449 Project 1 - Group 5

Members:
- Alexis Martin
- Shadi Nachat
- Angel Calderon
- Nilay Jain
- Kevin Loi
- Yuling Ge

Course Number: 449
Section: 2
Semester: Fall 2023
Project: 1

Project Summary

Objective:
Design and implement a back-end Web Service to manage course enrollment and waiting lists, mirroring the functionality of TitanOnline.

Core Tasks:

Classes:
- Attributes: Department, course code, section number, name, and instructor. Track current and maximum enrollment.
- Student Functions: List classes, enroll, drop.
- Instructor Functions: View enrollment, view dropped students, administratively drop students.
- Registrar Functions: Add/remove sections, change instructors, freeze automatic enrollment.

Waiting Lists:
- Automatic addition if class is full. Max list size: 15. Max waiting lists per student: 3.
- Student Functions: View position, remove from list.
- Instructor Functions: View waiting list.

Database Design:
- SQL schema in approximately third normal form.
- Pre-populate with sample data.

Service Implementation:

- Use FastAPI for endpoints and resource representations.
- Stateless service.
- JSON format for input and output.

Task 1 - Define the API

Our RESTful service exposes the following resources and operations (Example Inputs and Outputs at very end of this document):

Query 1: Student can list all available classes

Format: GET http://localhost:5000/student/available_classes
Example: GET http://localhost:5000/student/available_classes

Query 2: Student can attempt to enroll in a class

Format: POST http://localhost:5000/student/enroll_in_class/student/{student_id}/class/{class_code}/section/{section_number}
Example: POST http://localhost:5000/student/enroll_in_class/student/11111111/class/CHEM101/section/01

Query 3: Student can drop a class

Format: DELETE http://localhost:5000/student/drop_class/student/{student_id}/class/{class_code}/section/{section_number}
Example: DELETE http://localhost:5000/student/drop_class/student/11111111/class/MATH101/section/01

Query 4: Instructor can view current enrollment for their classes

Format: GET http://localhost:5000/instructor/enrollment/instructor/{instructor_id}
Example: GET http://localhost:5000/instructor/enrollment/instructor/100

Query 5: Instructor can view students who have dropped the class

Format: GET
http://localhost:5000/instructor/dropped/instructor/{instructor_id}/class/{class_code}/section/{section_number}
Example: GET
http://localhost:5000/instructor/dropped/instructor/100/class/CPSC449/section/01

Query 6: Instructor can drop students administratively (e.g. if they do not show up to class)

Format: DELETE
http://localhost:5000/instructor/drop_student/student/{student_id}/class/{class_code}/section/{section_number}
Example: DELETE
http://localhost:5000/instructor/drop_student/student/11111111/class/CPSC449/section/01

Query 7: Instructor can drop students administratively (e.g. if they do not show up to class)

Format: POST http://localhost:5000/registrar/new_class
```
Body: {
    "class_code": "CPSC449",
    "section_number": "04",
    "class_name": "Database Systems",
    "department": "Computer Science",
    "auto_enrollment": true,
    "max_enrollment": 30,
    "max_waitlist": 15,
    "c_instructor_id": "100"
 }
```

Query 8: Registrar can remove existing sections

Format: DELETE
http://localhost:5000/registrar/remove_section/class/{class_code}/section/{section_number}
Example: DELETE http://localhost:5000/registrar/remove_class/code/CPSC449/section/04

Query 9: Registrar can change instructor for a section

Format: PATCH
http://localhost:5000/registrar/change_instructor/class/{class_code}/section/{section_number}/new_instructor/{instructor_id}
Example: PATCH
http://localhost:5000/registrar/change_instructor/class/CPSC449/section/01/new_instructor/101

Query 10: Freeze automatic enrollment from waiting lists (e.g., during the second week of classes)

Format: PATCH
http://localhost:5000/registrar/freeze_enrollment/class/{class_code}/section/{section_number}
Example: PATCH http://localhost:5000/registrar/freeze_enrollment/class/CPSC449/section/01

Query 11: Student can view their current position on the waiting list

Format: GET
http://localhost:5000/student/waitlist_position/student/{student_id}/class/{class_code}/section/{section_number}
Example: GET
http://localhost:5000/student/waitlist_position/student/44444444/class/ENGL205/section/01

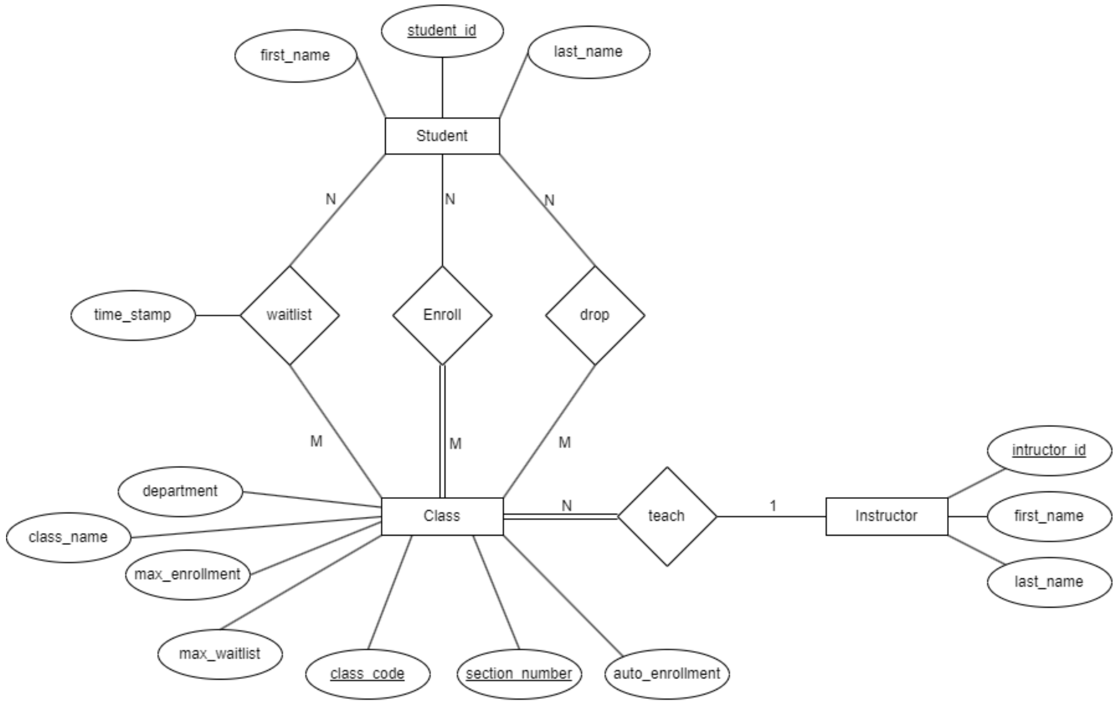
Query 12: Student can remove themselves from a waiting list

