Fall 2023, CPSC 449, Section 1

Project 3

Edwin Peraza

Micah Baumann

Vivian Cao

Liam Hernandez

Gaurav Warad

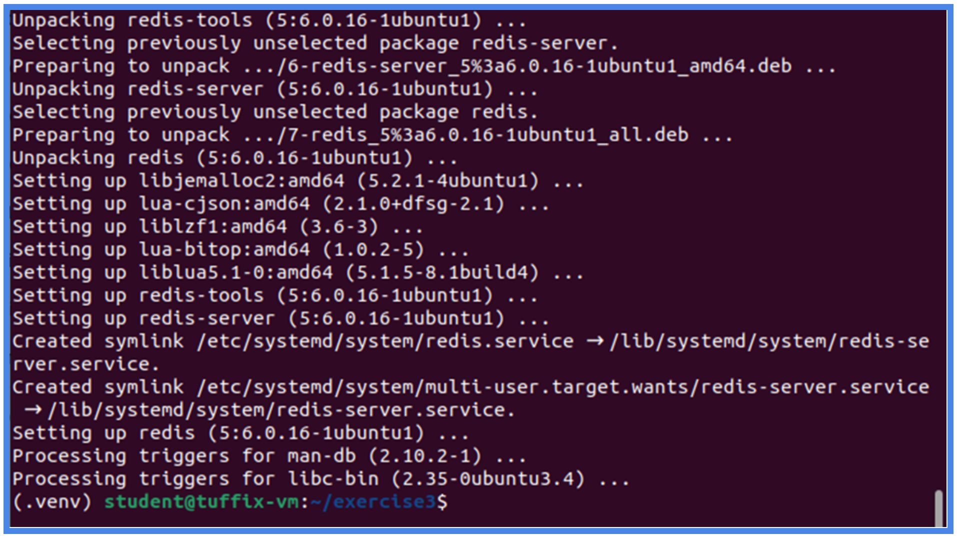
Task 1: Install and configure databases, tools, and libraries

We referenced the necessary commands from exercise 3 to install redis and necessary components.

Install Redis and its Python client libraries with the commands:

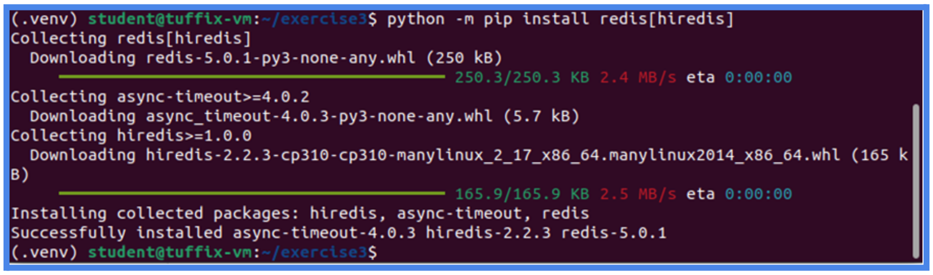
sudo apt update

sudo apt install --yes redis



```
Unpacking redis-tools (5:6.0.16-1ubuntu1) ...
Selecting previously unselected package redis-server.
Preparing to unpack .../6-redis-server_5%3a6.0.16-1ubuntu1_amd64.deb ...
Unpacking redis-server (5:6.0.16-1ubuntu1) ...
Selecting previously unselected package redis.
Preparing to unpack .../7-redis_5%3a6.0.16-1ubuntu1_all.deb ...
Unpacking redis (5:6.0.16-1ubuntu1) ...
Setting up libjemalloc2:amd64 (5.2.1-4ubuntu1) ...
Setting up lua-cjson:amd64 (2.1.0+dfsg-2.1) ...
Setting up liblzf1:amd64 (3.6-3) ...
Setting up lua-bitop:amd64 (1.0.2-5) ...
Setting up liblua5.1-0:amd64 (5.1.5-8.1build4) ...
Setting up redis-tools (5:6.0.16-1ubuntu1) ...
Setting up redis-server (5:6.0.16-1ubuntu1) ...
Created symlink /etc/systemd/system/redis.service → /lib/systemd/system/redis-se
rver.service.
Created symlink /etc/systemd/system/multi-user.target.wants/redis-server.service
 → /lib/systemd/system/redis-server.service.
Setting up redis (5:6.0.16-1ubuntu1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.4) ...
(.venv) student@tuffix-vm:~/exercise3$
```

python -m pip install redis[hiredis]



```
(.venv) student@tuffix-vm:~/exercise3$ python -m pip install redis[hiredis]
Collecting redis[hiredis]
  Downloading redis-5.0.1-py3-none-any.whl (250 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 250.3/250.3 KB 2.4 MB/s eta 0:00:00
Collecting async-timeout>=4.0.2
  Downloading async_timeout-4.0.3-py3-none-any.whl (5.7 kB)
Collecting hiredis>=1.0.0
  Downloading hiredis-2.2.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (165 k
B)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 165.9/165.9 KB 2.5 MB/s eta 0:00:00
Installing collected packages: hiredis, async-timeout, redis
Successfully installed async-timeout-4.0.3 hiredis-2.2.3 redis-5.0.1
(.venv) student@tuffix-vm:~/exercise3$
```

Install and configure the AWS CLI

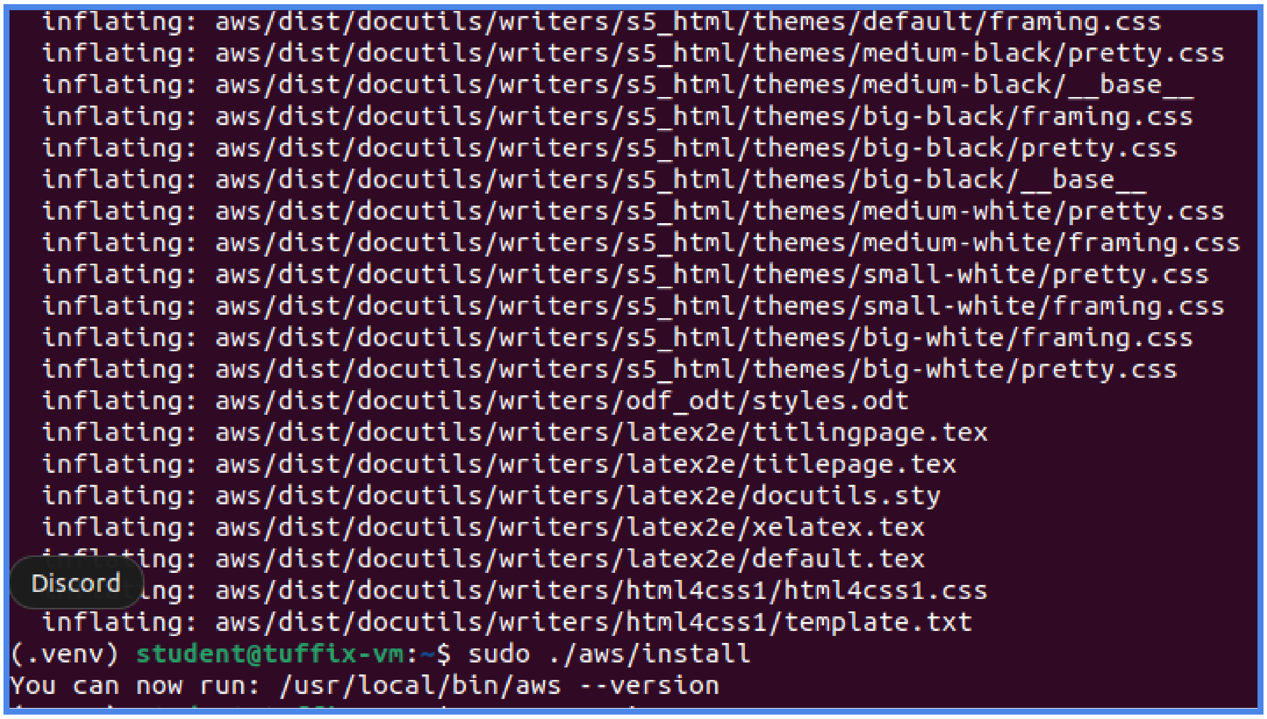
We referenced AWS User Guide and AWS CLI Command Guide to complete this project requirement.

The following commands were used to install the AWS CLI:

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

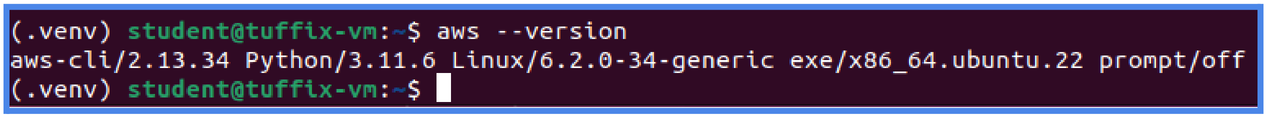
unzip awscliv2.zip

sudo ./aws/install



```
  inflating: aws/dist/docutils/writers/s5_html/themes/default/framing.css
  inflating: aws/dist/docutils/writers/s5_html/themes/medium-black/pretty.css
  inflating: aws/dist/docutils/writers/s5_html/themes/medium-black/__base__
  inflating: aws/dist/docutils/writers/s5_html/themes/big-black/framing.css
  inflating: aws/dist/docutils/writers/s5_html/themes/big-black/pretty.css
  inflating: aws/dist/docutils/writers/s5_html/themes/big-black/__base__
  inflating: aws/dist/docutils/writers/s5_html/themes/medium-white/pretty.css
  inflating: aws/dist/docutils/writers/s5_html/themes/medium-white/framing.css
  inflating: aws/dist/docutils/writers/s5_html/themes/small-white/pretty.css
  inflating: aws/dist/docutils/writers/s5_html/themes/small-white/framing.css
  inflating: aws/dist/docutils/writers/s5_html/themes/big-white/framing.css
  inflating: aws/dist/docutils/writers/s5_html/themes/big-white/pretty.css
  inflating: aws/dist/docutils/writers/odf_odt/styles.odt
  inflating: aws/dist/docutils/writers/latex2e/titlingpage.tex
  inflating: aws/dist/docutils/writers/latex2e/titlepage.tex
  inflating: aws/dist/docutils/writers/latex2e/docutils.sty
  inflating: aws/dist/docutils/writers/latex2e/xelatex.tex
  inflating: aws/dist/docutils/writers/latex2e/default.tex
  inflating: aws/dist/docutils/writers/html4css1/html4css1.css
  inflating: aws/dist/docutils/writers/html4css1/template.txt
(.venv) student@tuffix-vm:~$ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
```

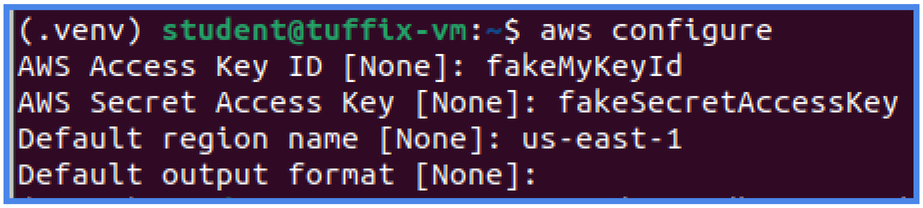
We ran the “aws –version” command to verify that the AWS CLI has been properly installed.



```
(.venv) student@tuffix-vm:~$ aws --version
aws-cli/2.13.34 Python/3.11.6 Linux/6.2.0-34-generic exe/x86_64.ubuntu.22 prompt/off
(.venv) student@tuffix-vm:~$ 
```

After that we followed the instructions on the Long Term credentials tab

Using the command “aws configure”, we configured some dummy credentials to use DynamoDB local. The aws configure command sets up the AWS Command Line Interface (CLI) with the necessary credentials and default region.



