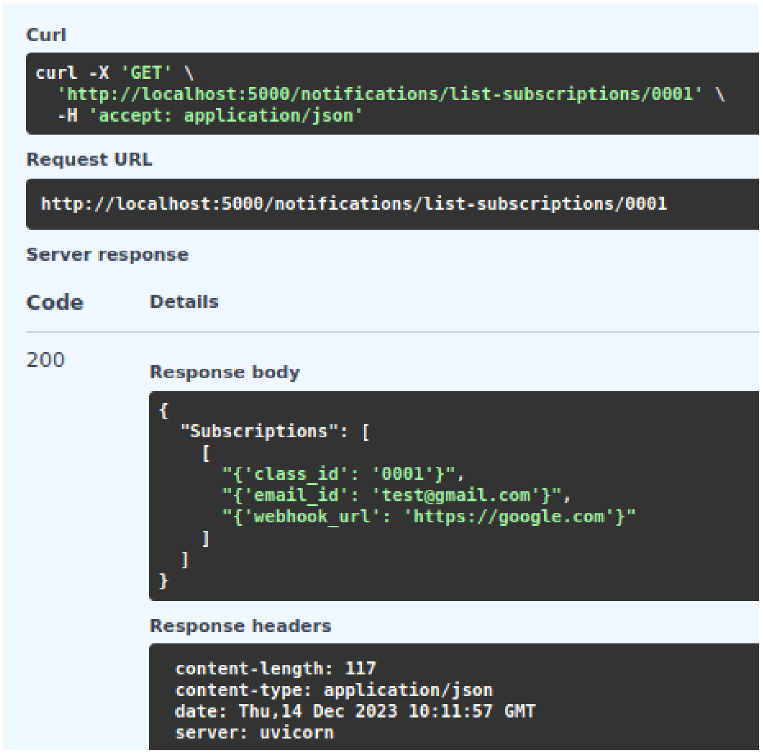
Response headers

```
 content-length: 40
 content-type: application/json
 date: Thu,14 Dec 2023 10:24:35 GMT
 server: uvicorn
```

GET /notifications/list-subscriptions/{student_id}

This API allows student to see details of all of their subscriptions including class_id, email_id, webhook_url

Here's a screenshot to see it's working


Curl

```
curl -X 'GET' \
  'http://localhost:5000/notifications/list-subscriptions/0001' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:5000/notifications/list-subscriptions/0001
```

Server response

Code Details

200

Response body

```
{
  "Subscriptions": [
    [
      "{'class_id': '0001'}",
      "{'email_id': 'test@gmail.com'}",
      "{'webhook_url': 'https://google.com'}"
    ]
  ]
}
```

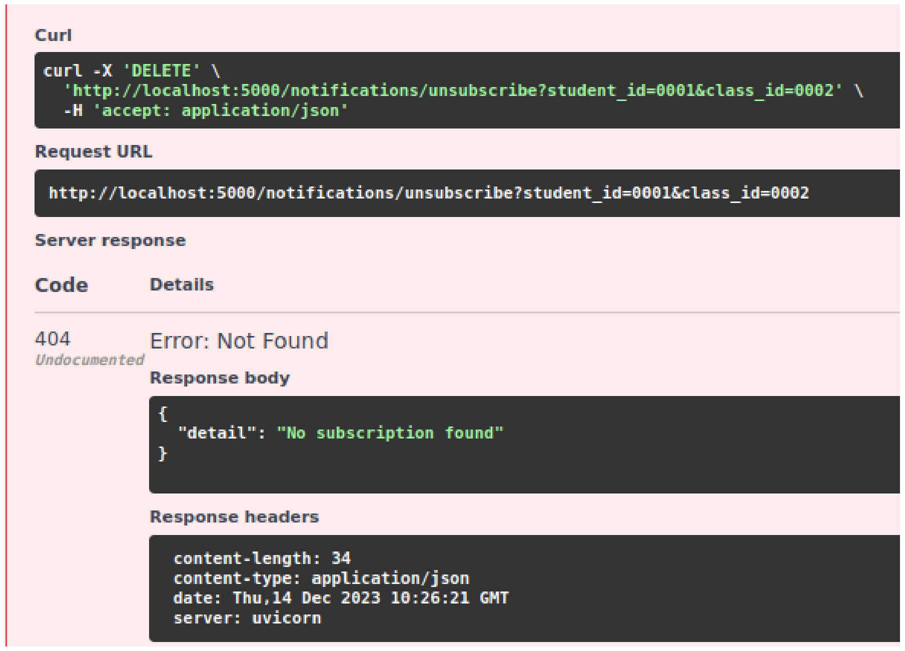
Response headers

```
 content-length: 117
 content-type: application/json
 date: Thu,14 Dec 2023 10:11:57 GMT
 server: uvicorn
```

DELETE /notifications/unsubscribe

This API allows student to unsubscribe from a course giving their student_id and class_id, it should cover the case where student hasn’t subscribed to the class and also check if student_id or class_id exists