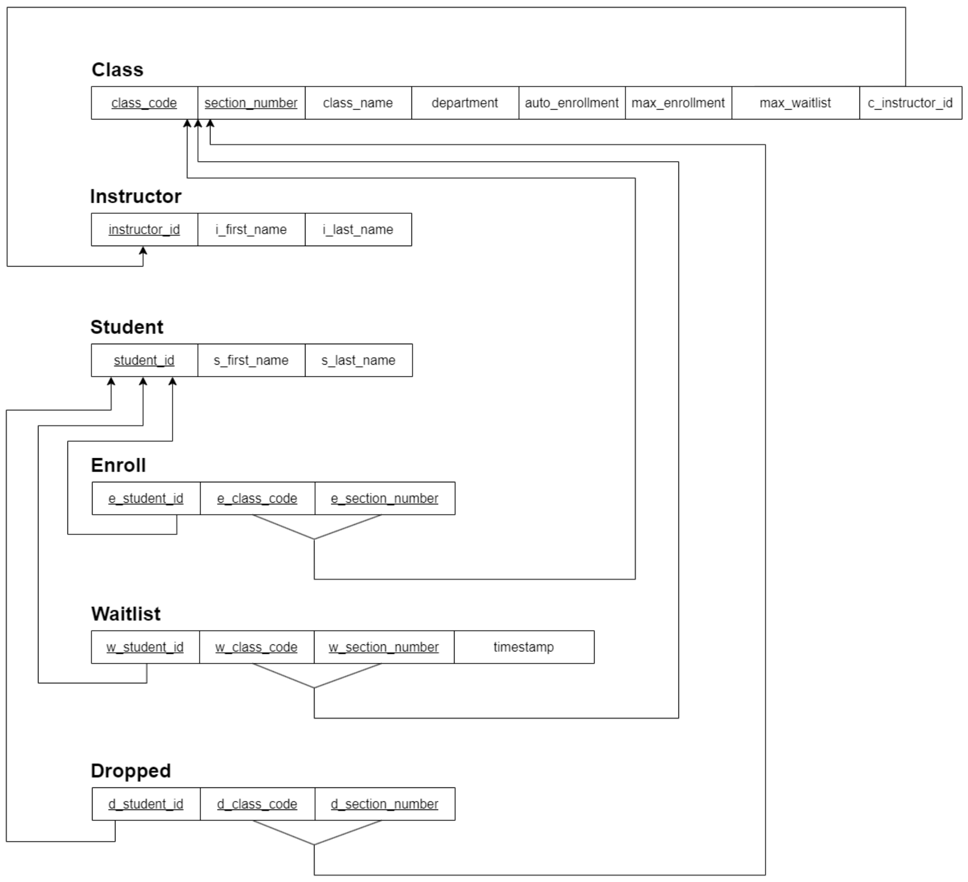
Format: DELETE
http://localhost:5000/student/remove_from_waitlist/student/{student_id}/class/{class_code}/section/{section_number}
Example: DELETE
http://localhost:5000/student/remove_from_waitlist/student/11111111/class/ENGL205/section/01

Query 13: Instructor can view the current waiting list for their course

Format: GET
http://localhost:5000/instructor/waitlist_for_class/instructor/{instructor_id}/class/{class_code}/section/{section_number}
Example: GET
http://localhost:5000/instructor/waitlist_for_class/instructor/102/class/CHEM101/section/02

Task 2 - Design the database





Task 3 - Implement the Service

To startup the database do the following:

1. Install dependencies
    - **sudo apt update**
    - **sudo apt install --yes ruby-foreman**
    - **python -m pip install 'fastapi[all]'**

2. Unpack the tarball
    - **tar zxvf group5.tar.gz**

3. From within the **/group5** folder run the following

    Populate the database with sample data
    - **./bin/init.sh**

    Start the api
    - **foreman start**

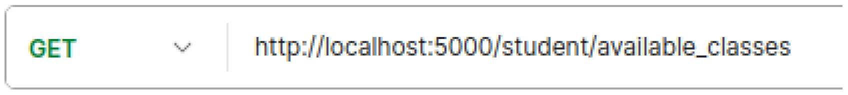
4. Access documentation
    - **http://localhost:5000/docs**

Example Inputs and Outputs

Using your preferred http request tool (we are using postman) make the following requests

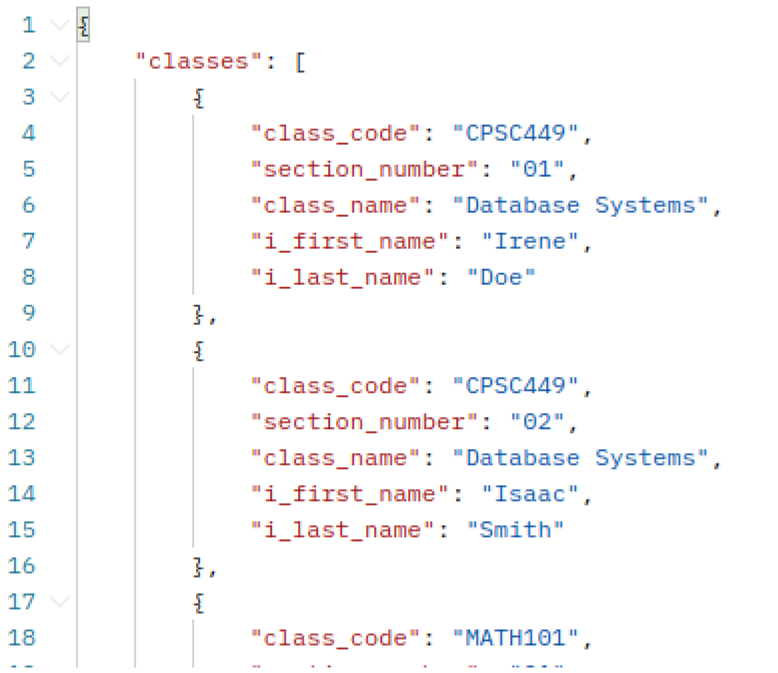
Query 1: Student can list all available classes

Request: GET http://localhost:5000/student/available_classes



GET http://localhost:5000/student/available_classes

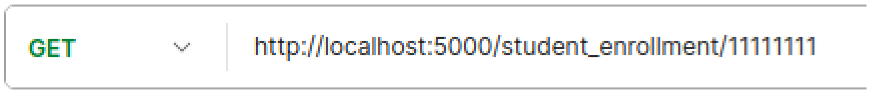
Response

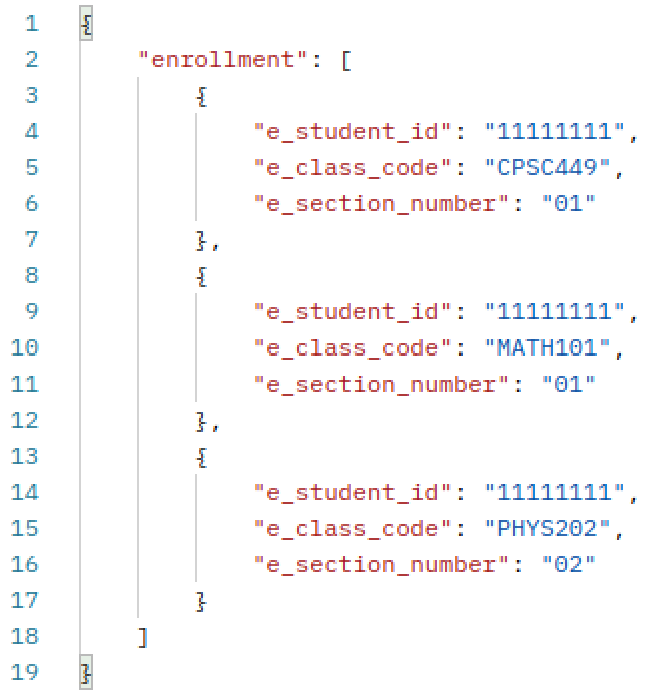


```
{
    "classes": [
        {
            "class_code": "CPSC449",
            "section_number": "01",
            "class_name": "Database Systems",
            "i_first_name": "Irene",
            "i_last_name": "Doe"
        },
        {
            "class_code": "CPSC449",
            "section_number": "02",
            "class_name": "Database Systems",
            "i_first_name": "Isaac",
            "i_last_name": "Smith"
        },
        {
            "class_code": "MATH101",
```