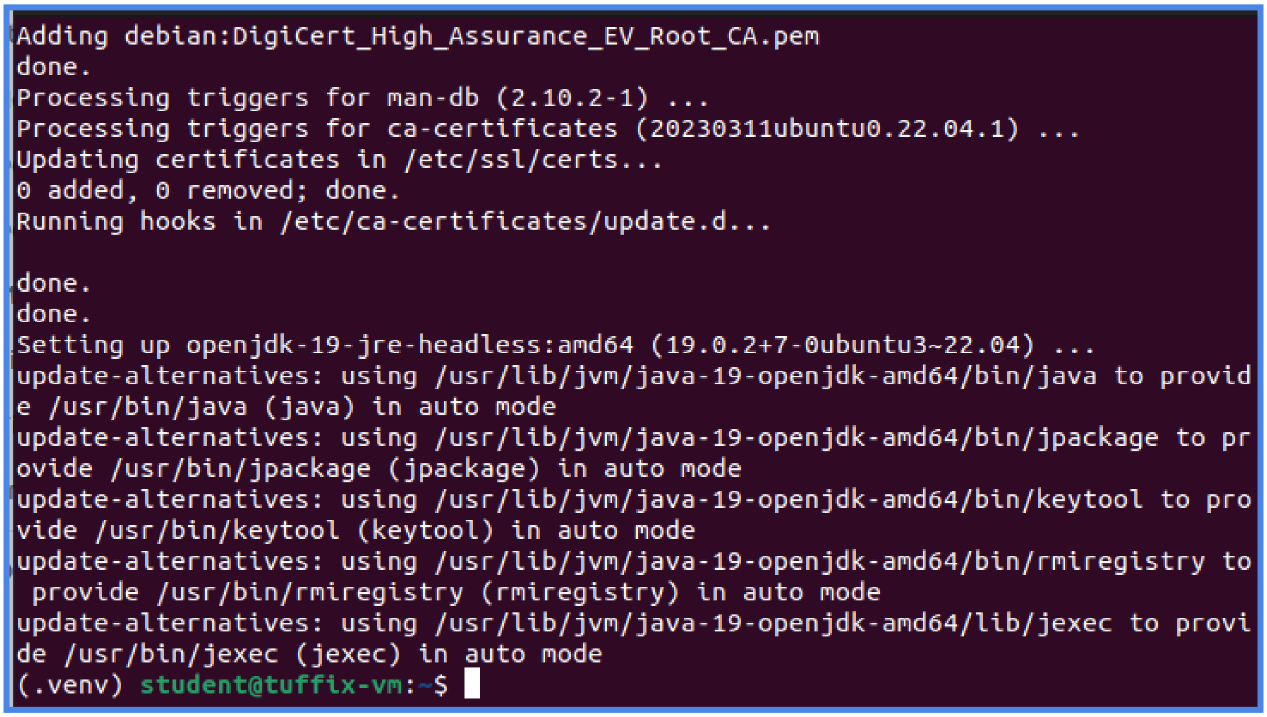
```
(.venv) student@tuffix-vm:~$ aws configure
AWS Access Key ID [None]: fakeMyKeyId
AWS Secret Access Key [None]: fakeSecretAccessKey
Default region name [None]: us-east-1
Default output format [None]:
```

Install and configure Amazon DynamoDB local

DynamoDB local requires a Java Runtime Environment to be installed, we used the following commands:

sudo apt update

sudo apt install --yes openjdk-19-jre-headless



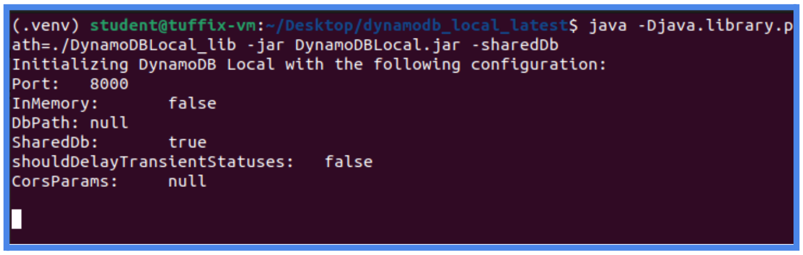
```
Adding debian:DigiCert_High_Assurance_EV_Root_CA.pem
done.
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for ca-certificates (20230311ubuntu0.22.04.1) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...

done.
done.
Setting up openjdk-19-jre-headless:amd64 (19.0.2+7-0ubuntu3~22.04) ...
update-alternatives: using /usr/lib/jvm/java-19-openjdk-amd64/bin/java to provid
e /usr/bin/java (java) in auto mode
update-alternatives: using /usr/lib/jvm/java-19-openjdk-amd64/bin/jpackage to pr
ovide /usr/bin/jpackage (jpackage) in auto mode
update-alternatives: using /usr/lib/jvm/java-19-openjdk-amd64/bin/keytool to pro
vide /usr/bin/keytool (keytool) in auto mode
update-alternatives: using /usr/lib/jvm/java-19-openjdk-amd64/bin/rmiregistry to
 provide /usr/bin/rmiregistry (rmiregistry) in auto mode
update-alternatives: using /usr/lib/jvm/java-19-openjdk-amd64/lib/jexec to provi
de /usr/bin/jexec (jexec) in auto mode
(.venv) student@tuffix-vm:~$ 
```

With a Java Runtime Environment we are ready to run DynamoDB locally.

First we downloaded DynamoDB local v2.x and extracted the contents of the zip file.

To start DynamoDB, we used the following command in the directory with the DynamoDBLocal.jar file. Command:

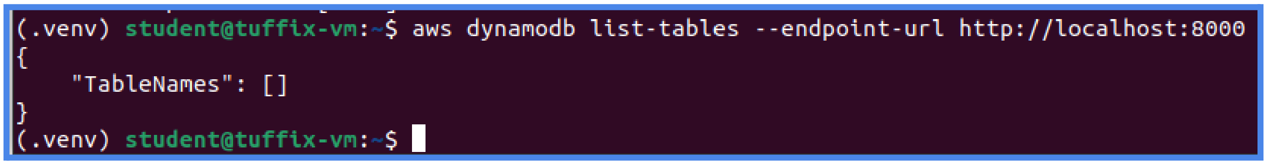
java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar -sharedDb



```
(.venv) student@tuffix-vm:~/Desktop/dynamodb_local_latest$ java -Djava.library.p
ath=./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

```

Now to test that it is working, the command to list DynamoDB tables was used:

aws dynamodb list-tables --endpoint-url http://localhost:8000



```
(.venv) student@tuffix-vm:~$ aws dynamodb list-tables --endpoint-url http://localhost:8000
{
    "TableNames": []
}
(.venv) student@tuffix-vm:~$ 
```

We added the dynamodb_local process to our Procfile:

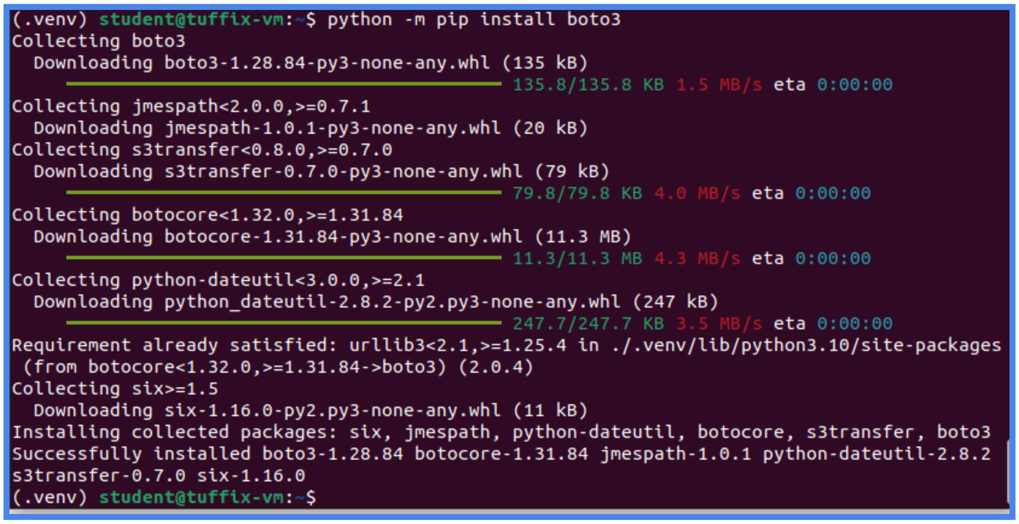
```
users_primary: ./bin/litefs mount -config ./users/etc/primary.yml
users_secondary_1: ./bin/litefs mount -config ./users/etc/secondary_1.yml
users_secondary_2: ./bin/litefs mount -config ./users/etc/secondary_2.yml
enroll: uvicorn --port $PORT enroll.api:app --reload
krakend: echo krakend.json | entr -nrz krakend run --port $PORT --config krakend.json
dynamodb_local: java -Djava.library.path=./bin/DynamoDBLocal_lib -jar ./bin/DynamoDBLocal.jar -sharedDb -port $PORT
```

By adding the process, we start a local instance of DynamoDB, with the ‘sharedDB’ flag indicating that a single database file is shared by all clients.

Install the AWS SDK for Python

We used the following command to install the latest version of the AWS SDK for Python (Boto3):

python -m pip install boto3



```
(.venv) student@tuffix-vm:~$ python -m pip install boto3
Collecting boto3
  Downloading boto3-1.28.84-py3-none-any.whl (135 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 135.8/135.8 KB 1.5 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 s3transfer<0.8.0,>=0.7.0
  Downloading s3transfer-0.7.0-py3-none-any.whl (79 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.8/79.8 KB 4.0 MB/s eta 0:00:00
Collecting botocore<1.32.0,>=1.31.84
  Downloading botocore-1.31.84-py3-none-any.whl (11.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.3/11.3 MB 4.3 MB/s eta 0:00:00
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 3.5 MB/s eta 0:00:00
Requirement already satisfied: urllib3<2.1,>=1.25.4 in ./.venv/lib/python3.10/site-packages
 (from botocore<1.32.0,>=1.31.84->boto3) (2.0.4)
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, boto3
Successfully installed boto3-1.28.84 botocore-1.31.84 jmespath-1.0.1 python-dateutil-2.8.2
s3transfer-0.7.0 six-1.16.0
(.venv) student@tuffix-vm:~$
```

Task 2: Partition the data for the enrollment service

We are using redis to maintain the waiting lists, while information and classes are stored in DynamoDB Local.