Here’s a screenshot of case where student not subscribe to a class



Curl

```
curl -X 'DELETE' \
  'http://localhost:5000/notifications/unsubscribe?student_id=0001&class_id=0002' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:5000/notifications/unsubscribe?student_id=0001&class_id=0002
```

Server response

Code Details

404 *Undocumented* Error: Not Found

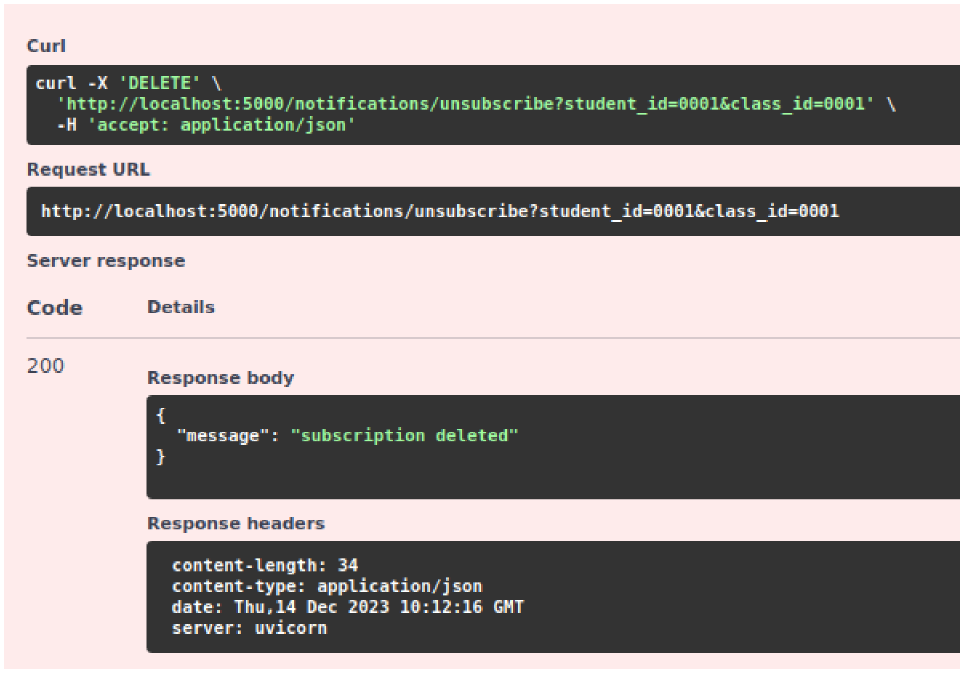
Response body

```
{
  "detail": "No subscription found"
}
```

Response headers

```
 content-length: 34
 content-type: application/json
 date: Thu,14 Dec 2023 10:26:21 GMT
 server: uvicorn
```

Here’s screenshot showing that student able to unsubscribe from a class



Curl

```
curl -X 'DELETE' \
  'http://localhost:5000/notifications/unsubscribe?student_id=0001&class_id=0001' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:5000/notifications/unsubscribe?student_id=0001&class_id=0001
```

Server response

Code Details

200

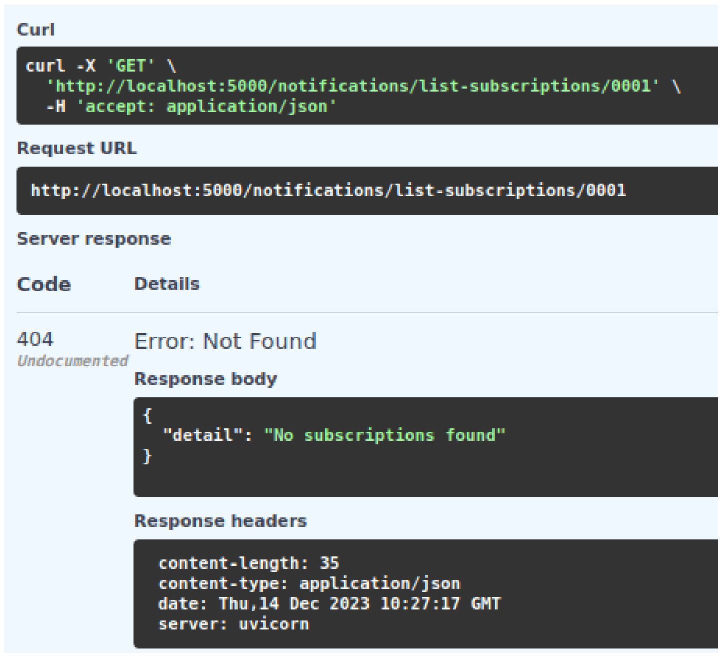
Response body

```
{
  "message": "subscription deleted"
}
```

Response headers

```
 content-length: 34
 content-type: application/json
 date: Thu,14 Dec 2023 10:12:16 GMT
 server: uvicorn
```

