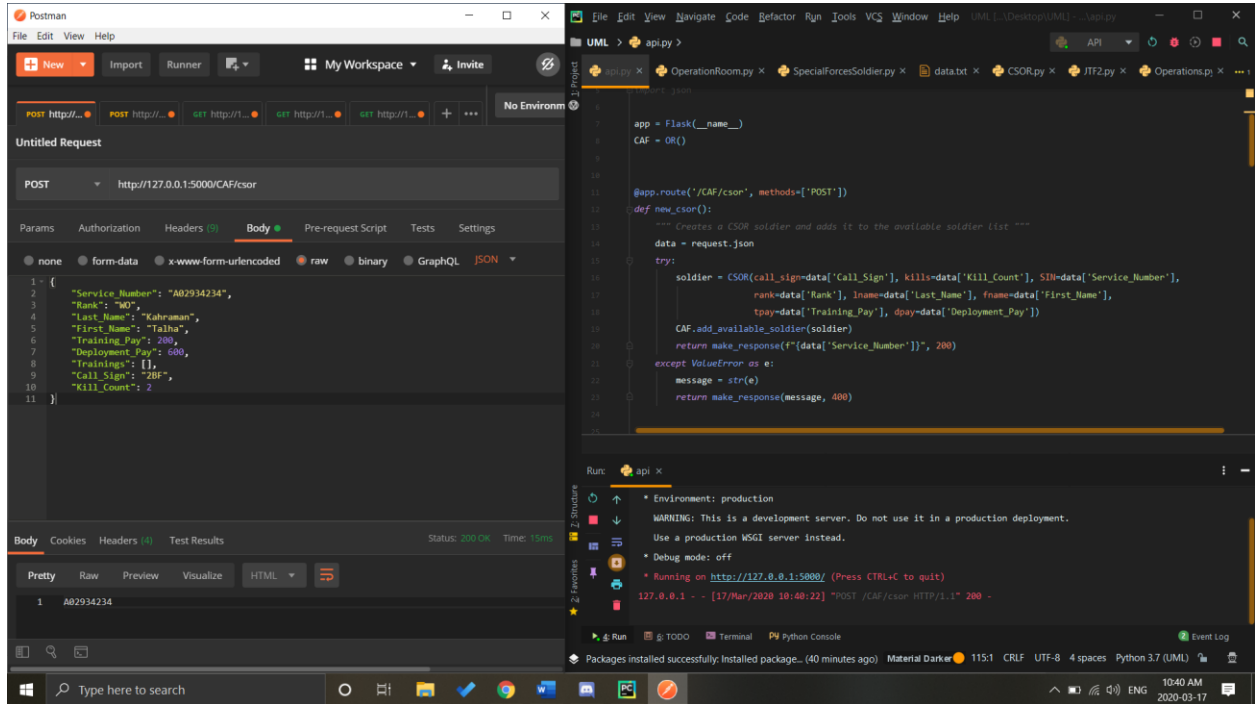


The Screenshots for the API Endpoints

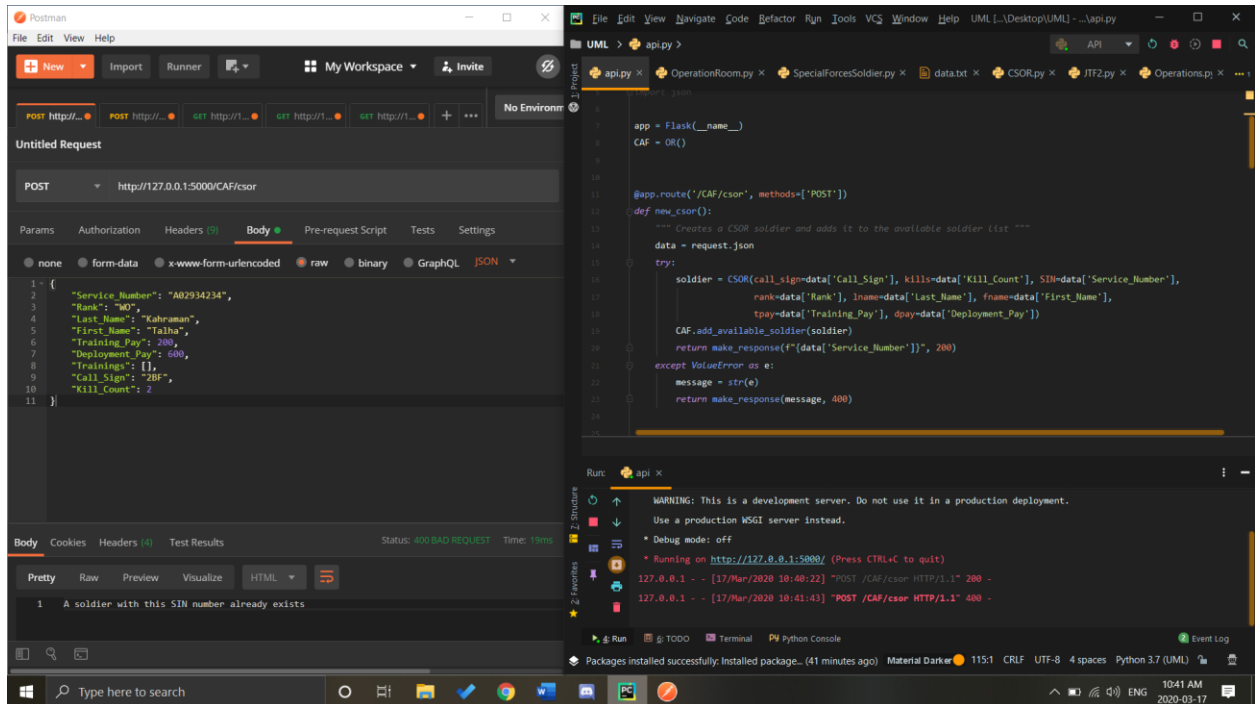
By: Sheikh Billah (Alif) & Shabnam Hashimi