1. Storing classes and enrollment data on DynamoDB Local

The DynamoDB database was created with the python script Catalog.py. The file employs a class that creates tables based on the table name, key schema, attribute definitions, and global indexes. The class also has a method to delete any previous table with a given name. Additionally, there is a method to populate the tables with the passed records.

```
   """Creates tables for the catalog database"""
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by
'create_table'.
        self.table = None

    def create_table(self, table_name, key_schema, attribute_definitions,
global_secondary_indexes):
        """
        Creates an Amazon DynamoDB table for the catalog database.

        :param table_name: The name of the table to create.
        :return: The newly created table.
        """
        try:
            self.table = self.dyn_resource.create_table(
                TableName = table_name,
                KeySchema = key_schema,
                AttributeDefinitions= attribute_definitions,
                ProvisionedThroughput={
                    "ReadCapacityUnits": 10,
                    "WriteCapacityUnits": 10,
                },
                GlobalSecondaryIndexes=global_secondary_indexes
            )
            self.table.wait_until_exists()
            print(f"Table {table_name} created successfully.")
        except ClientError as err:
            print(
                "Couldn't create table {}. Here's why: {}: {}".format(
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            )
            raise
        else:
            return self.table

    def put_items(self, table_name, items):
        """
        Adds items to the specified DynamoDB table.

        :param table_name: The name of the table to add items to.
        :param items: A list of dictionaries, where each dictionary
represents an item to add.
        """
        table = self.dyn_resource.Table(table_name)
        for item in items:
            try:
                table.put_item(Item=item)
            except ClientError as e:
                print(f"Error adding item to {table_name}:
{e.response['Error']['Message']}")
                raise e

    def delete_table_if_exists(self, table_name):
        """
        Deletes the specified DynamoDB table if it exists.

        :param table_name: The name of the table to delete.
        """
        try:
            table = self.dyn_resource.Table(table_name)
            if table.table_status == 'ACTIVE':
                table.delete()
                table.wait_until_not_exists()
                print(f"Table {table_name} deleted successfully.")
            else:
                print(f"Table {table_name} does not exist.")
        except ClientError as e:
            if e.response['Error']['Code'] == 'ResourceNotFoundException':
                print(f"Table {table_name} does not exist.")
            else:
                raise
```

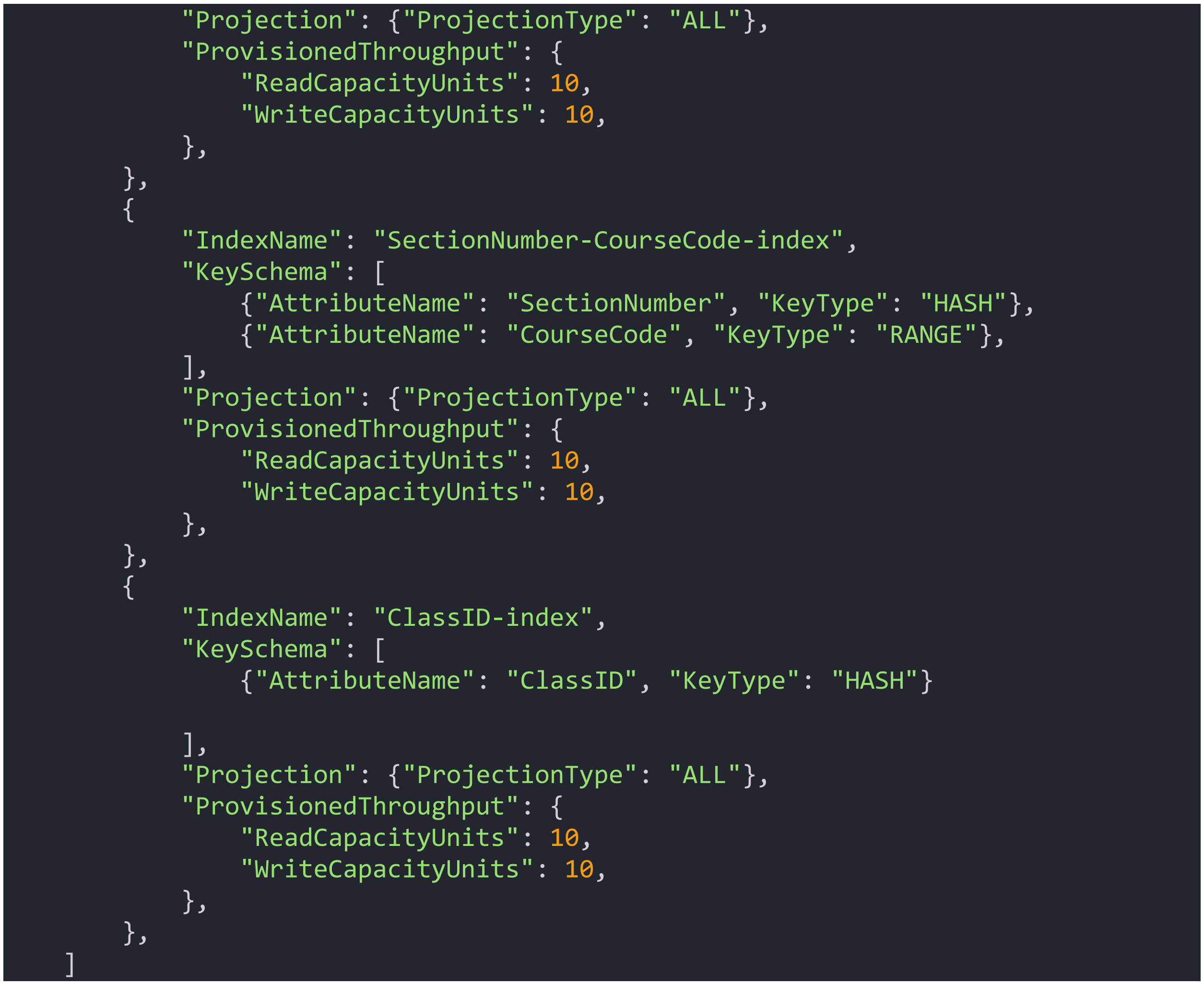
The “Users” table stores basic information about the users and this information is duplicated from the Sqlite3 database for the users service.

```
# Define the key schema and attribute definitions for the "Users" table
users_key_schema = [
    {"AttributeName": "UserId", "KeyType": "HASH"}
]

users_attribute_definitions = [
    {"AttributeName": "UserId", "AttributeType": "N"},
    {"AttributeName": "Email", "AttributeType": "S"}
]

classes_global_secondary_indexes = [
        {
            "IndexName": "Email-index",
            "KeySchema": [
                {"AttributeName": "Email", "KeyType": "HASH"},
                {"AttributeName": "UserId", "KeyType": "RANGE"},
            ],
            "Projection": {"ProjectionType": "ALL"},
            "ProvisionedThroughput": {
                "ReadCapacityUnits": 10,
                "WriteCapacityUnits": 10,
            },
        },
    ]

# Create the "Users" table
my_catalog.create_table("Users", users_key_schema, users_attribute_definitions,
classes_global_secondary_indexes)
```

The “Classes” table stores all the basic information about a class including the course code, section number, class name, department, current enrollment, and max enrollment. On top of that, it includes a flag to determine if the class is still active or if it has been set as inactive by a registrar.

Several indexes are created to query the table with different parameters.

```
# Define the key schema and attribute definitions for the "Classes" table
classes_key_schema = [
    {"AttributeName": "ClassID", "KeyType": "HASH"}
]

classes_attribute_definitions = [
    {"AttributeName": "ClassID", "AttributeType": "N"},
    {"AttributeName": "CourseCode", "AttributeType": "S"},
    {"AttributeName": "SectionNumber", "AttributeType": "N"},
    {"AttributeName": "State", "AttributeType": "S"}
]

classes_global_secondary_indexes = [
        {
            "IndexName": "State-index",
            "KeySchema": [
                {"AttributeName": "State", "KeyType": "HASH"},
                {"AttributeName": "ClassID", "KeyType": "RANGE"},
            ],
```



```
            "Projection": {"ProjectionType": "ALL"},
            "ProvisionedThroughput": {
                "ReadCapacityUnits": 10,
                "WriteCapacityUnits": 10,
            },
        },
        {
            "IndexName": "SectionNumber-CourseCode-index",
            "KeySchema": [
                {"AttributeName": "SectionNumber", "KeyType": "HASH"},
                {"AttributeName": "CourseCode", "KeyType": "RANGE"},
            ],
            "Projection": {"ProjectionType": "ALL"},
            "ProvisionedThroughput": {
                "ReadCapacityUnits": 10,
                "WriteCapacityUnits": 10,
            },
        },
        {
            "IndexName": "ClassID-index",
            "KeySchema": [
                {"AttributeName": "ClassID", "KeyType": "HASH"}

            ],
            "Projection": {"ProjectionType": "ALL"},
            "ProvisionedThroughput": {
                "ReadCapacityUnits": 10,
                "WriteCapacityUnits": 10,
            },
        },
    ]
```

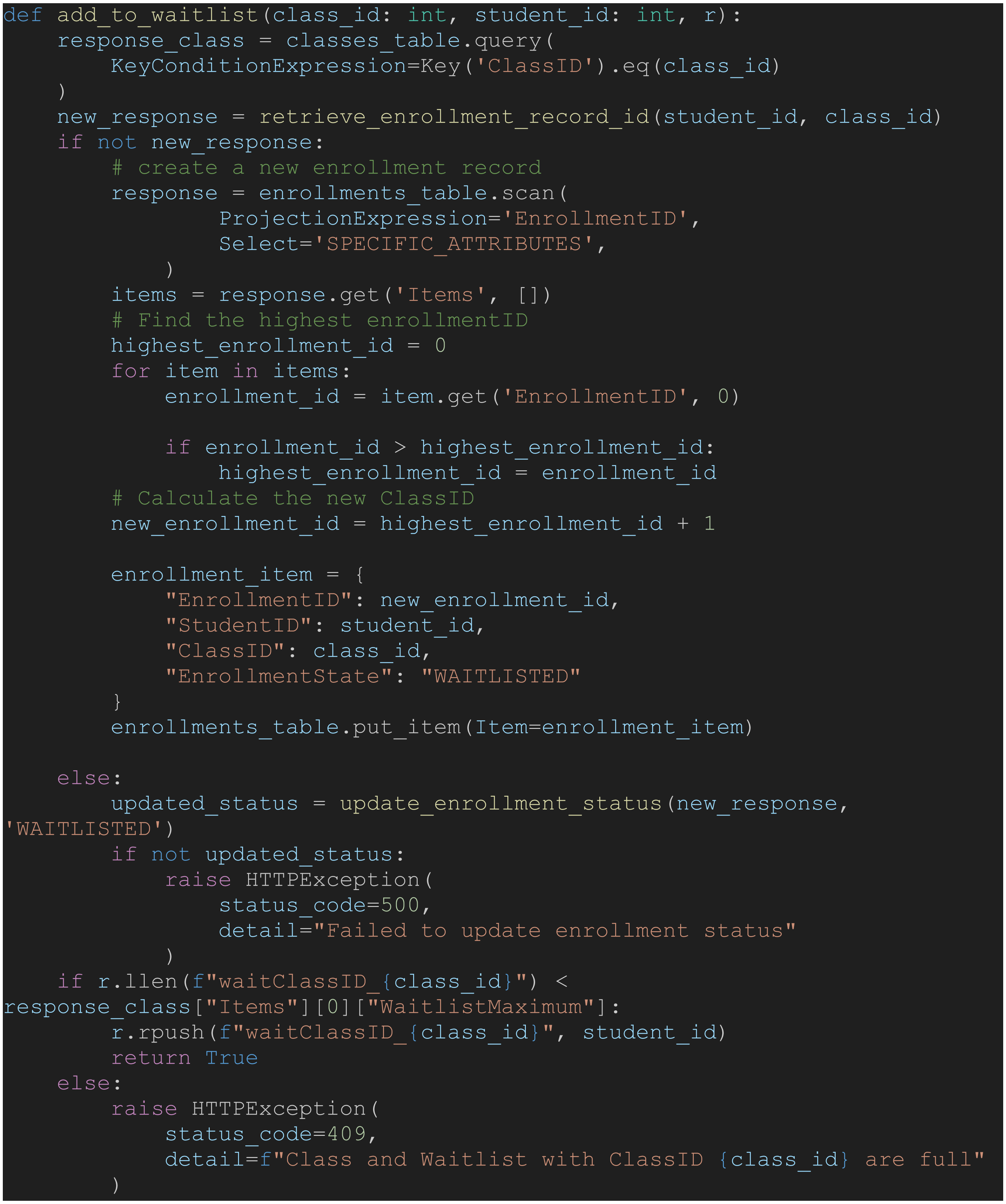
The “Enrollments” table stores every enrollment made in the enrollment service. It associates a student ID with an enrollment and sets an status for that enrollment which can be either “DROPPED”, “ENROLLED”. or “WAITLISTED”. The table includes several indexes to query the table with different parameters.

The tables are pre populated with the function “put_items” in the catalog class for demonstration purposes.