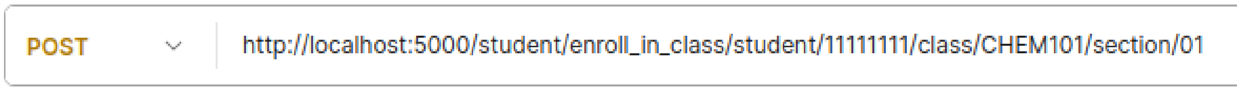
Query 2: Student can attempt to enroll in a class

Before: GET http://localhost:5000/student_enrollment/11111111

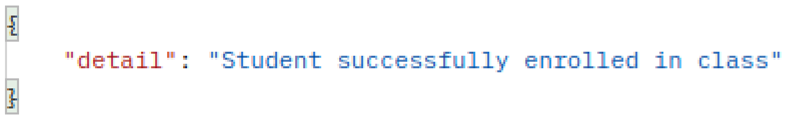

GET http://localhost:5000/student_enrollment/11111111


```
{
    "enrollment": [
        {
            "e_student_id": "11111111",
            "e_class_code": "CPSC449",
            "e_section_number": "01"
        },
        {
            "e_student_id": "11111111",
            "e_class_code": "MATH101",
            "e_section_number": "01"
        },
        {
            "e_student_id": "11111111",
            "e_class_code": "PHYS202",
            "e_section_number": "02"
        }
    ]
}
```

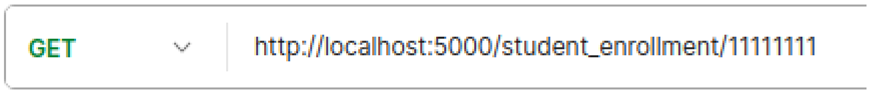
Request: POST
http://localhost:5000/student/enroll_in_class/student/11111111/class/CHEM101/section/01

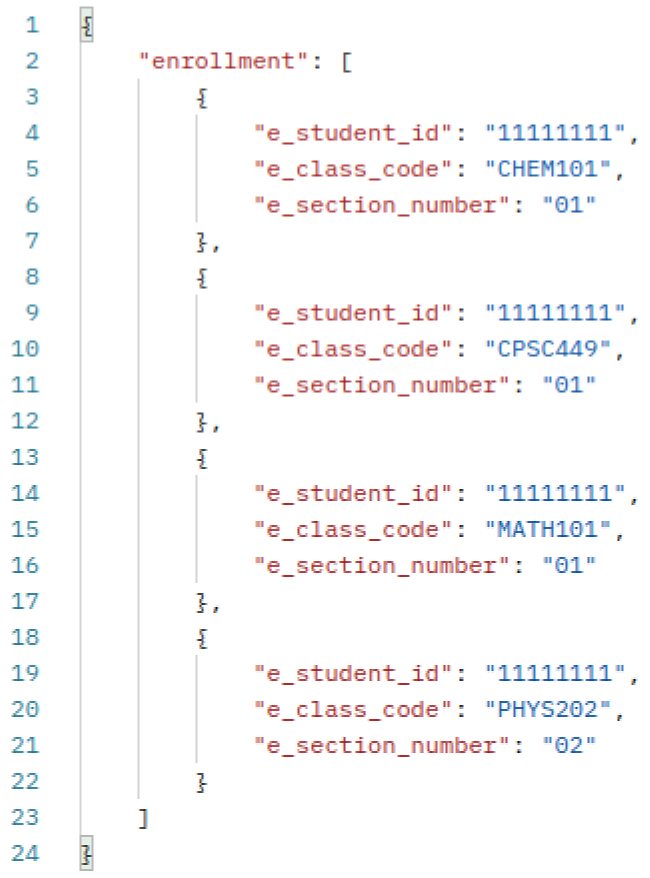

POST http://localhost:5000/student/enroll_in_class/student/11111111/class/CHEM101/section/01

Response:


```
{
    "detail": "Student successfully enrolled in class"
}
```

After: GET http://localhost:5000/student_enrollment/11111111

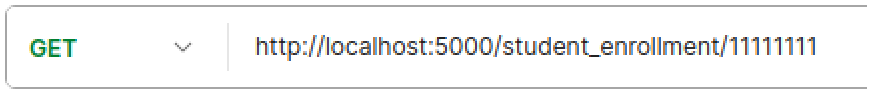

GET http://localhost:5000/student_enrollment/11111111



```json
{
    "enrollment": [
        {
            "e_student_id": "11111111",
            "e_class_code": "CHEM101",
            "e_section_number": "01"
        },
        {
            "e_student_id": "11111111",
            "e_class_code": "CPSC449",
            "e_section_number": "01"
        },
        {
            "e_student_id": "11111111",
            "e_class_code": "MATH101",
            "e_section_number": "01"
        },
        {
            "e_student_id": "11111111",
            "e_class_code": "PHYS202",
            "e_section_number": "02"
        }
    ]
}
```

Query 3: Student can drop a class

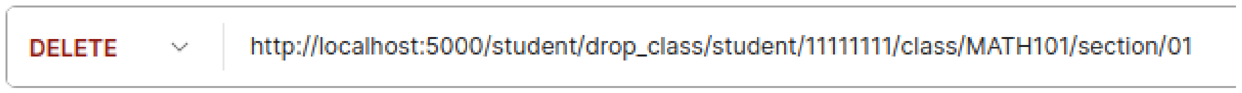
Before: GET http://localhost:5000/student_enrollment/11111111



http://localhost:5000/student_enrollment/11111111

```
{
    "enrollment": [
        {
            "e_student_id": "11111111",
            "e_class_code": "CHEM101",
            "e_section_number": "01"
        },
        {
            "e_student_id": "11111111",
            "e_class_code": "CPSC449",
            "e_section_number": "01"
        },
        {
            "e_student_id": "11111111",
            "e_class_code": "MATH101",
            "e_section_number": "01"
        },
        {
            "e_student_id": "11111111",
            "e_class_code": "PHYS202",
            "e_section_number": "02"
        }
    ]
}
```

Request: DELETE
http://localhost:5000/student/drop_class/student/11111111/class/MATH101/section/01