POST /entitymanager/entity1

- Success

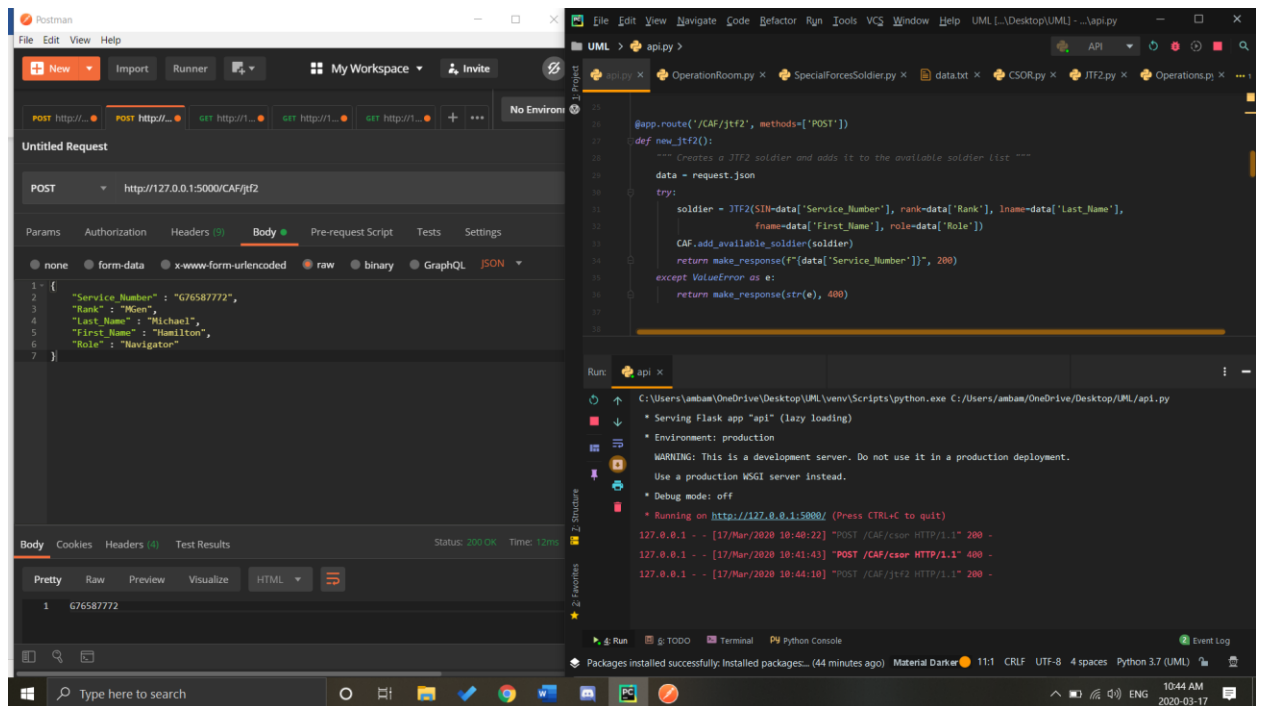


- Error

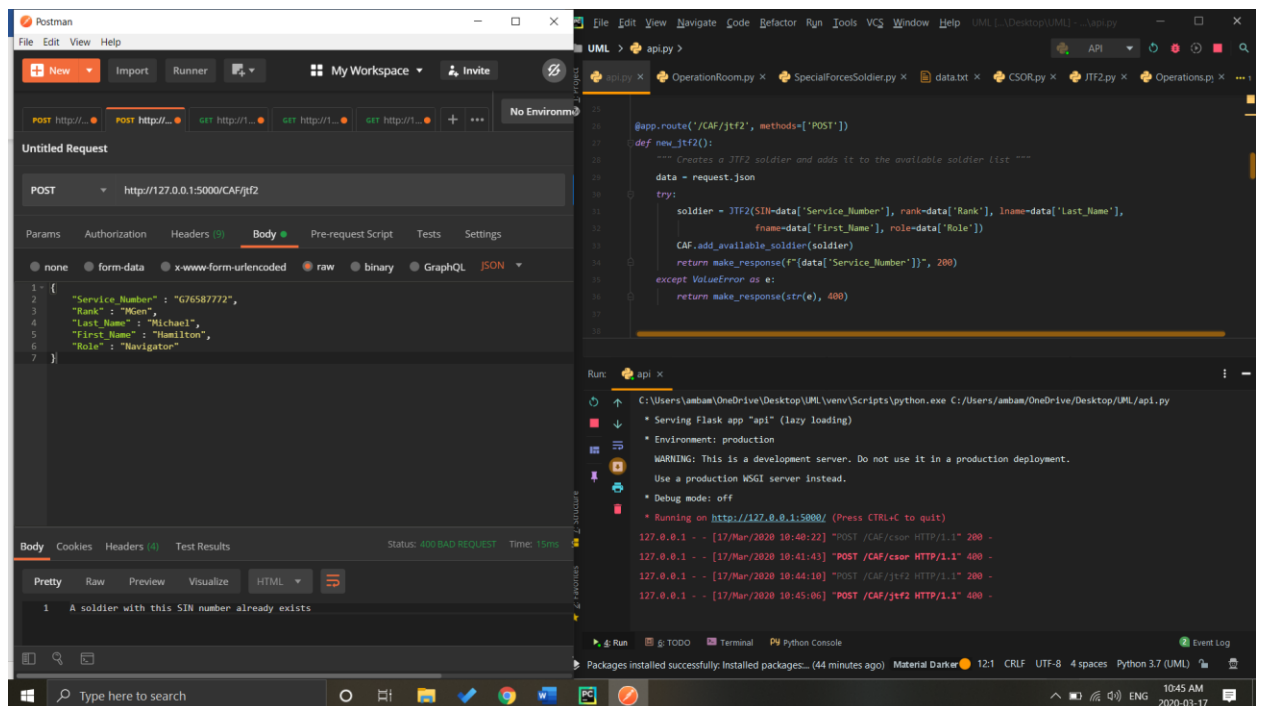


POST /entitymanager/entity2

- Success



- Error



PUT /entitymanager/entities/<entity_id>

- Success

The screenshot displays two windows side-by-side. On the left is the Postman application, showing a PUT request to `http://127.0.0.1:5000/CAF/soldier/G76587772` with a JSON body. The response status is 200 OK. On the right is the UML IDE, showing the Python code for the `update_soldier` endpoint. The code includes logic to find a soldier by ID and update their information. The terminal at the bottom shows the request being processed successfully.

```
PUT http://127.0.0.1:5000/CAF/soldier/G76587772
```

```
{
  "Role": "Night Nav",
  "Missions": 112,
  "Rank": "Gen",
  "Last_Name": "Hamilton",
  "First_Name": "Michael",
  "Training_Pay": 599,
  "Deployment_Pay": 799
}
```

```
def update_soldier(SIN):
    """ Finds a soldier by the service number and updates the rest of the info """
    soldier = CAF.get_soldier_by_ID(SIN)
    data = request.json
    try:
        if not soldier:
            raise ValueError("No soldier with the following the Service Number exists")
        if soldier.get_division() == "CSOR":
            soldier.update_soldier_info(call_sign=data['Call_Sign'], kills=data['Kill_Count'], rank=data['Rank'],
                                       lname=data['Last_Name'], fname=data['First_Name'], tpay=data['Training_Pay'],
                                       dpay=data['Deployment_Pay'])
        else:
            soldier.update_soldier_info(role=data['Role'], missions=data['Missions'], rank=data['Rank'],
                                       lname=data['Last_Name'], fname=data['First_Name'], tpay=data['Training_Pay'],
                                       dpay=data['Deployment_Pay'])
        CAF.write_to_the_file()
        return make_response("", 200)
    except ValueError as e:
        return make_response(str(e), 404)
    except KeyError:
        return make_response("Wrong key name! Maybe you've entered wrong attributes for the soldier type", 400)
```