All the endpoints in the enrollment service have been updated to utilize DynamoDB instead of SQLite3.

2. Maintaining Waiting Lists with Redis:

The add_to_waitlist function adds a student to the Redis waitlist for the class if the waitlist is not full, then changes their status in the DynamoDB database.



```
def add_to_waitlist(class_id: int, student_id: int, r):
    response_class = classes_table.query(
        KeyConditionExpression=Key('ClassID').eq(class_id)
    )
    new_response = retrieve_enrollment_record_id(student_id, class_id)
    if not new_response:
        # create a new enrollment record
        response = enrollments_table.scan(
                ProjectionExpression='EnrollmentID',
                Select='SPECIFIC_ATTRIBUTES',
            )
        items = response.get('Items', [])
        # Find the highest enrollmentID
        highest_enrollment_id = 0
        for item in items:
            enrollment_id = item.get('EnrollmentID', 0)

            if enrollment_id > highest_enrollment_id:
                highest_enrollment_id = enrollment_id
        # Calculate the new ClassID
        new_enrollment_id = highest_enrollment_id + 1

        enrollment_item = {
            "EnrollmentID": new_enrollment_id,
            "StudentID": student_id,
            "ClassID": class_id,
            "EnrollmentState": "WAITLISTED"
        }
        enrollments_table.put_item(Item=enrollment_item)

    else:
        updated_status = update_enrollment_status(new_response,
'WAITLISTED')
        if not updated_status:
            raise HTTPException(
                status_code=500,
                detail="Failed to update enrollment status"
            )
    if r.llen(f"waitClassID_{class_id}") <
response_class["Items"][0]["WaitlistMaximum"]:
        r.rpush(f"waitClassID_{class_id}", student_id)
        return True
    else:
        raise HTTPException(
            status_code=409,
            detail=f"Class and Waitlist with ClassID {class_id} are full"
        )
```

When a student drops a class, the next student in the queue is popped from the waitlist and is enrolled in the class. Excerpt from the enrollmentdrop endpoint:

```
            next_on_waitlist = int(r.lpop(f"waitClassID_{classid}"))
            if next_on_waitlist:
                new_status = 'ENROLLED'
                new_response = retrieve_enrollment_record_id(next_on_waitlist, classid)
                new_updated_status = update_enrollment_status(new_response, new_status)
                updated_current_enrollment = update_current_enrollment(classid, increment=True)
```

If the student is on the waitlist, they can drop from the waitlist.

```
@app.delete("/waitlistdrop/{studentid}/{classid}/{username}/{email}")
def remove_student_from_waitlist(studentid: int, classid: int, username: str, email: str, r = Depends(get_redis)):
    """API to drop a class from waitlist.

    Args:
        studentid: The student's ID.
        classid: The class ID.

    Returns:
        A dictionary with a message indicating the student's enrollment status.
    """
    check_user(studentid, username, email)
    status = get_enrollment_status(studentid, classid)
    if status == 'DROPPED':
        raise HTTPException(
            status_code=409,
            detail=f"Student with StudentID {studentid} is already dropped from class with ClassID {classid}"
        )
    if status is None:
        raise HTTPException(
            status_code=404,
            detail=f"Student with StudentID {studentid} is not enrolled in class with ClassID {classid}"
        )
    elif status == 'ENROLLED':
        raise HTTPException(
            status_code=409,
            detail=f"Student with StudentID {studentid} is enrolled in class with
ClassID {classid}"
        )
    if status == 'WAITLISTED':
        new_status = 'DROPPED'
        updated_status =
update_enrollment_status(retrieve_enrollment_record_id(studentid, classid),
new_status)
        if not updated_status:
            raise HTTPException(
                status_code=500,
                detail="Student was not on the waitlist"
            )

        exists = r.lrem(f"waitClassID_{classid}", 0, studentid)
        if exists == 0:
            raise HTTPException(
                status_code=400,
                detail={"Error": "No such student found in the given class on the
waitlist"}
            )

    return {"Element removed": studentid}
```

A student can also check their position on the waitlist.

```
@app.get("/waitlist/{studentid}/{classid}/{username}/{email}")
def view_waitlist_position(studentid: int, classid: int, username: str, email:
str, r = Depends(get_redis)):
    """API to view a student's position on the waitlist.

    Args:
        studentid: The student's ID.
        classid: The class ID.

    Returns:
        A dictionary with a message indicating the student's position on the waitlist.
    """
    check_user(studentid, username, email)
    position = r.lpos(f"waitClassID_{classid}", studentid)

    if position:
        message = f"Student {studentid} is on the waitlist for class {classid} in position"
    else:
        message = f"Student {studentid} is not on the waitlist for class {classid}"
        raise HTTPException(
            status_code=404,
            detail=message,
        )
    return {message: position}
```

Instructors can view all students on a waitlist for their class.

```
@app.get("/instructorwaitlist/{instructorid}/{classid}/{username}/{email}")
def view_waitlist(instructorid: int, classid: int, username: str, email: str, r = Depends(get_redis)):
    """API to view the waitlist for a class.

    Args:
        instructorid: The instructor's ID.

    Returns:
        A dictionary with a list of students on the waitlist for the instructor's classes.
    """
    check_user(instructorid, username, email)
    if not is_instructor_for_class(instructorid, classid):
        raise HTTPException(
            status_code=403,
            detail=f"Instructor with InstructorID {instructorid} is not an instructor for class with ClassID {classid}"
        )
    waitlisted_students = get_students_for_class(classid, 'WAITLISTED')
    if not waitlisted_students:
        raise HTTPException(status_code=404, detail="No waitlisted students found
for this class.")

    student_ids = r.lrange(f"waitClassID_{classid}", 0, -1)
    if not len(student_ids):
        raise HTTPException(status_code=404, detail="No students found in the
waitlist for this class")
    return {"Waitlist": [{"student_id": int(student)} for student in
student_ids]}
```

Task 3: Testing

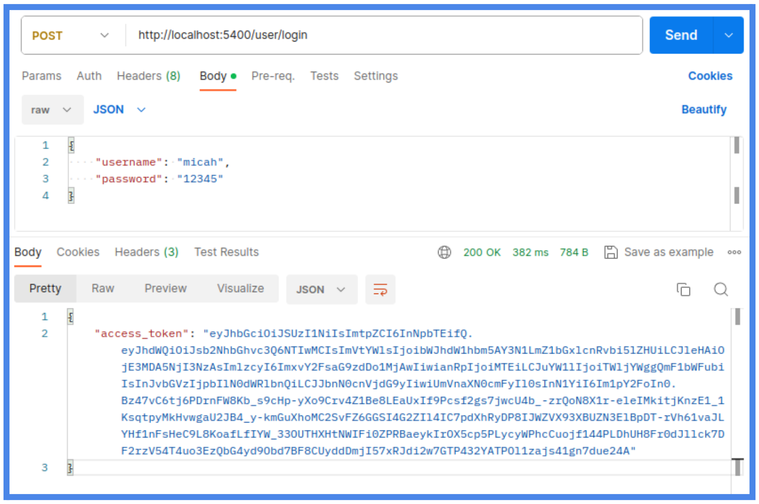
Each of our enrollment API endpoints continues to function after the SQLite database has been removed.

To facilitate testing, begin by utilizing the login endpoint with the provided credentials. Use the username 'micah' and password '12345' to obtain an access token that encompasses all the required roles for the subsequent endpoint.

KrakenD should be running in port 5400.

LOGIN ENDPOINT: /user/login

This endpoint is used to authenticate an user.



http://localhost:5400/user/login

```
{
    "username": "micah",
    "password": "12345"
}
```

200 OK 382 ms 784 B

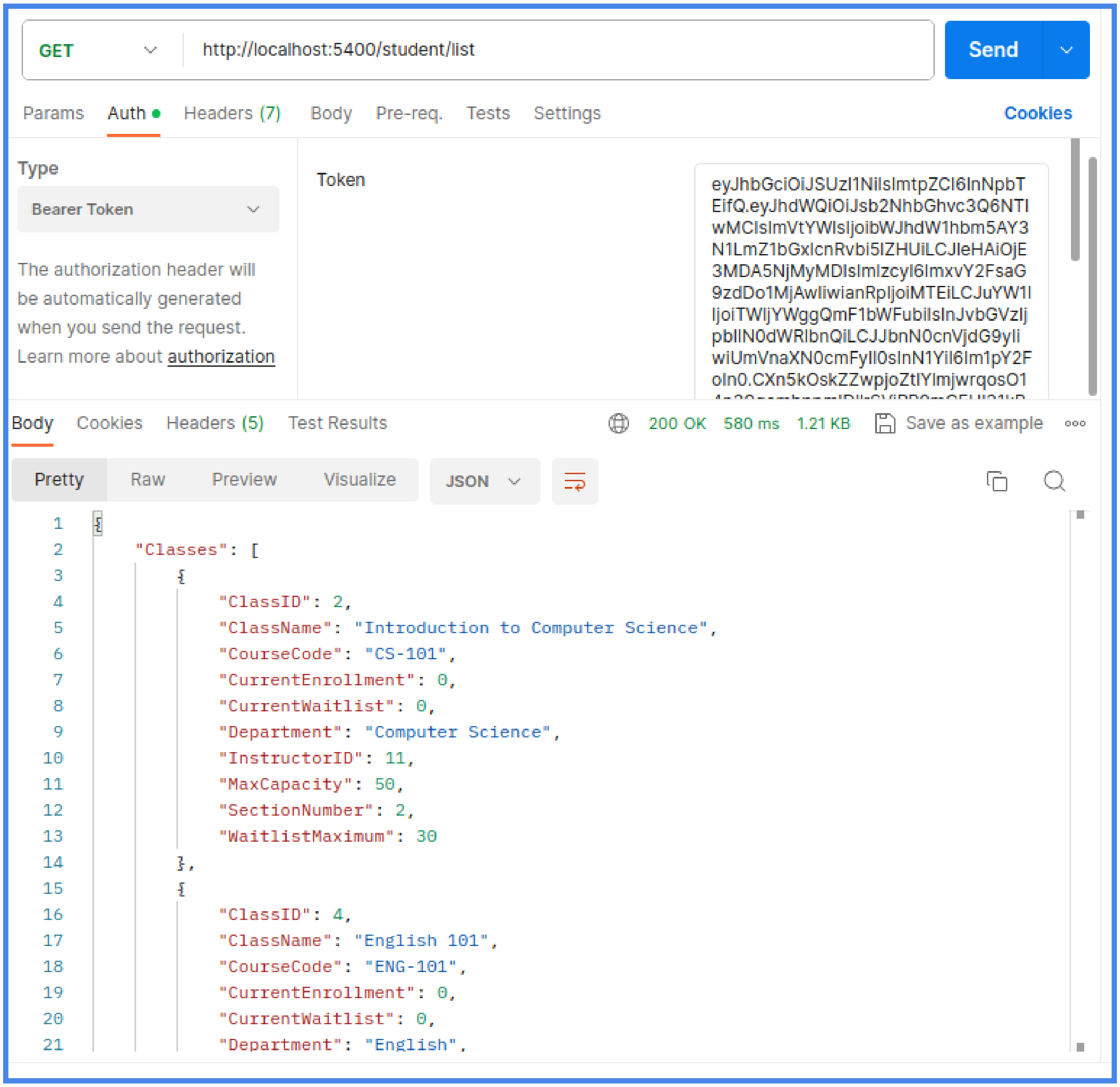
```
{
    "access_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6InNpbTEifQ.
        eyJhdWQiOiJsb2NhbGhvc3Q6NTIwMCIsImVtYWlsIjoibWJhdW1hbm5AY3N1LmZ1bGxlcnRvbi5lZHUiLCJleHAiO
        jE3MDA5NjI3NzAsImlzcyI6ImxvY2FsaG9zdDo1MjAwIiwianRpIjoiMTEiLCJuYW1lIjoiTWljYWggQmF1bWFubi
        IsInJvbGVzIjpbIlN0dWRlbnQiLCJJbnN0cnVjdG9yIiwiUmVnaXN0cmFyIl0sInN1YiI6Im1pY2FoIn0.
        Bz47vC6tj6PDrnFW8Kb_s9cHp-yXo9Crv4Z1Be8LEaUxIf9Pcsf2gs7jwcU4b_-zrQoN8X1r-eleIMkitjKnzE1_1
        KsqtpyMkHvwgaU2JB4_y-kmGuXhoMC2SvFZ6GGSI4G2Zl14IC7pdXhRyDP8IJWZVX93XBUZN3E1BpDT-rVh61vaJL
        YHf1nFsHeC9L8KoafLfIYW_33OUTHXHtNWIFi0ZPRBaeykIrOX5cp5PLycyWPhcCuojf144PLDhUH8Fr0dJ1lck7D
        F2rzV54T4uo3EzQbG4yd9Obd7BF8CUyddDmjI57xRJdi2w7GTP432YATPOl1zajs41gn7due24A"
}
```

This access_token includes the three different roles (Student, Registrar, and Instructor) and can be used to test all the endpoints.