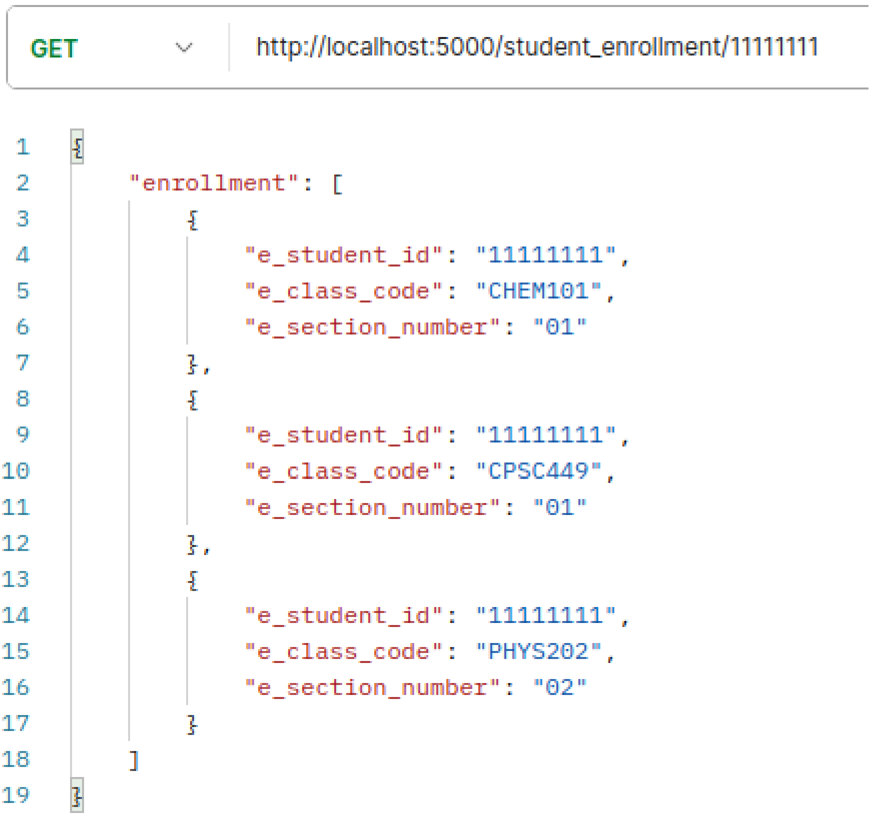


http://localhost:5000/student/drop_class/student/11111111/class/MATH101/section/01

Response:

```
{
    "detail": "Class successfully dropped."
}
```

After: GET http://localhost:5000/student_enrollment/11111111

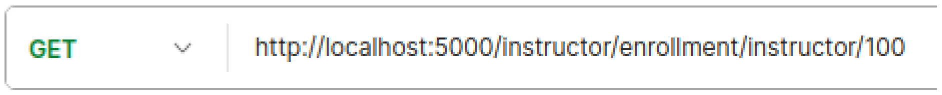


GET http://localhost:5000/student_enrollment/11111111

```
{
    "enrollment": [
        {
            "e_student_id": "11111111",
            "e_class_code": "CHEM101",
            "e_section_number": "01"
        },
        {
            "e_student_id": "11111111",
            "e_class_code": "CPSC449",
            "e_section_number": "01"
        },
        {
            "e_student_id": "11111111",
            "e_class_code": "PHYS202",
            "e_section_number": "02"
        }
    ]
}
```

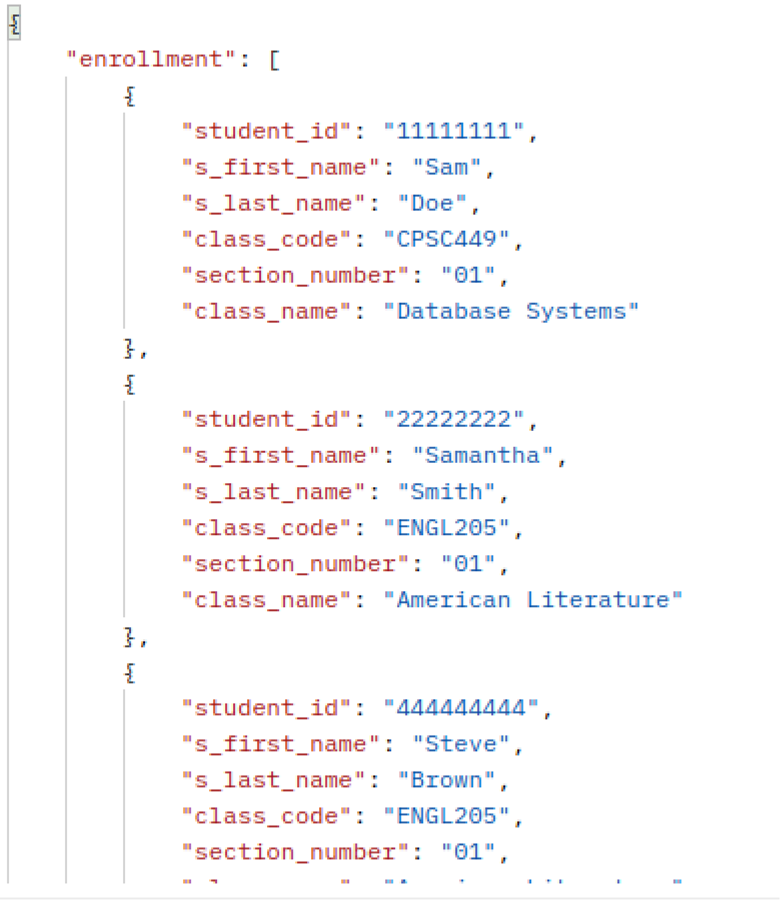
Query 4: Instructor can view current enrollment for their classes

Request: GET http://localhost:5000/instructor/enrollment/instructor/100



```
GET    http://localhost:5000/instructor/enrollment/instructor/100
```

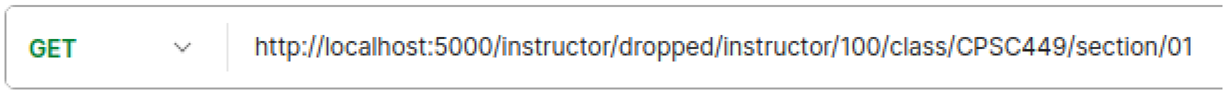
Response:



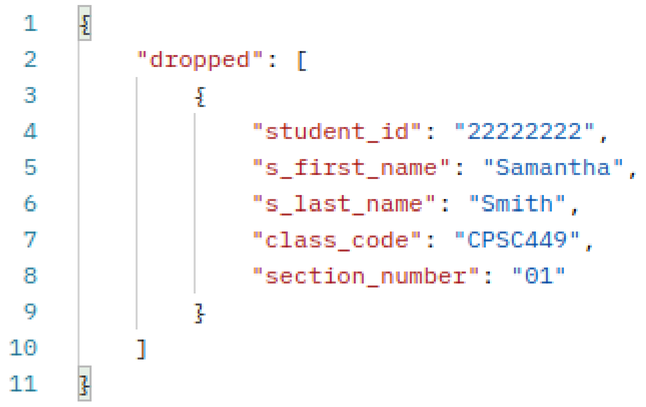
```
{
    "enrollment": [
        {
            "student_id": "11111111",
            "s_first_name": "Sam",
            "s_last_name": "Doe",
            "class_code": "CPSC449",
            "section_number": "01",
            "class_name": "Database Systems"
        },
        {
            "student_id": "22222222",
            "s_first_name": "Samantha",
            "s_last_name": "Smith",
            "class_code": "ENGL205",
            "section_number": "01",
            "class_name": "American Literature"
        },
        {
            "student_id": "444444444",
            "s_first_name": "Steve",
            "s_last_name": "Brown",
            "class_code": "ENGL205",
            "section_number": "01",
```

Query 5: Instructor can view students who have dropped the class

Request: GET
http://localhost:5000/instructor/dropped/instructor/100/class/CPSC449/section/01

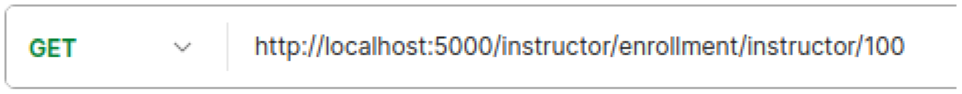

GET http://localhost:5000/instructor/dropped/instructor/100/class/CPSC449/section/01