View current subscription to see if student’s unsubscribed successfully


Curl

```
curl -X 'GET' \
  'http://localhost:5000/notifications/list-subscriptions/0001' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:5000/notifications/list-subscriptions/0001
```

Server response

Code Details

404 *Undocumented* Error: Not Found

Response body

```
{
  "detail": "No subscriptions found"
}
```

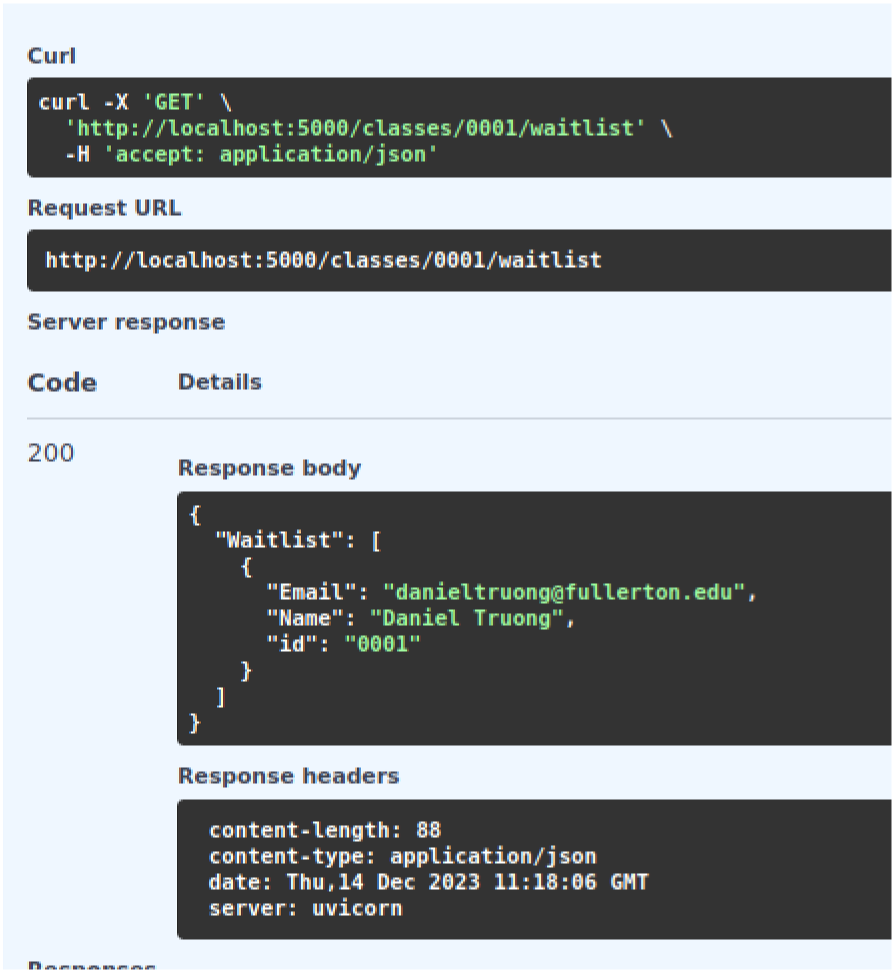
Response headers

```
  content-length: 35
  content-type: application/json
  date: Thu,14 Dec 2023 10:27:17 GMT
  server: uvicorn
```

Task 2: Producing enrollment notifications

Objective: The objective of this task is to create a producer that publish exchange whenever first student in waitlist get auto enrolled due to some student drop the class

Here’s screenshot of student who’s currently waitlisted



Curl

```
curl -X 'GET' \
  'http://localhost:5000/classes/0001/waitlist' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:5000/classes/0001/waitlist
```

Server response

Code Details

200

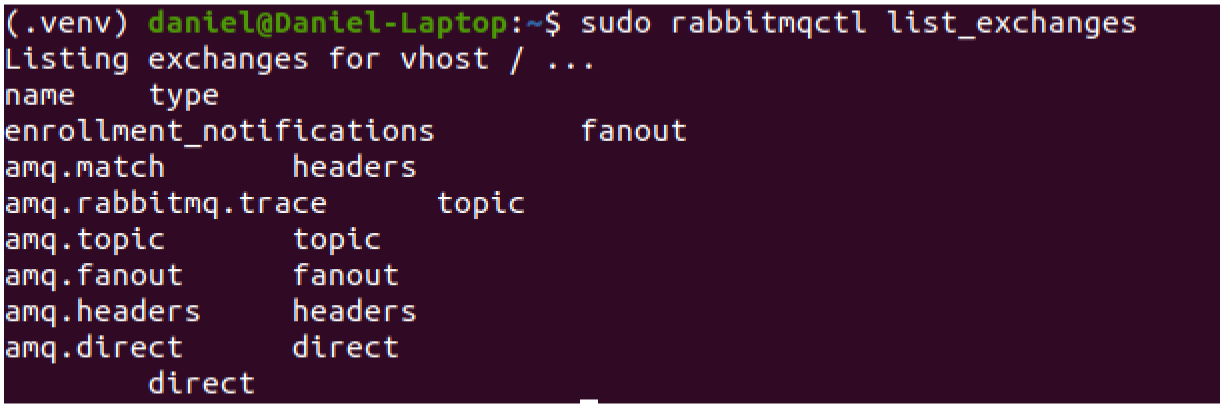
Response body

```
{
  "Waitlist": [
    {
      "Email": "danieltruong@fullerton.edu",
      "Name": "Daniel Truong",
      "id": "0001"
    }
  ]
}
```