Student related endpoints

LIST ENDPOINT: /student/list

This endpoint returns a list of all the available classes that are currently set to active.



GET http://localhost:5400/student/list

Type: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization

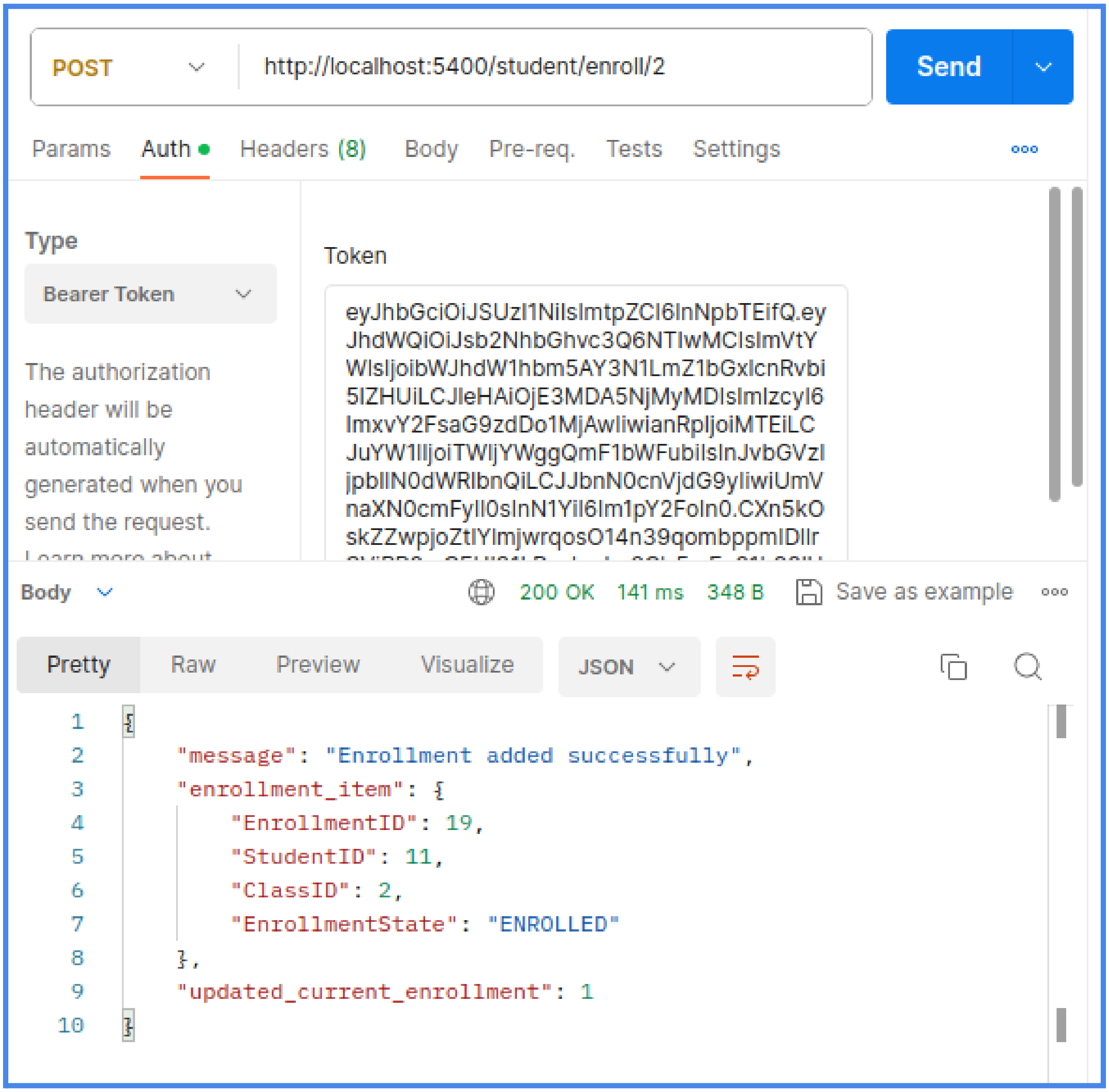
Token: eyJhbGciOiJSUzI1NiIsImtpZCI6InNpbTEifQ.eyJhdWQiOiJsb2NhbGhvc3Q6NTIwMCIsImVtYWlsIjoibWJhdW1hbm5AY3N1LmZ1bGxlcnRvbi5lZHUiLCJleHAiOjE3MDA5NjMyMDIsImlzcyI6ImxvY2FsaG9zdDo1MjAwIiwianRpIjoiMTEiLCJuYW1lIjoiTWljYWggQmF1bWFubiIsInJvbGVzIjpbIlN0dWRlbnQiLCJJbnN0cnVjdG9yIiwiUmVnaXN0cmFyIl0sInN1YiI6Im1pY2FoIn0.CXn5kOskZZwpjoZtlYlmjwrqosO1

200 OK 580 ms 1.21 KB

```
{
    "Classes": [
        {
            "ClassID": 2,
            "ClassName": "Introduction to Computer Science",
            "CourseCode": "CS-101",
            "CurrentEnrollment": 0,
            "CurrentWaitlist": 0,
            "Department": "Computer Science",
            "InstructorID": 11,
            "MaxCapacity": 50,
            "SectionNumber": 2,
            "WaitlistMaximum": 30
        },
        {
            "ClassID": 4,
            "ClassName": "English 101",
            "CourseCode": "ENG-101",
            "CurrentEnrollment": 0,
            "CurrentWaitlist": 0,
            "Department": "English",
```

ENROLL ENDPOINT: /student/enroll/{classid}

This endpoint enrolls the student in the desired class.



POST http://localhost:5400/student/enroll/2

Type: Bearer Token

The authorization header will be automatically generated when you send the request.

Token

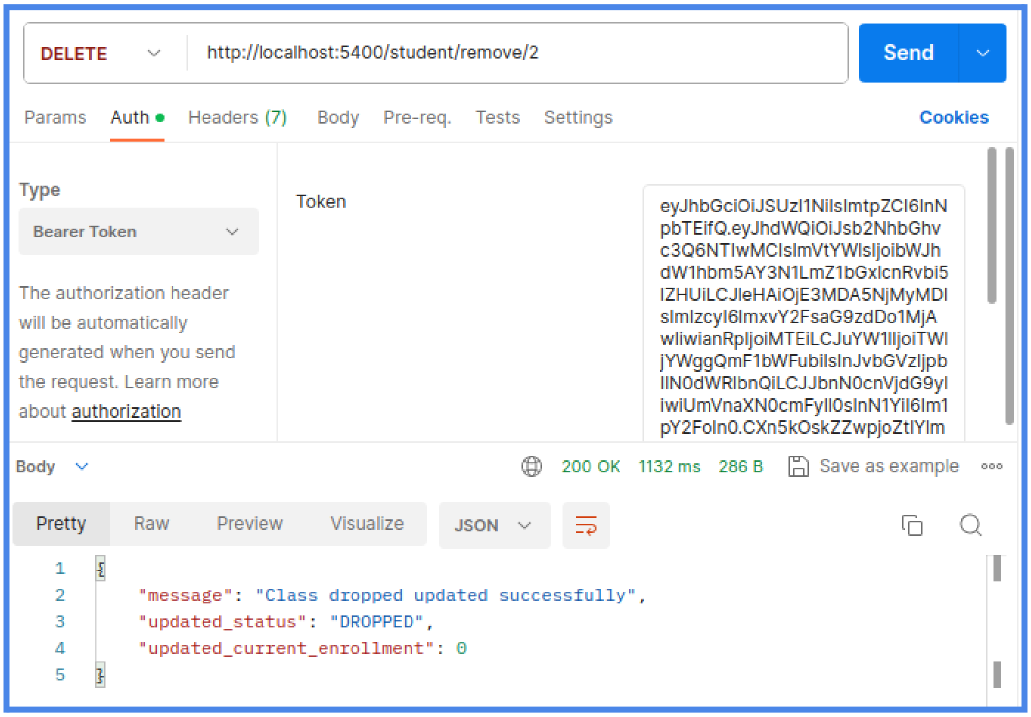
```
eyJhbGciOiJSUzI1NiIsImtpZCI6InNpbTEifQ.ey
JhdWQiOiJsb2NhbGhvc3Q6NTIwMCIsImVtY
WlsIjoibWJhdW1hbm5AY3N1LmZ1bGxlcnRvbi
5lZHUiLCJleHAiOjE3MDA5NjMyMDIsImlzcyI6
ImxvY2FsaG9zdDo1MjAwIiwianRpIjoiMTEiLC
JuYW1lIjoiTWljYWggQmF1bWFubiIsInJvbGVzI
jpbIlN0dWRlbnQiLCJJbnN0cnVjdG9yIiwiUmV
naXN0cmFyIl0sInN1YiI6Im1pY2FoIn0.CXn5kO
skZZwpjoZtlYlmjwrqosO14n39qombppmIDllr
```

200 OK 141 ms 348 B

```json
{
    "message": "Enrollment added successfully",
    "enrollment_item": {
        "EnrollmentID": 19,
        "StudentID": 11,
        "ClassID": 2,
        "EnrollmentState": "ENROLLED"
    },
    "updated_current_enrollment": 1
}
```

DROP ENDPOINT: /student/enroll/{classid}

This endpoint drops the student from the specified class.