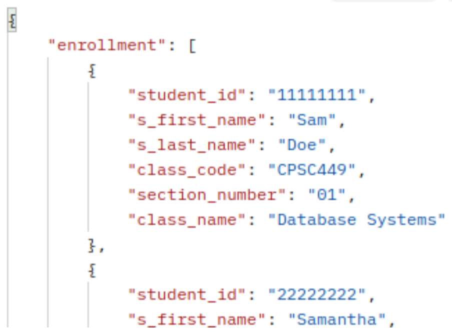
Response:


```
{
    "dropped": [
        {
            "student_id": "22222222",
            "s_first_name": "Samantha",
            "s_last_name": "Smith",
            "class_code": "CPSC449",
            "section_number": "01"
        }
    ]
}
```

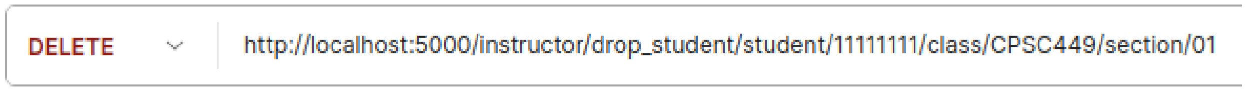
Query 6: Instructor can drop students administratively (e.g. if they do not show up to class)

Before: GET http://localhost:5000/instructor/enrollment/instructor/100

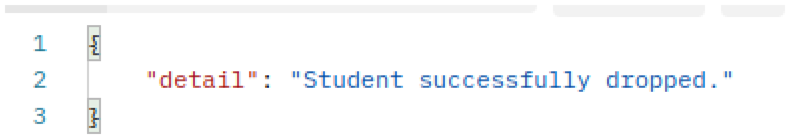

GET http://localhost:5000/instructor/enrollment/instructor/100


```
{
    "enrollment": [
        {
            "student_id": "11111111",
            "s_first_name": "Sam",
            "s_last_name": "Doe",
            "class_code": "CPSC449",
            "section_number": "01",
            "class_name": "Database Systems"
        },
        {
            "student_id": "22222222",
            "s_first_name": "Samantha",
```

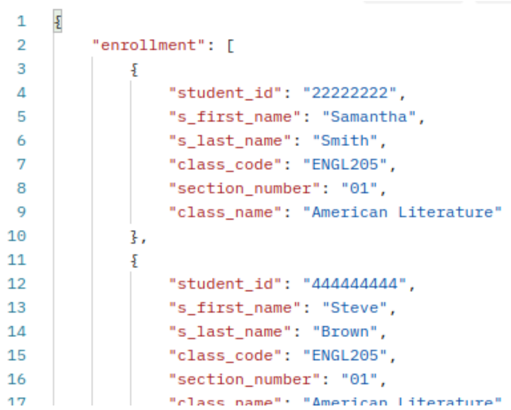
Request: DELETE http://localhost:5000/instructor/drop_student/student/11111111/class/CPSC449/section/01


DELETE http://localhost:5000/instructor/drop_student/student/11111111/class/CPSC449/section/01

Response:

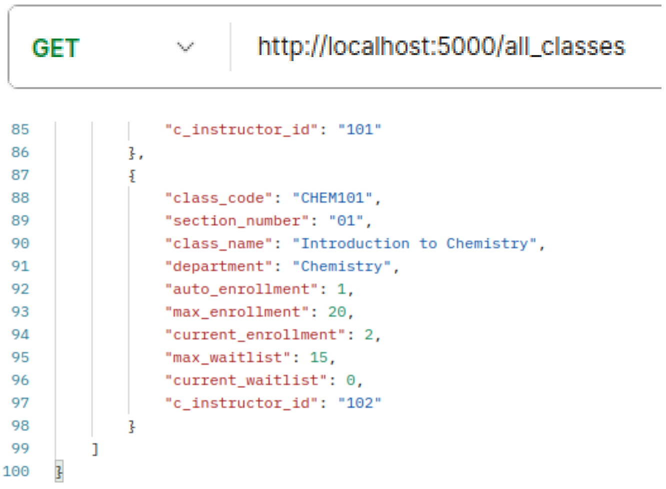

```
{
    "detail": "Student successfully dropped."
}
```

After: GET http://localhost:5000/instructor/enrollment/instructor/100


```
{
    "enrollment": [
        {
            "student_id": "22222222",
            "s_first_name": "Samantha",
            "s_last_name": "Smith",
            "class_code": "ENGL205",
            "section_number": "01",
            "class_name": "American Literature"
        },
        {
            "student_id": "444444444",
            "s_first_name": "Steve",
            "s_last_name": "Brown",
            "class_code": "ENGL205",
            "section_number": "01",
            "class_name": "American Literature"
```

Query 7: Registrar can add new section

Before: GET http://localhost:5000/all_classes

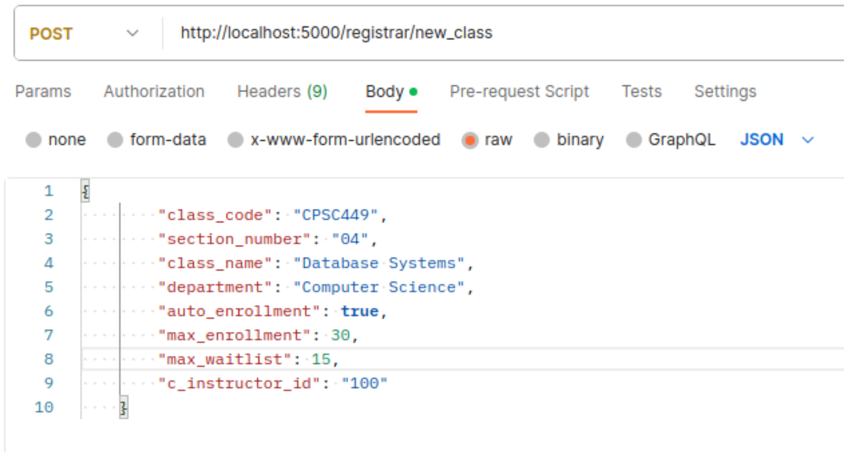


GET http://localhost:5000/all_classes

```
            "c_instructor_id": "101"
        },
        {
            "class_code": "CHEM101",
            "section_number": "01",
            "class_name": "Introduction to Chemistry",
            "department": "Chemistry",
            "auto_enrollment": 1,
            "max_enrollment": 20,
            "current_enrollment": 2,
            "max_waitlist": 15,
            "current_waitlist": 0,
            "c_instructor_id": "102"
        }
    ]
}
```

Request: POST http://localhost:5000/registrar/new_class

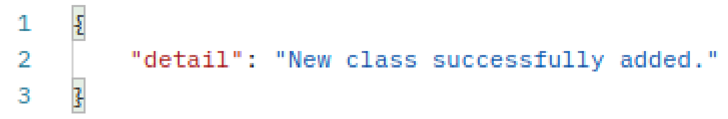
```
Body: {
    "class_code": "CPSC449",
    "section_number": "04",
    "class_name": "Database Systems",
    "department": "Computer Science",
    "auto_enrollment": true,
    "max_enrollment": 30,
    "max_waitlist": 15,
    "c_instructor_id": "100",
}
```