Response headers

```
content-length: 88
content-type: application/json
date: Thu,14 Dec 2023 11:18:06 GMT
server: uvicorn
```

By dropping student 0002, then student 0001 should be auto enrolled in a class and the exchange will be published via RabbitMQ



```
(.venv) daniel@Daniel-Laptop:~$ sudo rabbitmqctl list_exchanges
Listing exchanges for vhost / ...
name    type
enrollment_notifications        fanout
amq.match       headers
amq.rabbitmq.trace      topic
amq.topic       topic
amq.fanout      fanout
amq.headers     headers
amq.direct      direct
        direct
```

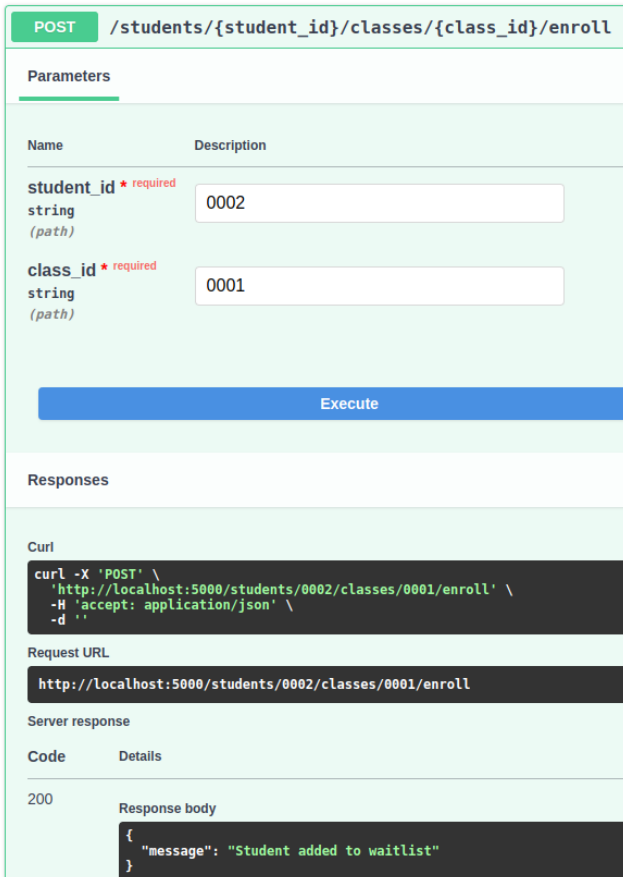
Task 3: Consuming enrollment notifications

Objective:

To create consumers to send an email webhook or both when a student is added to a class from the waitlist.

For the implementation, when a student is added to a class from the waitlist, the producer sends a message to the exchange in RabbitMQ. Now the email and webhook consumers will send an email or trigger a webhook when this message is received.

To test this implementation, let's add student 2 to the waitlist.



Name	Description
student_id * required string (path)	0002
class_id * required string (path)	0001

Curl

```
curl -X 'POST' \
  'http://localhost:5000/students/0002/classes/0001/enroll' \
  -H 'accept: application/json' \
  -d ''
```