DELETE http://localhost:5400/student/remove/2

Send

Params Auth Headers (7) Body Pre-req. Tests Settings Cookies

Type: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization

Token

```
eyJhbGciOiJSUzI1NiIsImtpZCI6InN
pbTEifQ.eyJhdWQiOiJsb2NhbGhv
c3Q6NTIwMCIsImVtYWlsIjoibWJh
dW1hbm5AY3N1LmZ1bGxlcnRvbi5
lZHUiLCJleHAiOjE3MDA5NjMyMDI
sImlzcyI6ImxvY2FsaG9zdDo1MjA
wIiwianRpIjoiMTEiLCJuYW1lIjoiTWl
jYWggQmF1bWFubiIsInJvbGVzIjpb
IlN0dWRlbnQiLCJJbnN0cnVjdG9yI
iwiUmVnaXN0cmFyIl0sInN1YiI6Im1
pY2FoIn0.CXn5kOskZZwpjoZtlYlm
```

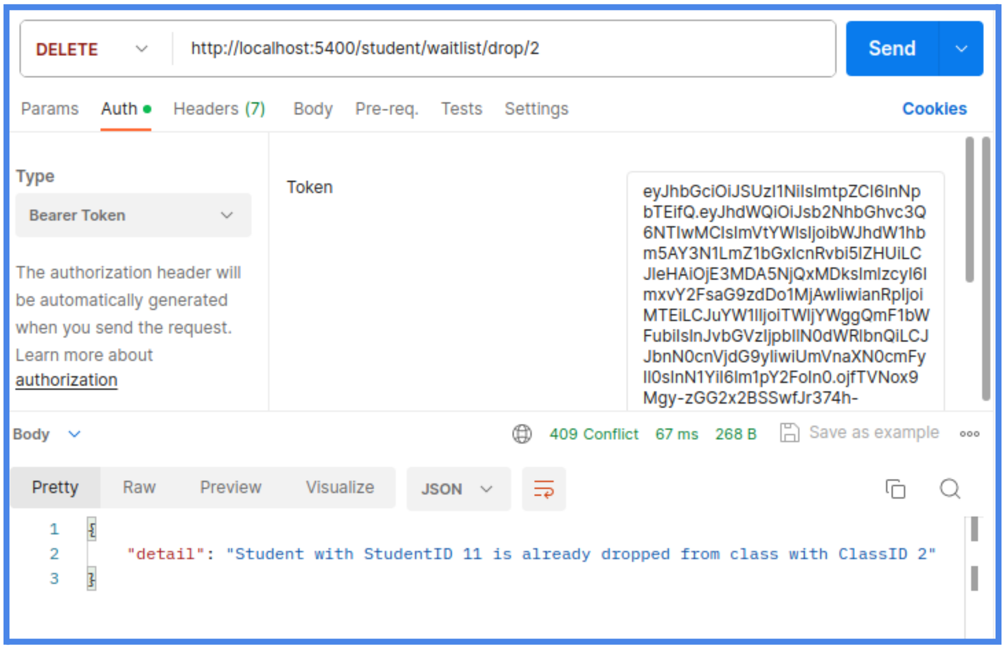
Body 200 OK 1132 ms 286 B Save as example

Pretty Raw Preview Visualize JSON

```
{
    "message": "Class dropped updated successfully",
    "updated_status": "DROPPED",
    "updated_current_enrollment": 0
}
```

DROP WAITLIST ENDPOINT: /student/waitlist/drop/{classid}

This endpoint drops the student from the waitlist of the desired class.



DELETE http://localhost:5400/student/waitlist/drop/2

Send

Params Auth Headers (7) Body Pre-req. Tests Settings Cookies

Type: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization

Token

```
eyJhbGciOiJSUzI1NiIsImtpZCI6InNp
bTEifQ.eyJhdWQiOiJsb2NhbGhvc3Q
6NTIwMCIsImVtYWlsIjoibWJhdW1hb
m5AY3N1LmZ1bGxlcnRvbi5lZHUiLC
JleHAiOjE3MDA5NjQxMDksImlzcyI6I
mxvY2FsaG9zdDo1MjAwIiwianRpIjoi
MTEiLCJuYW1lIjoiTWljYWggQmF1bW
FubiIsInJvbGVzIjpbIlN0dWRlbnQiLCJ
JbnN0cnVjdG9yIiwiUmVnaXN0cmFy
Il0sInN1YiI6Im1pY2FoIn0.ojfTVNox9
Mgy-zGG2x2BSSwfJr374h-
```

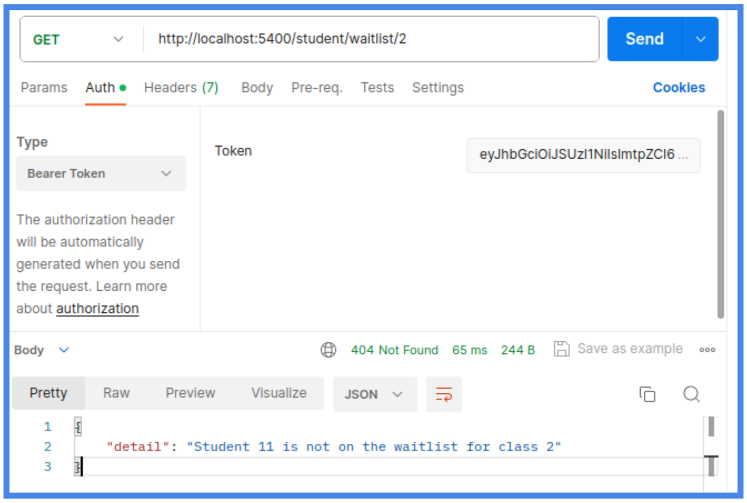
Body 409 Conflict 67 ms 268 B Save as example

Pretty Raw Preview Visualize JSON

```
{
    "detail": "Student with StudentID 11 is already dropped from class with ClassID 2"
}
```

WAITLIST ENDPOINT: /student/waitlist/{classid}

This endpoint returns the user's position on the waitlist.



GET http://localhost:5400/student/waitlist/2

Type: Bearer Token

Token: eyJhbGciOiJSUzI1NiIsImtpZCI6 ...

The authorization header will be automatically generated when you send the request. Learn more about authorization

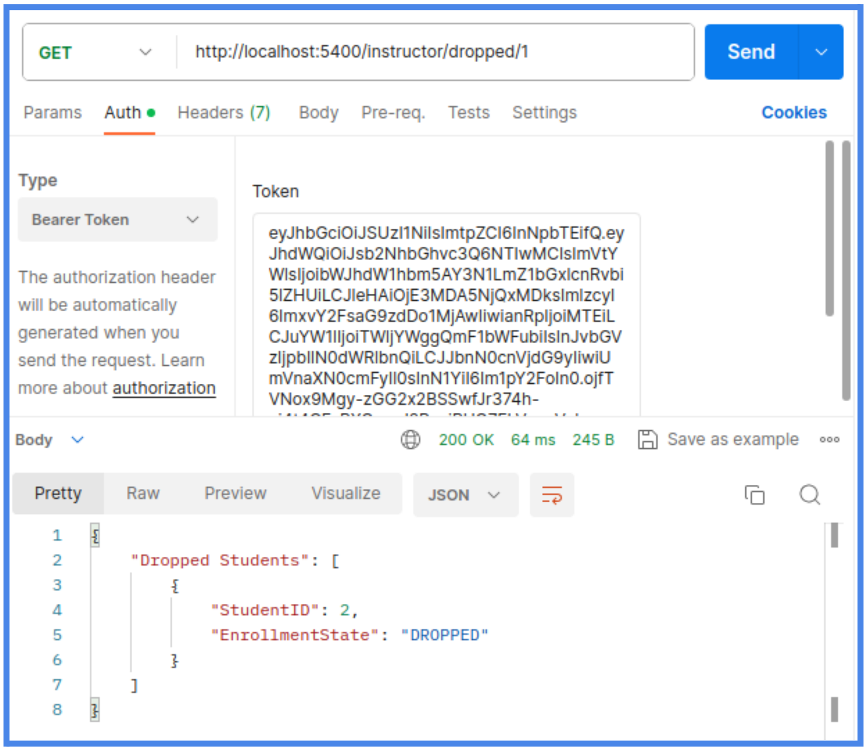
404 Not Found 65 ms 244 B

```
{
    "detail": "Student 11 is not on the waitlist for class 2"
}
```

Instructor related endpoints

DROPPED ENDPOINT: /instructor/dropped/{classid}

This endpoint returns all the students that have dropped the specified class. Again, only if the instructor teaches this class.



GET http://localhost:5400/instructor/dropped/1

Type: Bearer Token

Token:

```
eyJhbGciOiJSUzI1NiIsImtpZCI6InNpbTEifQ.ey
JhdWQiOiJsb2NhbGhvc3Q6NTIwMCIsImVtY
WlsIjoibWJhdW1hbm5AY3N1LmZ1bGxlcnRvbi
5lZHUiLCJleHAiOjE3MDA5NjQxMDksImlzcyI
6ImxvY2FsaG9zdDo1MjAwIiwianRpIjoiMTEiL
CJuYW1lIjoiTWljYWggQmF1bWFubiIsInJvbGV
zIjpbIlN0dWRlbnQiLCJJbnN0cnVjdG9yIiwiU
mVnaXN0cmFyIl0sInN1YiI6Im1pY2FoIn0.ojfT
VNox9Mgy-zGG2x2BSSwfJr374h-
```

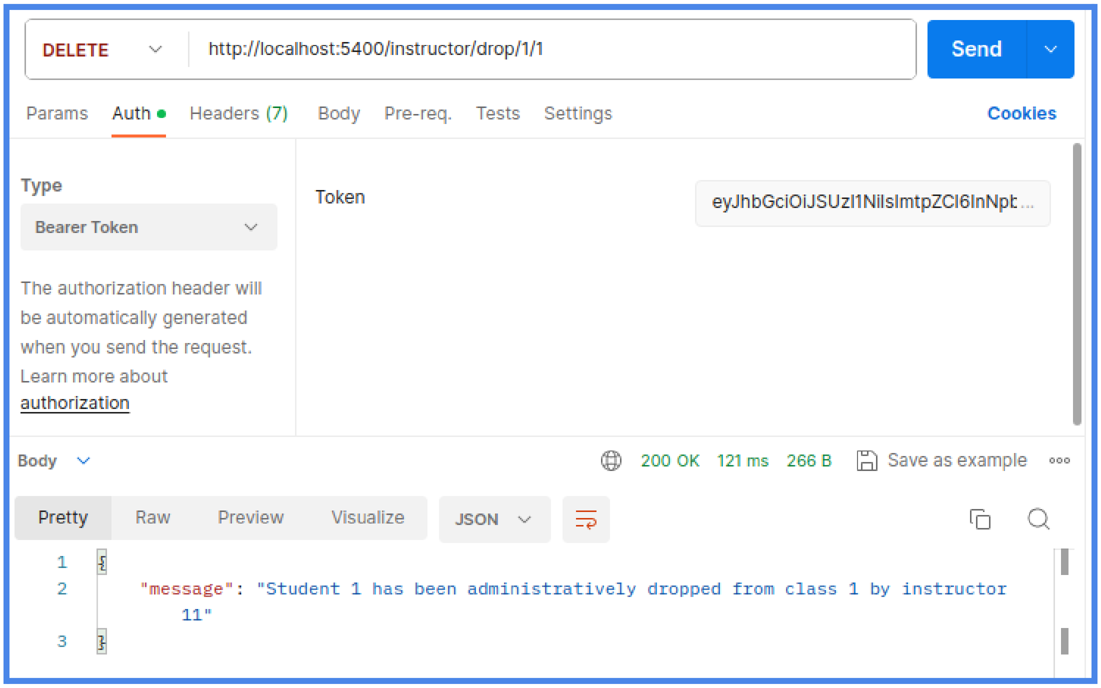
The authorization header will be automatically generated when you send the request. Learn more about authorization

200 OK 64 ms 245 B

```
{
    "Dropped Students": [
        {
            "StudentID": 2,
            "EnrollmentState": "DROPPED"
        }
    ]
}
```

DROP ENDPOINT: /instructor/drop/{classid}/{studentid}

This endpoint can be used to administratively drop a student from a class. It can only be used if the instructor teaches the class.



DELETE http://localhost:5400/instructor/drop/1/1

Type: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization

Token: eyJhbGciOiJSUzI1NiIsImtpZCI6InNpb...