POST http://localhost:5000/registrar/new_class

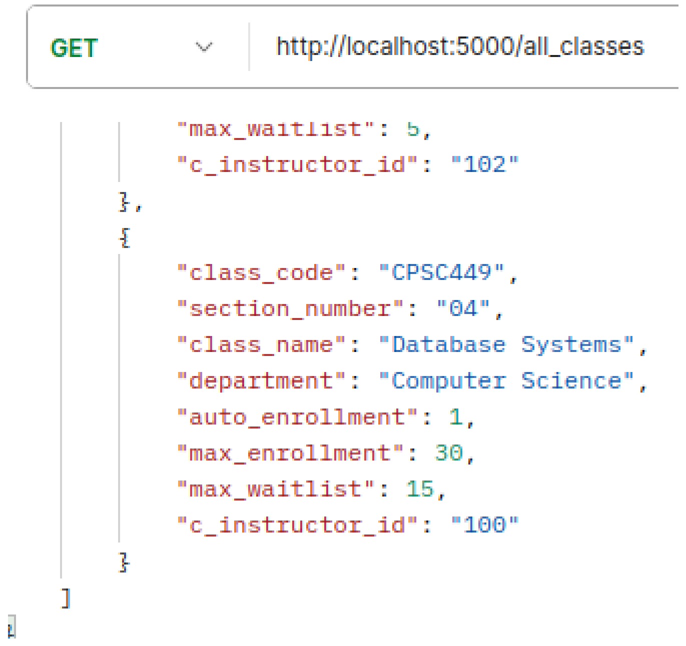
```
{
    "class_code": "CPSC449",
    "section_number": "04",
    "class_name": "Database Systems",
    "department": "Computer Science",
    "auto_enrollment": true,
    "max_enrollment": 30,
    "max_waitlist": 15,
    "c_instructor_id": "100"
}
```

Response:



```
{
    "detail": "New class successfully added."
}
```

After: GET http://localhost:5000/all_classes

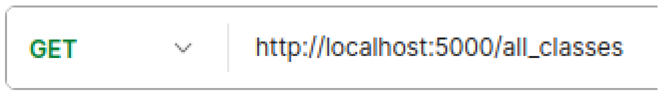


GET http://localhost:5000/all_classes

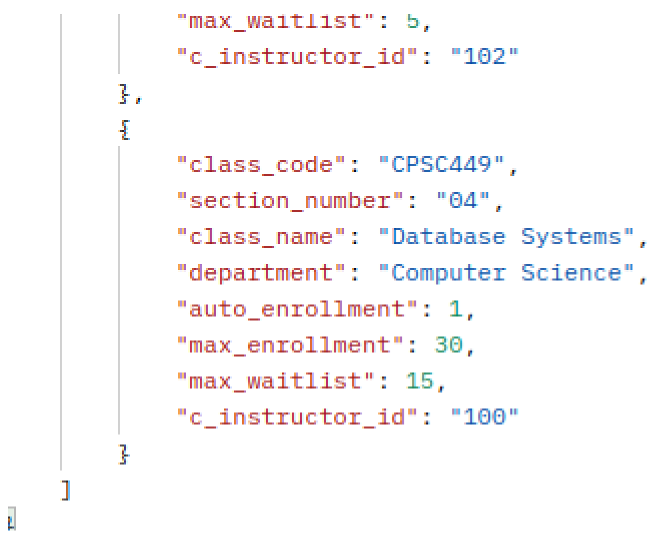
```
            "max_waitlist": 5,
            "c_instructor_id": "102"
        },
        {
            "class_code": "CPSC449",
            "section_number": "04",
            "class_name": "Database Systems",
            "department": "Computer Science",
            "auto_enrollment": 1,
            "max_enrollment": 30,
            "max_waitlist": 15,
            "c_instructor_id": "100"
        }
    ]
}
```

Query 8: Registrar can remove existing sections

Before: GET http://localhost:5000/all_classes

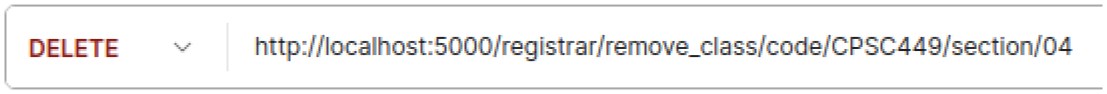


GET http://localhost:5000/all_classes



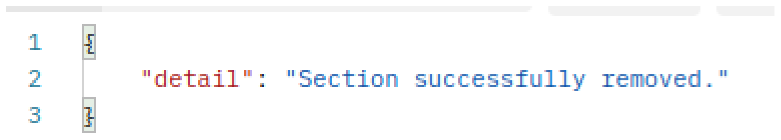
```
            "max_waitlist": 5,
            "c_instructor_id": "102"
        },
        {
            "class_code": "CPSC449",
            "section_number": "04",
            "class_name": "Database Systems",
            "department": "Computer Science",
            "auto_enrollment": 1,
            "max_enrollment": 30,
            "max_waitlist": 15,
            "c_instructor_id": "100"
        }
    ]
}
```

Request: DELETE http://localhost:5000/registrar/remove_class/code/CPSC449/section/04



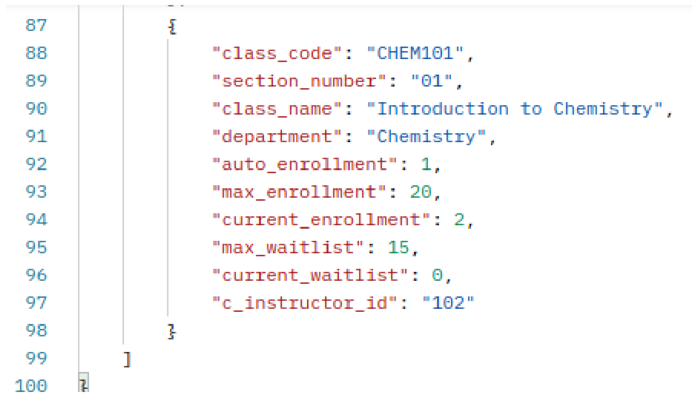
DELETE http://localhost:5000/registrar/remove_class/code/CPSC449/section/04

Response:



```
{
    "detail": "Section successfully removed."
}
```

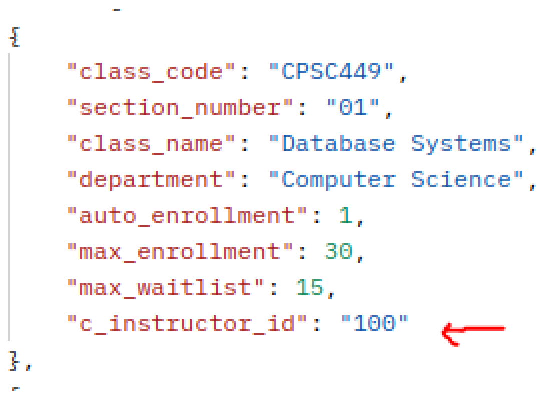
After: GET http://localhost:5000/all_classes



```
        {
            "class_code": "CHEM101",
            "section_number": "01",
            "class_name": "Introduction to Chemistry",
            "department": "Chemistry",
            "auto_enrollment": 1,
            "max_enrollment": 20,
            "current_enrollment": 2,
            "max_waitlist": 15,
            "current_waitlist": 0,
            "c_instructor_id": "102"
        }
    ]
}
```

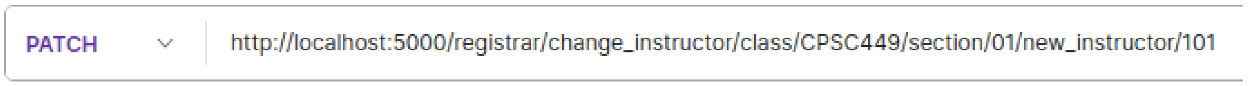
Query 9: Registrar can change instructor for a section