Request URL

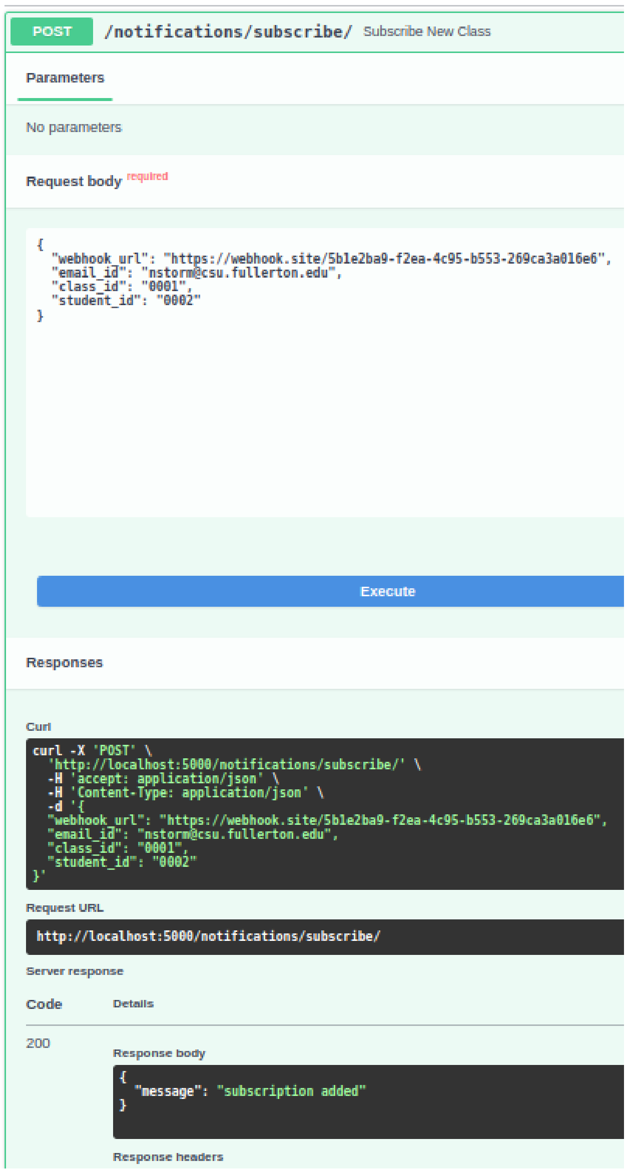
```
http://localhost:5000/students/0002/classes/0001/enroll
```

Server response

Code	Details
200	Response body

```
{
  "message": "Student added to waitlist"
}
```

Now we will subscribe student 0002 to notifications.


POST `/notifications/subscribe/` Subscribe New Class

Parameters

No parameters

Request body required

```
{
  "webhook_url": "https://webhook.site/5b1e2ba9-f2ea-4c95-b553-269ca3a016e6",
  "email_id": "nstorm@csu.fullerton.edu",
  "class_id": "0001",
  "student_id": "0002"
}
```

Execute

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5000/notifications/subscribe/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "webhook_url": "https://webhook.site/5b1e2ba9-f2ea-4c95-b553-269ca3a016e6",
  "email_id": "nstorm@csu.fullerton.edu",
  "class_id": "0001",
  "student_id": "0002"
}'
```

Request URL

```
http://localhost:5000/notifications/subscribe/
```

Server response

Code Details

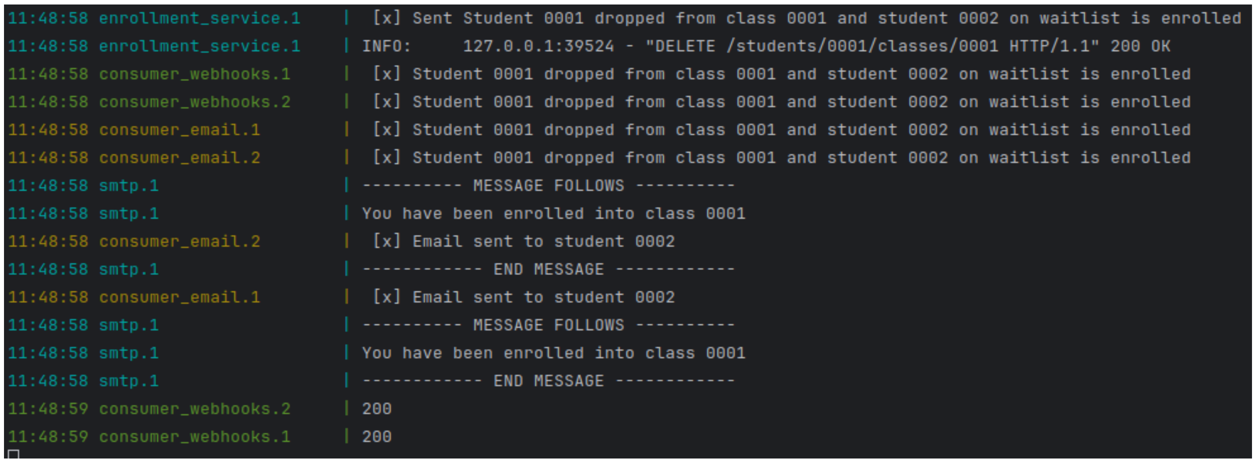
200

Response body

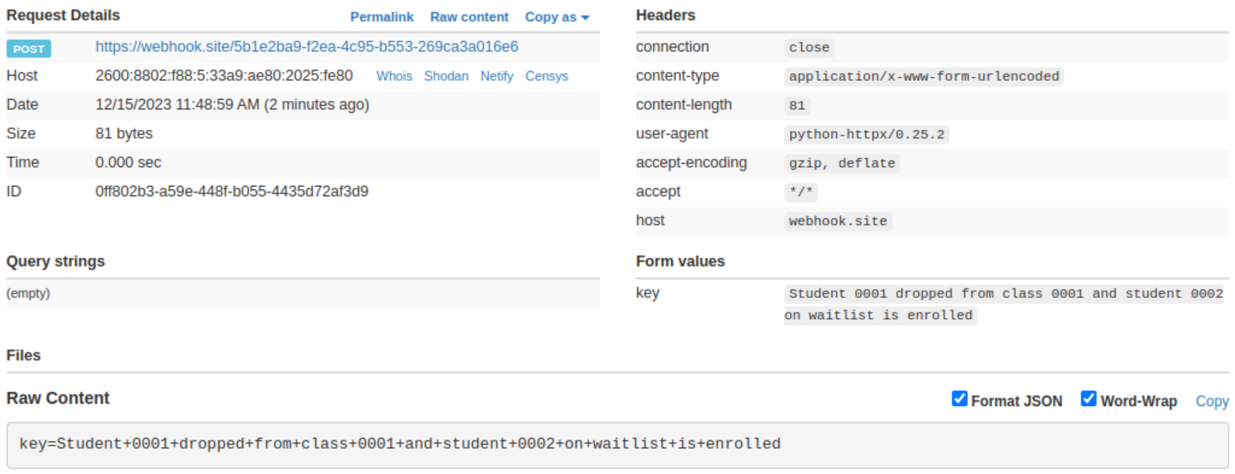
```
{
  "message": "subscription added"
}
```

Response headers

The webhook url used is a free site to monitor webhook calls. Now we will remove student 0001 from the class and student 0002 will be added to the class from the waitlist. This is how our profile output looks after doing that.