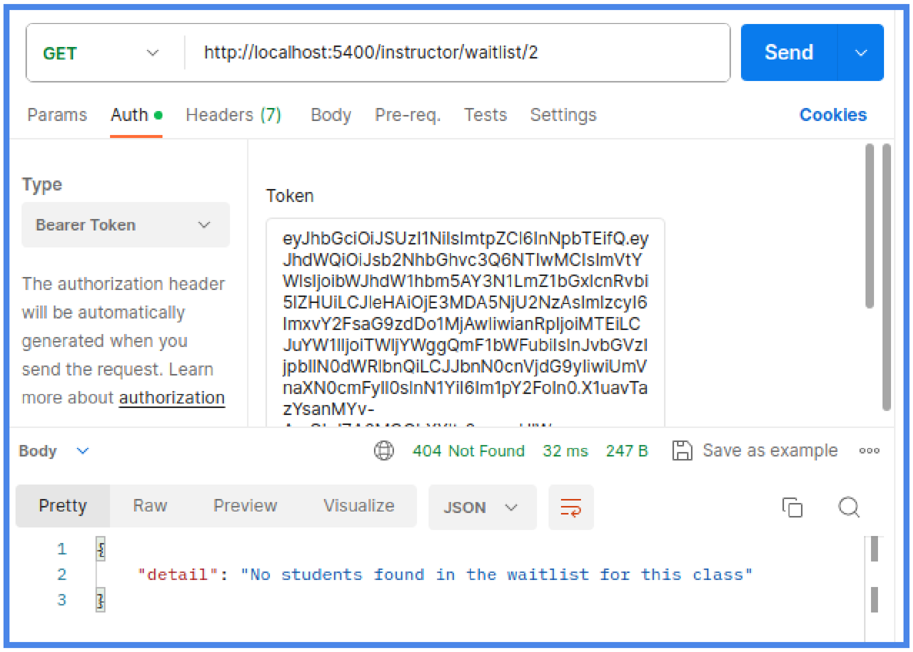
200 OK 121 ms 266 B

JSON

```
{
    "message": "Student 1 has been administratively dropped from class 1 by instructor
       11"
}
```

WAITLIST ENDPOINT: /instructor/waitlist/{classid}

This endpoint returns all the students on the waitlist for the given class.



GET http://localhost:5400/instructor/waitlist/2

Type: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization

Token:

```
eyJhbGciOiJSUzI1NiIsImtpZCI6InNpbTEifQ.ey
JhdWQiOiJsb2NhbGhvc3Q6NTIwMCIsImVtY
WlsIjoibWJhdW1hbm5AY3N1LmZ1bGxlcnRvbi
5lZHUiLCJleHAiOjE3MDA5NjU2NzAsImlzcyI6
ImxvY2FsaG9zdDo1MjAwIiwianRpIjoiMTEiLC
JuYW1lIjoiTWljYWggQmF1bWFubiIsInJvbGVzI
jpbIlN0dWRlbnQiLCJJbnN0cnVjdG9yIiwiUmV
naXN0cmFyIl0sInN1YiI6Im1pY2FoIn0.X1uavTa
zYsanMYv-
```

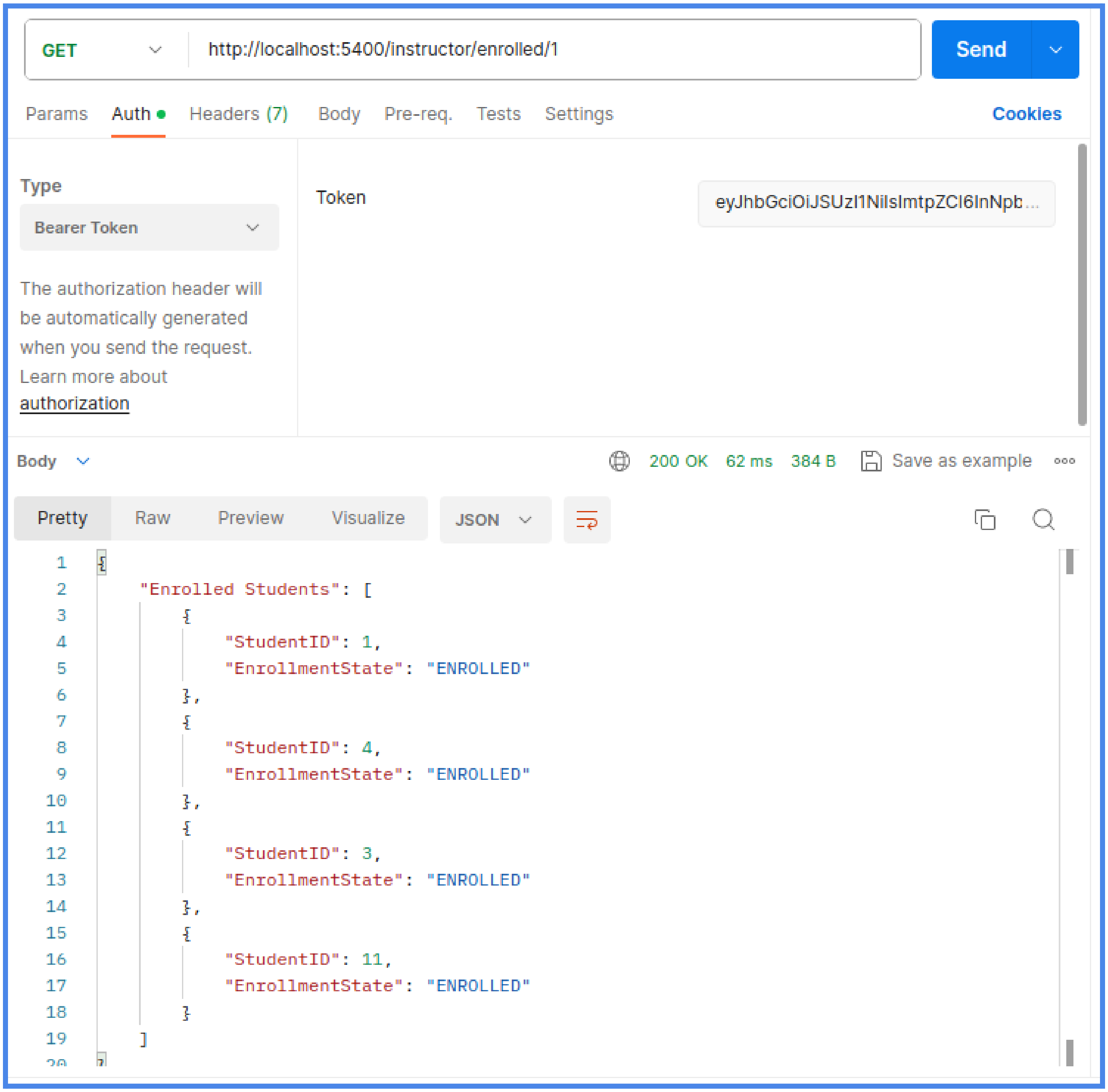
404 Not Found 32 ms 247 B

JSON

```
{
    "detail": "No students found in the waitlist for this class"
}
```

ENROLLED ENDPOINT: /instructor/enrolled/{classid}

This endpoint returns all the students enrolled in a specific class if the instructor teaches this class.



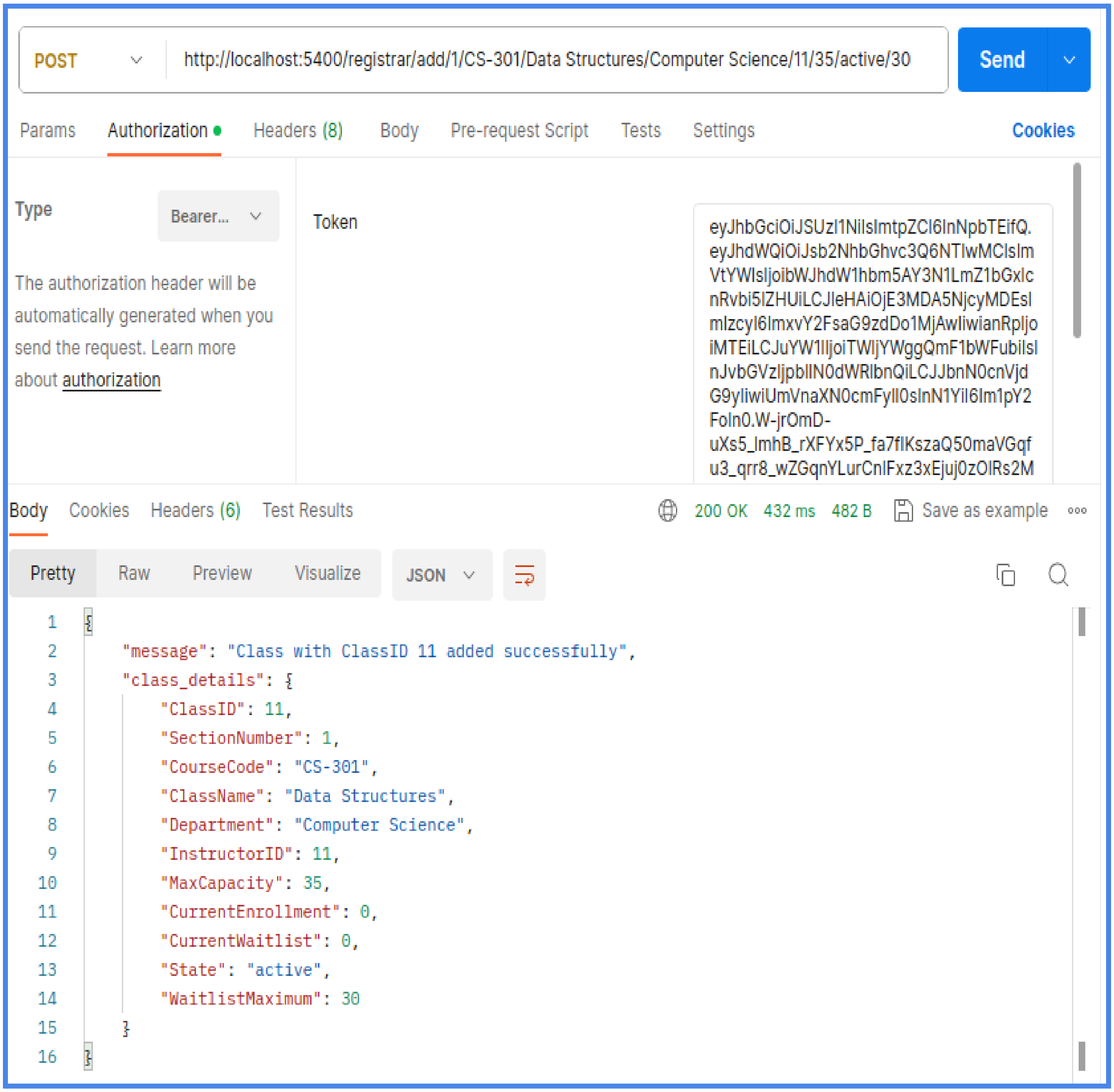
GET
http://localhost:5400/instructor/enrolled/1
Send
Params
Auth
Headers (7)
Body
Pre-req.
Tests
Settings
Cookies
Type
Bearer Token
The authorization header will be automatically generated when you send the request. Learn more about authorization
Token
eyJhbGciOiJSUzI1NiIsImtpZCI6InNpb...
Body
200 OK
62 ms
384 B
Save as example
Pretty
Raw
Preview
Visualize
JSON

```
{
    "Enrolled Students": [
        {
            "StudentID": 1,
            "EnrollmentState": "ENROLLED"
        },
        {
            "StudentID": 4,
            "EnrollmentState": "ENROLLED"
        },
        {
            "StudentID": 3,
            "EnrollmentState": "ENROLLED"
        },
        {
            "StudentID": 11,
            "EnrollmentState": "ENROLLED"
        }
    ]
}
```

Registrar related endpoints

ADD ENDPOINT:
/registrar/add/{sectionid}/{coursecode}/{classname}/{department}/{professorid}/{enrollmax}/{status}/{waitmax}

This creates a new class and adds it to the table. It can only be used by a registrar.



http://localhost:5400/registrar/add/1/CS-301/Data Structures/Computer Science/11/35/active/30