Before: GET http://localhost:5000/all_classes



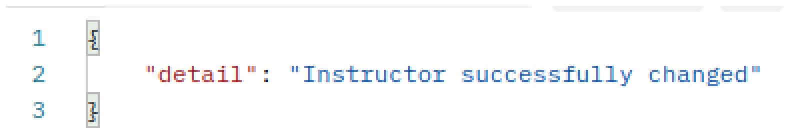
```
{
    "class_code": "CPSC449",
    "section_number": "01",
    "class_name": "Database Systems",
    "department": "Computer Science",
    "auto_enrollment": 1,
    "max_enrollment": 30,
    "max_waitlist": 15,
    "c_instructor_id": "100"
},
```

Request: PATCH
http://localhost:5000/registrar/change_instructor/class/CPSC449/section/01/new_instructor/101



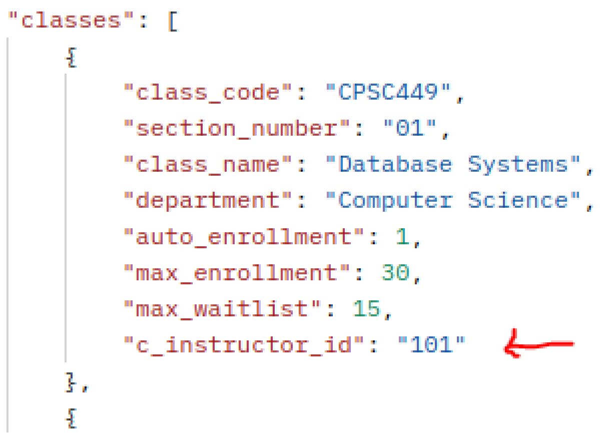
```
PATCH    http://localhost:5000/registrar/change_instructor/class/CPSC449/section/01/new_instructor/101
```

Response:



```
{
    "detail": "Instructor successfully changed"
}
```

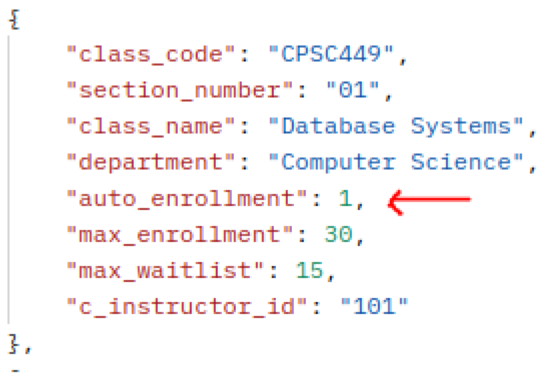
After: GET http://localhost:5000/all_classes



```
"classes": [
    {
        "class_code": "CPSC449",
        "section_number": "01",
        "class_name": "Database Systems",
        "department": "Computer Science",
        "auto_enrollment": 1,
        "max_enrollment": 30,
        "max_waitlist": 15,
        "c_instructor_id": "101"
    },
    {
```

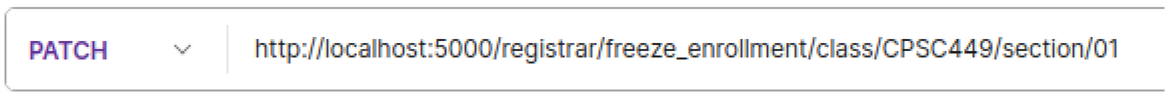
Query 10: Freeze automatic enrollment from waiting lists (e.g., during the second week of classes)

Before: GET http://localhost:5000/all_classes



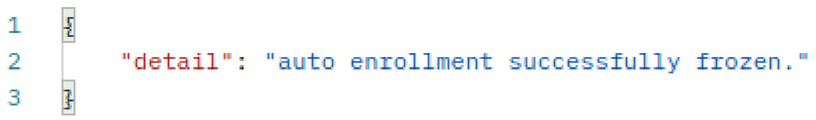
```
{
    "class_code": "CPSC449",
    "section_number": "01",
    "class_name": "Database Systems",
    "department": "Computer Science",
    "auto_enrollment": 1,
    "max_enrollment": 30,
    "max_waitlist": 15,
    "c_instructor_id": "101"
},
```

Request: PATCH http://localhost:5000/registrar/freeze_enrollment/class/CPSC449/section/01



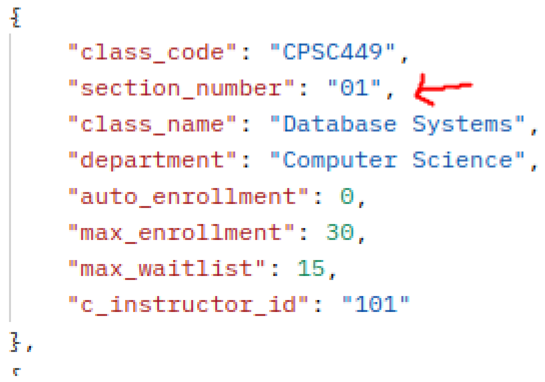
```
PATCH   http://localhost:5000/registrar/freeze_enrollment/class/CPSC449/section/01
```

Response:



```
{
    "detail": "auto enrollment successfully frozen."
}
```

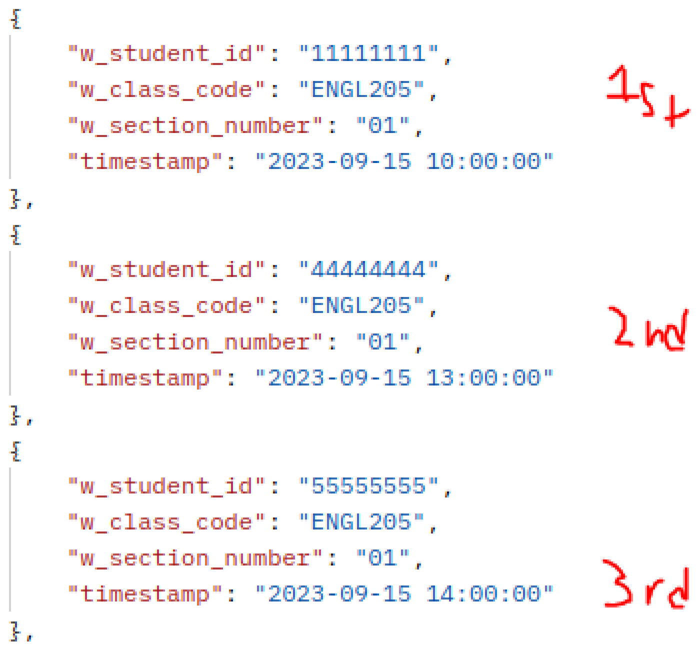
After: GET http://localhost:5000/all_classes



```
{
    "class_code": "CPSC449",
    "section_number": "01",
    "class_name": "Database Systems",
    "department": "Computer Science",
    "auto_enrollment": 0,
    "max_enrollment": 30,
    "max_waitlist": 15,
    "c_instructor_id": "101"
},
```