```
11:48:58 enrollment_service.1    |  [x] Sent Student 0001 dropped from class 0001 and student 0002 on waitlist is enrolled
11:48:58 enrollment_service.1    | INFO:     127.0.0.1:39524 - "DELETE /students/0001/classes/0001 HTTP/1.1" 200 OK
11:48:58 consumer_webhooks.1     |  [x] Student 0001 dropped from class 0001 and student 0002 on waitlist is enrolled
11:48:58 consumer_webhooks.2     |  [x] Student 0001 dropped from class 0001 and student 0002 on waitlist is enrolled
11:48:58 consumer_email.1        |  [x] Student 0001 dropped from class 0001 and student 0002 on waitlist is enrolled
11:48:58 consumer_email.2        |  [x] Student 0001 dropped from class 0001 and student 0002 on waitlist is enrolled
11:48:58 smtp.1                  | ---------- MESSAGE FOLLOWS ----------
11:48:58 smtp.1                  | You have been enrolled into class 0001
11:48:58 consumer_email.2        |  [x] Email sent to student 0002
11:48:58 smtp.1                  | ------------ END MESSAGE ------------
11:48:58 consumer_email.1        |  [x] Email sent to student 0002
11:48:58 smtp.1                  | ---------- MESSAGE FOLLOWS ----------
11:48:58 smtp.1                  | You have been enrolled into class 0001
11:48:58 smtp.1                  | ------------ END MESSAGE ------------
11:48:59 consumer_webhooks.2     | 200
11:48:59 consumer_webhooks.1     | 200
```



Request Details Permalink Raw content Copy as

POST	https://webhook.site/5b1e2ba9-f2ea-4c95-b553-269ca3a016e6
Host	2600:8802:f88:5:33a9:ae80:2025:fe80 Whois Shodan Netify Censys
Date	12/15/2023 11:48:59 AM (2 minutes ago)
Size	81 bytes
Time	0.000 sec
ID	0ff802b3-a59e-448f-b055-4435d72af3d9

Headers

connection	close
content-type	application/x-www-form-urlencoded
content-length	81
user-agent	python-httpx/0.25.2
accept-encoding	gzip, deflate
accept	*/*
host	webhook.site

Query strings

(empty)

Form values

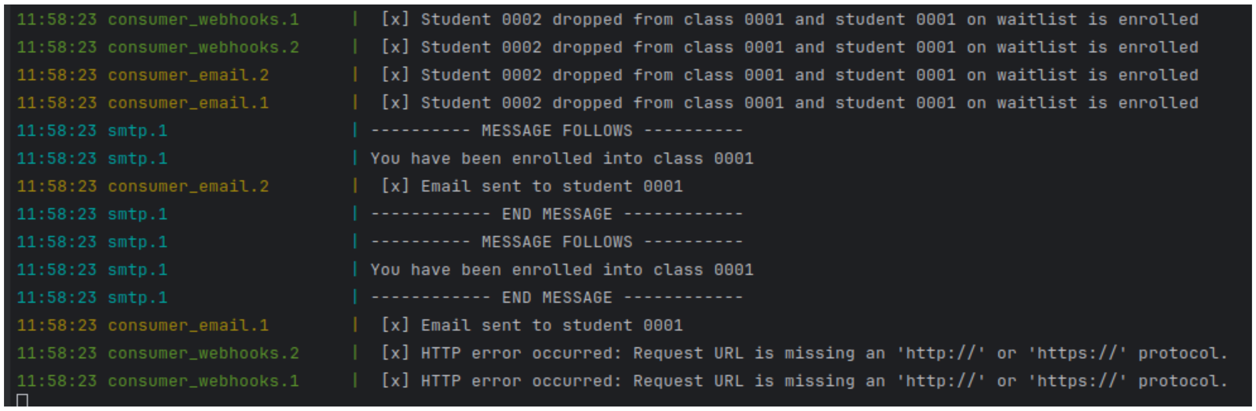
key	Student 0001 dropped from class 0001 and student 0002 on waitlist is enrolled

Files

Raw Content ☑ Format JSON ☑ Word-Wrap Copy