The authorization header will be automatically generated when you send the request. Learn more about authorization

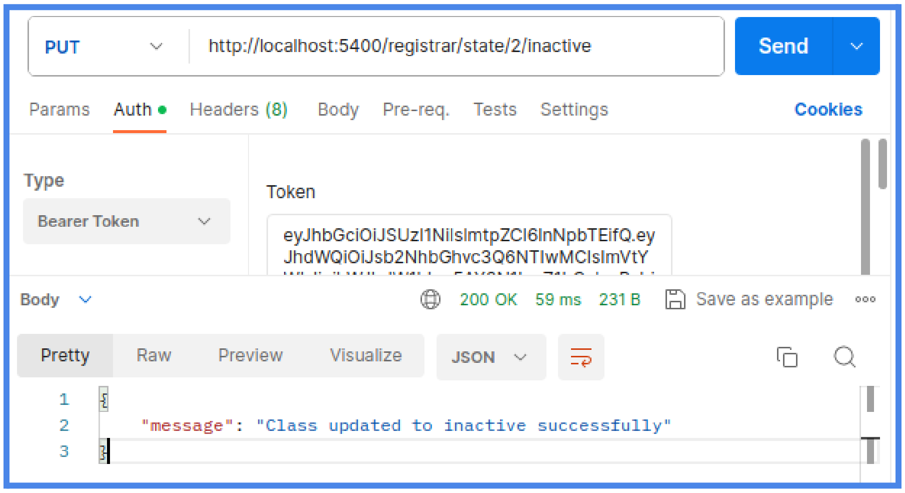
```
eyJhbGciOiJSUzI1NiIsImtpZCI6InNpbTEifQ.
eyJhdWQiOiJsb2NhbGhvc3Q6NTIwMCIsIm
VtYWlsIjoibWJhdW1hbm5AY3N1LmZ1bGxlc
nRvbi5lZHUiLCJleHAiOjE3MDA5NjcyMDEsI
mlzcyI6ImxvY2FsaG9zdDo1MjAwIiwianRpIjo
iMTEiLCJuYW1lIjoiTWljYWggQmF1bWFubiIsI
nJvbGVzIjpbIlN0dWRlbnQiLCJJbnN0cnVjd
G9yIiwiUmVnaXN0cmFyIl0sInN1YiI6Im1pY2
FoIn0.W-jrOmD-
uXs5_ImhB_rXFYx5P_fa7fIKszaQ50maVGqf
u3_qrr8_wZGqnYLurCnIFxz3xEjuj0zOIRs2M
```

200 OK 432 ms 482 B

```
{
    "message": "Class with ClassID 11 added successfully",
    "class_details": {
        "ClassID": 11,
        "SectionNumber": 1,
        "CourseCode": "CS-301",
        "ClassName": "Data Structures",
        "Department": "Computer Science",
        "InstructorID": 11,
        "MaxCapacity": 35,
        "CurrentEnrollment": 0,
        "CurrentWaitlist": 0,
        "State": "active",
        "WaitlistMaximum": 30
    }
}
```

STATE ENDPOINT: /registrar/state/{classid}/{state}

This endpoint changes the state of a class.



PUT http://localhost:5400/registrar/state/2/inactive

Type: Bearer Token

Token

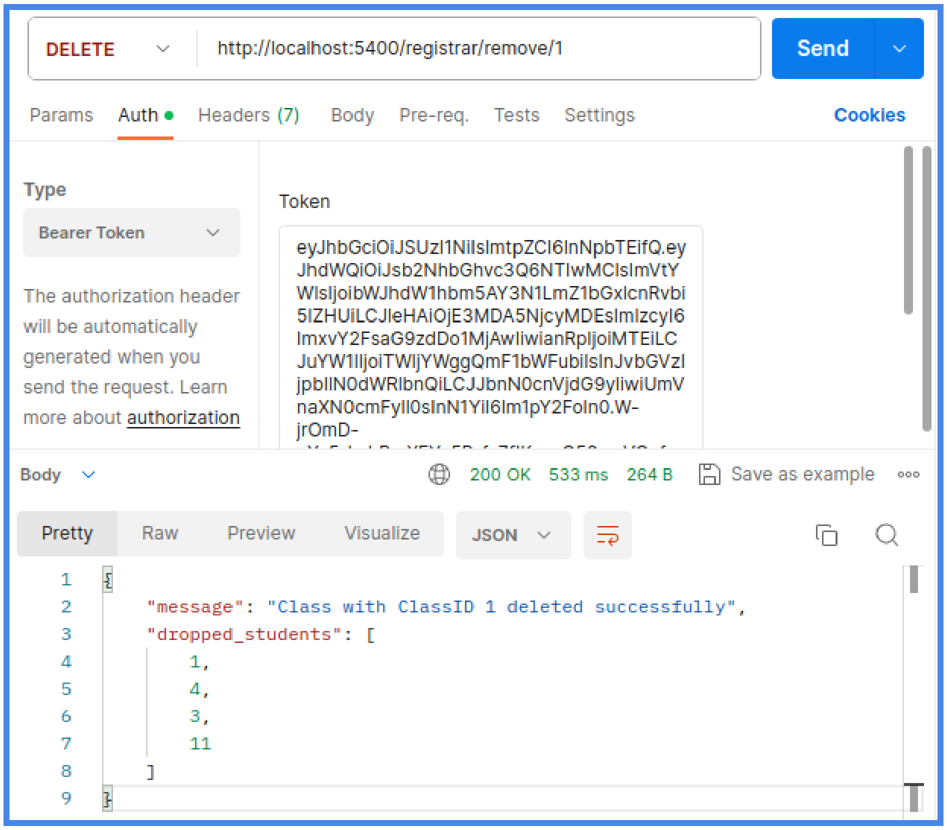
```
eyJhbGciOiJSUzI1NiIsImtpZCI6InNpbTEifQ.ey
JhdWQiOiJsb2NhbGhvc3Q6NTIwMCIsImVtY
```

200 OK 59 ms 231 B

```
{
    "message": "Class updated to inactive successfully"
}
```

REMOVE ENDPOINT: /registrar/remove/{classid}

This endpoint removes a class from the "Classes" table. If there are any students enrolled in the class, the endpoint will also drop these students from the class.



DELETE http://localhost:5400/registrar/remove/1

Type: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization

Token

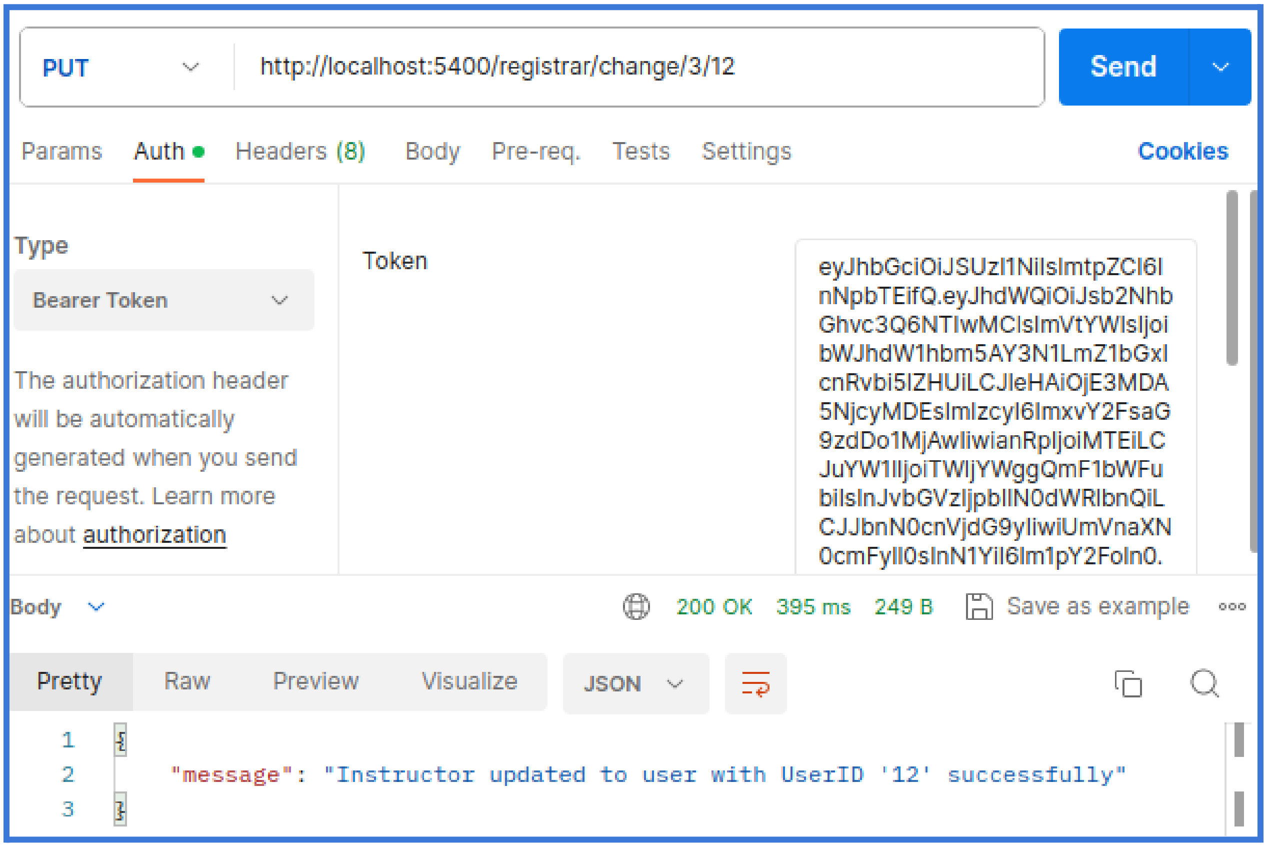
```
eyJhbGciOiJSUzI1NiIsImtpZCI6InNpbTEifQ.ey
JhdWQiOiJsb2NhbGhvc3Q6NTIwMCIsImVtY
WlsIjoibWJhdW1hbm5AY3N1LmZ1bGxlcnRvbi
5lZHUiLCJleHAiOjE3MDA5NjcyMDEsImlzcyI6
ImxvY2FsaG9zdDo1MjAwIiwianRpIjoiMTEiLC
JuYW1lIjoiTWljYWggQmF1bWFubiIsInJvbGVzI
jpbIlN0dWRlbnQiLCJJbnN0cnVjdG9yIiwiUmV
naXN0cmFyIl0sInN1YiI6Im1pY2FoIn0.W-
jrOmD-
```

200 OK 533 ms 264 B

```
{
    "message": "Class with ClassID 1 deleted successfully",
    "dropped_students": [
        1,
        4,
        3,
        11
    ]
}
```

CHANGE ENDPOINT: /registrar/change/{classid}/{newprofessorid}

This endpoint changes the instructor for a given class.



http://localhost:5400/registrar/change/3/12

Type: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization

Token

```
eyJhbGciOiJSUzI1NiIsImtpZCI6I
nNpbTEifQ.eyJhdWQiOiJsb2Nhb
Ghvc3Q6NTIwMCIsImVtYWlsIjoi
bWJhdW1hbm5AY3N1LmZ1bGxl
cnRvbi5lZHUiLCJleHAiOjE3MDA
5NjcyMDEsImlzcyI6ImxvY2FsaG
9zdDo1MjAwIiwianRpIjoiMTEiLC
JuYW1lIjoiTWljYWggQmF1bWFu
bilsInJvbGVzIjpbIlN0dWRlbnQiL
CJJbnN0cnVjdG9yIiwiUmVnaXN
0cmFyIl0sInN1YiI6Im1pY2FoIn0.
```

200 OK 395 ms 249 B

```
{
    "message": "Instructor updated to user with UserID '12' successfully"
}
```

Additional instructions on how to run the project are included in README.md