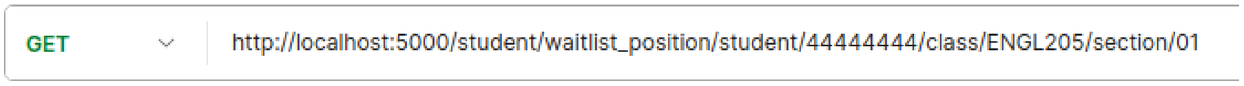
Query 11: Student can view their current position on the waiting list

Before: GET http://localhost:5000/waitlist



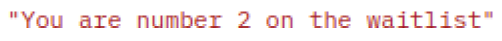
```
{
    "w_student_id": "11111111",
    "w_class_code": "ENGL205",
    "w_section_number": "01",
    "timestamp": "2023-09-15 10:00:00"
},
{
    "w_student_id": "44444444",
    "w_class_code": "ENGL205",
    "w_section_number": "01",
    "timestamp": "2023-09-15 13:00:00"
},
{
    "w_student_id": "55555555",
    "w_class_code": "ENGL205",
    "w_section_number": "01",
    "timestamp": "2023-09-15 14:00:00"
},
```

Request: GET
http://localhost:5000/student/waitlist_position/student/44444444/class/ENGL205/section/01



```
http://localhost:5000/student/waitlist_position/student/44444444/class/ENGL205/section/01
```

Response:

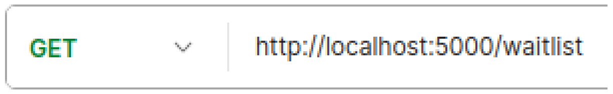


```
"You are number 2 on the waitlist"
```

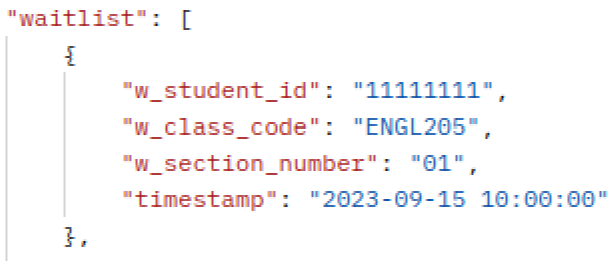
Query 12: Student can remove themselves from a waiting list

Example: DELETE
http://localhost:5000/student/remove_from_waitlist/student/11111111/class/MATH101/section/02

Before: GET http://localhost:5000/waitlist

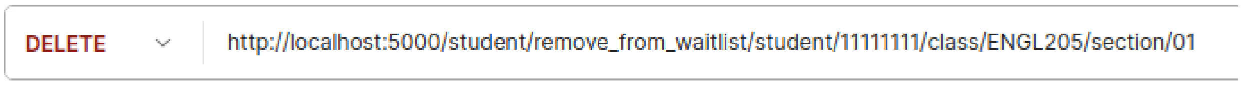


```
http://localhost:5000/waitlist
```



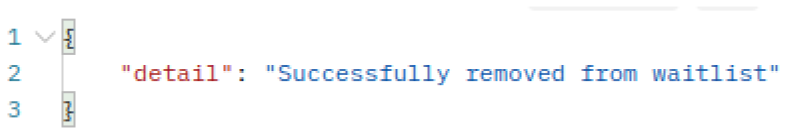
```
"waitlist": [
    {
        "w_student_id": "11111111",
        "w_class_code": "ENGL205",
        "w_section_number": "01",
        "timestamp": "2023-09-15 10:00:00"
    },
```

Request: DELETE
http://localhost:5000/student/remove_from_waitlist/student/11111111/class/ENGL205/section/01



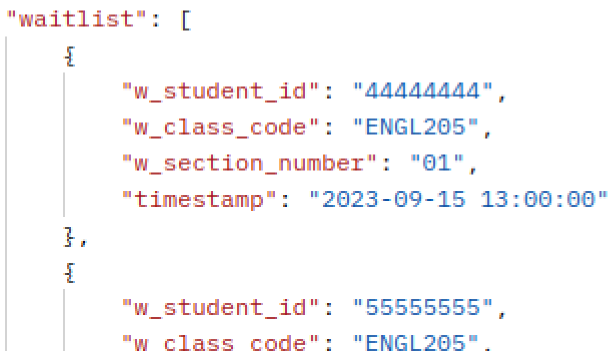
```
http://localhost:5000/student/remove_from_waitlist/student/11111111/class/ENGL205/section/01
```

Response:



```
{
    "detail": "Successfully removed from waitlist"
}
```

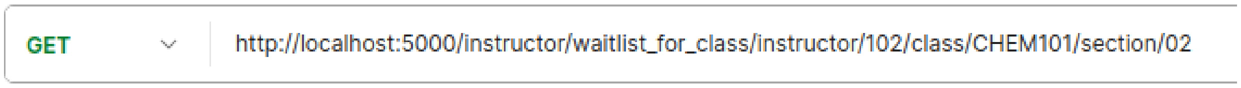
After: GET http://localhost:5000/waitlist



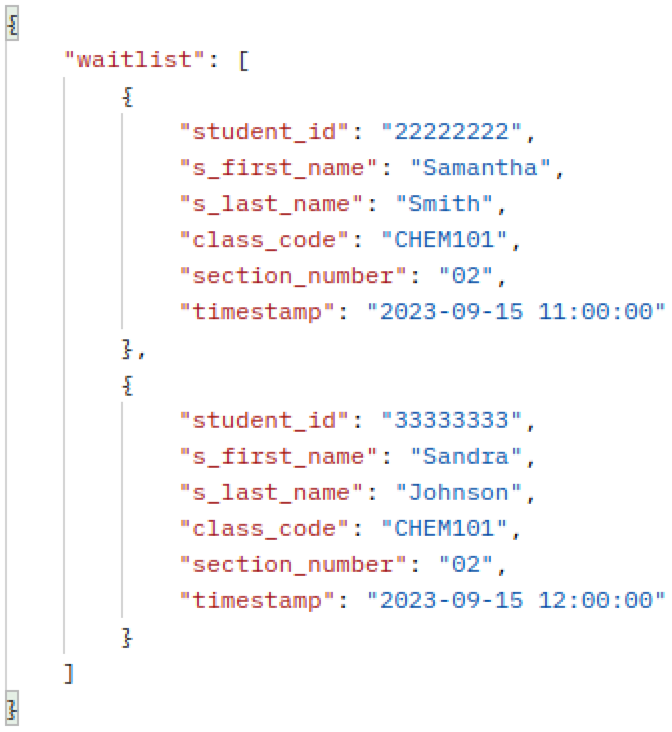
```
"waitlist": [
    {
        "w_student_id": "44444444",
        "w_class_code": "ENGL205",
        "w_section_number": "01",
        "timestamp": "2023-09-15 13:00:00"
    },
    {
        "w_student_id": "55555555",
        "w class code": "ENGL205".
```

Query 13: Instructor can view the current waiting list for their course

Request: GET
http://localhost:5000/instructor/waitlist_for_class/instructor/102/class/CHEM101/section/02


GET http://localhost:5000/instructor/waitlist_for_class/instructor/102/class/CHEM101/section/02

Response:



```
{
    "waitlist": [
        {
            "student_id": "22222222",
            "s_first_name": "Samantha",
            "s_last_name": "Smith",
            "class_code": "CHEM101",
            "section_number": "02",
            "timestamp": "2023-09-15 11:00:00"
        },
        {
            "student_id": "33333333",
            "s_first_name": "Sandra",
            "s_last_name": "Johnson",
            "class_code": "CHEM101",
            "section_number": "02",
            "timestamp": "2023-09-15 12:00:00"
        }
    ]
}
```