```
key=Student+0001+dropped+from+class+0001+and+student+0002+on+waitlist+is+enrolled
```

As you can see, there are two consumers of each type running and also the SMTP mail server is running. The consumer emails let us know the email was sent to student 0002 and you can see the email server received 2 messages since we have 2 email consumers running. The consumer webhooks are both returning the status code from the POST request. The output of the webhook url site is above, showing that the site received our webhook. We have also implemented error handling if the webhook url is invalid and will still acknowledge the message.



```
11:58:23 consumer_webhooks.1     |  [x] Student 0002 dropped from class 0001 and student 0001 on waitlist is enrolled
11:58:23 consumer_webhooks.2     |  [x] Student 0002 dropped from class 0001 and student 0001 on waitlist is enrolled
11:58:23 consumer_email.2        |  [x] Student 0002 dropped from class 0001 and student 0001 on waitlist is enrolled
11:58:23 consumer_email.1        |  [x] Student 0002 dropped from class 0001 and student 0001 on waitlist is enrolled
11:58:23 smtp.1                  | ---------- MESSAGE FOLLOWS ----------
11:58:23 smtp.1                  | You have been enrolled into class 0001
11:58:23 consumer_email.2        |  [x] Email sent to student 0001
11:58:23 smtp.1                  | ------------ END MESSAGE ------------
11:58:23 smtp.1                  | ---------- MESSAGE FOLLOWS ----------
11:58:23 smtp.1                  | You have been enrolled into class 0001
11:58:23 smtp.1                  | ------------ END MESSAGE ------------
11:58:23 consumer_email.1        |  [x] Email sent to student 0001
11:58:23 consumer_webhooks.2     |  [x] HTTP error occurred: Request URL is missing an 'http://' or 'https://' protocol.
11:58:23 consumer_webhooks.1     |  [x] HTTP error occurred: Request URL is missing an 'http://' or 'https://' protocol.
```

Task 4: Cache waitlist position

Objective:

The objective of this task is to implement HTTP conditional requests to cache the **GET /students/{student_id}/waitlist/{class_id}** that let students view their current waitlist position.

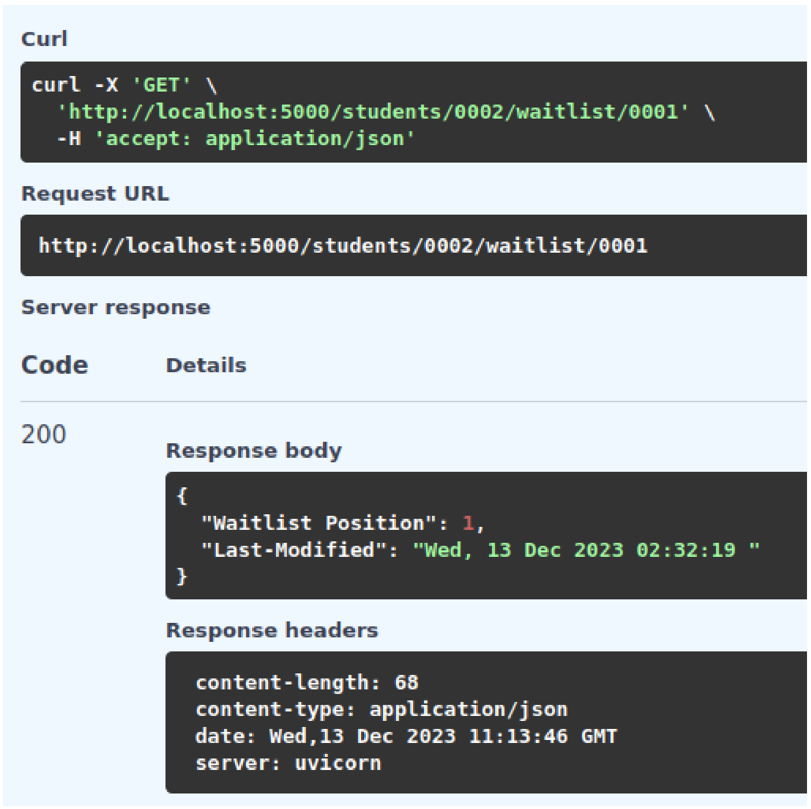
For our implementation, I’ve implemented the Last-Modified HTTP header returned by the server in the response to a client's request, and also added the optional **if_modified_since** header so that if client includes ‘If_modified_since” we can determine whether the resource been modified or not within specific time range to see if we should return a **304 Not Modified status.** If the client doesn’t include ‘if_modified_since’, we’ll assume the version is outdated and provide **200 Status.**

Here’s the code snippet implemented the caching:

```
@router.get("/students/{student_id}/waitlist/{class_id}",
tags=['Waitlist'], summary="Get waitlist position for a student in a
class")
```

```
def view_waiting_list(student_id: str, class_id: str, if_modified_since:
str = Header(None)):

   # check if student exists in the database

   student_data = qh.query_student(dynamodb_client, student_id)

   if not student_data:

       raise HTTPException(status_code=status.HTTP_404_NOT_FOUND,
detail="No student found")

   # check if class exists in the database

   class_data = qh.query_class(dynamodb_client, class_id)

   if not class_data:

       raise HTTPException(status_code=status.HTTP_404_NOT_FOUND,
detail="No class found")

   waitlist_key = f"waitlist:{class_id}"

   if not r.exists(waitlist_key):

       raise HTTPException(status_code=status.HTTP_404_NOT_FOUND,
detail="No waitlist found")

   waitlist_data = r.lrange(waitlist_key, 0, -1)

   if not waitlist_data:

       raise HTTPException(status_code=status.HTTP_404_NOT_FOUND,
detail="No waitlist found")

   # Get last modified time of waitlist using Redis

   last_modified_time = r.lastsave()

   if if_modified_since:

       client_modified_time =
datetime.datetime.strptime(if_modified_since, "%a, %d %b %Y %H:%M:%S %Z")

       if client_modified_time >= last_modified_time:

           return status.HTTP_304_NOT_MODIFIED
   id = f"s#{student_id}".encode('utf-8')

   if id not in waitlist_data:

       raise HTTPException(status_code=status.HTTP_404_NOT_FOUND,
detail="Student is not on waitlist")

   # Get student's position on waitlist

   position = waitlist_data.index(id) + 1

   # Convert last modified time to string

   last_modified_str = last_modified_time.strftime("%a, %d %b %Y %H:%M:%S
%Z")

   return {"Waitlist Position": position, "Last-Modified":
last_modified_str}
```

Here's screenshot if client's request doesn't include if_modified_since header:



Curl

```
curl -X 'GET' \
  'http://localhost:5000/students/0002/waitlist/0001' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:5000/students/0002/waitlist/0001
```

Server response

Code Details

200

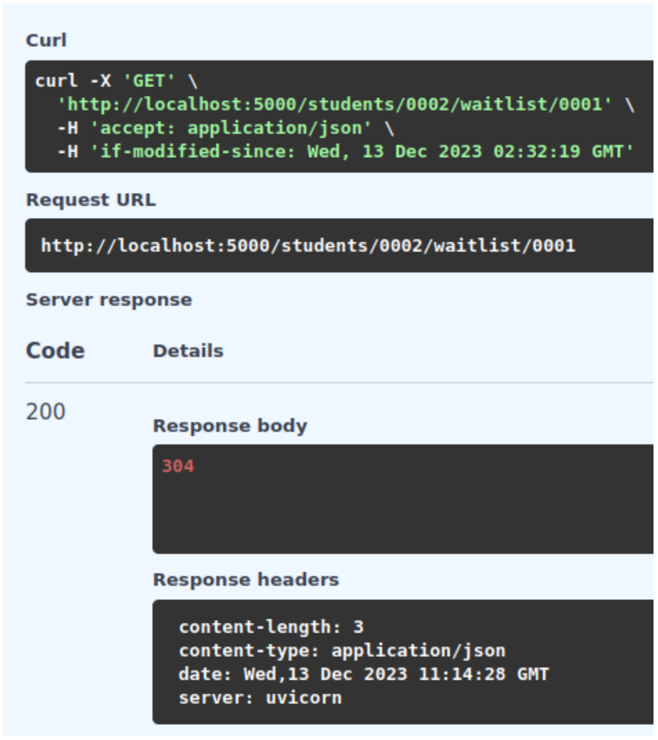
Response body

```
{
  "Waitlist Position": 1,
  "Last-Modified": "Wed, 13 Dec 2023 02:32:19 "
}
```

Response headers

```
 content-length: 68
 content-type: application/json
 date: Wed,13 Dec 2023 11:13:46 GMT
 server: uvicorn
```

Here's screenshot if client's request include if_modified_since header and cached:



Curl

```
curl -X 'GET' \
  'http://localhost:5000/students/0002/waitlist/0001' \
  -H 'accept: application/json' \
  -H 'if-modified-since: Wed, 13 Dec 2023 02:32:19 GMT'
```

Request URL

```
http://localhost:5000/students/0002/waitlist/0001
```

Server response

Code Details

200

Response body

```
304
```

Response headers

```
 content-length: 3
 content-type: application/json
 date: Wed,13 Dec 2023 11:14:28 GMT
 server: uvicorn
```