- Error

The screenshot displays the same Postman and UML IDE windows. In Postman, the PUT request to `http://127.0.0.1:5000/CAF/soldier/G7658777k` (note the typo in the ID) has failed with a 404 NOT FOUND status. The response body in Postman says "No soldier with the following the Service Number exists". In the UML IDE, the code is the same, but the terminal shows the error response being returned.

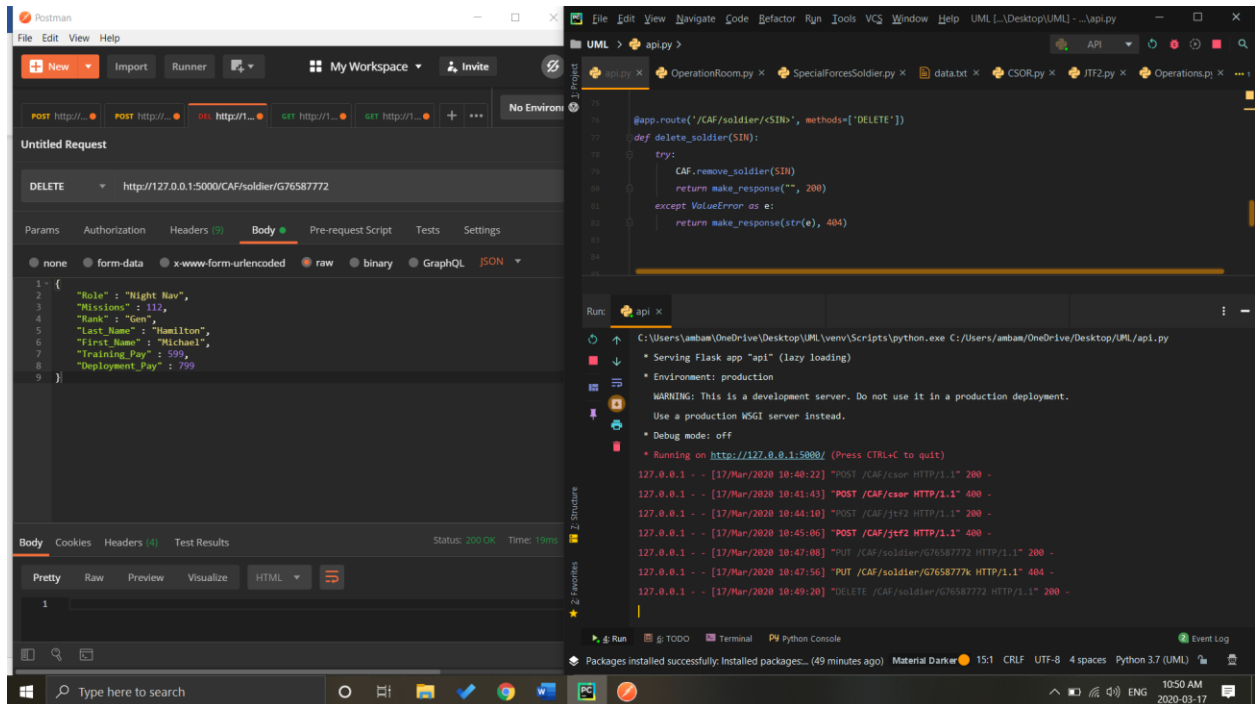
```
PUT http://127.0.0.1:5000/CAF/soldier/G7658777k
```

```
{
  "Role": "Night Nav",
  "Missions": 112,
  "Rank": "Gen",
  "Last_Name": "Hamilton",
  "First_Name": "Michael",
  "Training_Pay": 599,
  "Deployment_Pay": 799
}
```

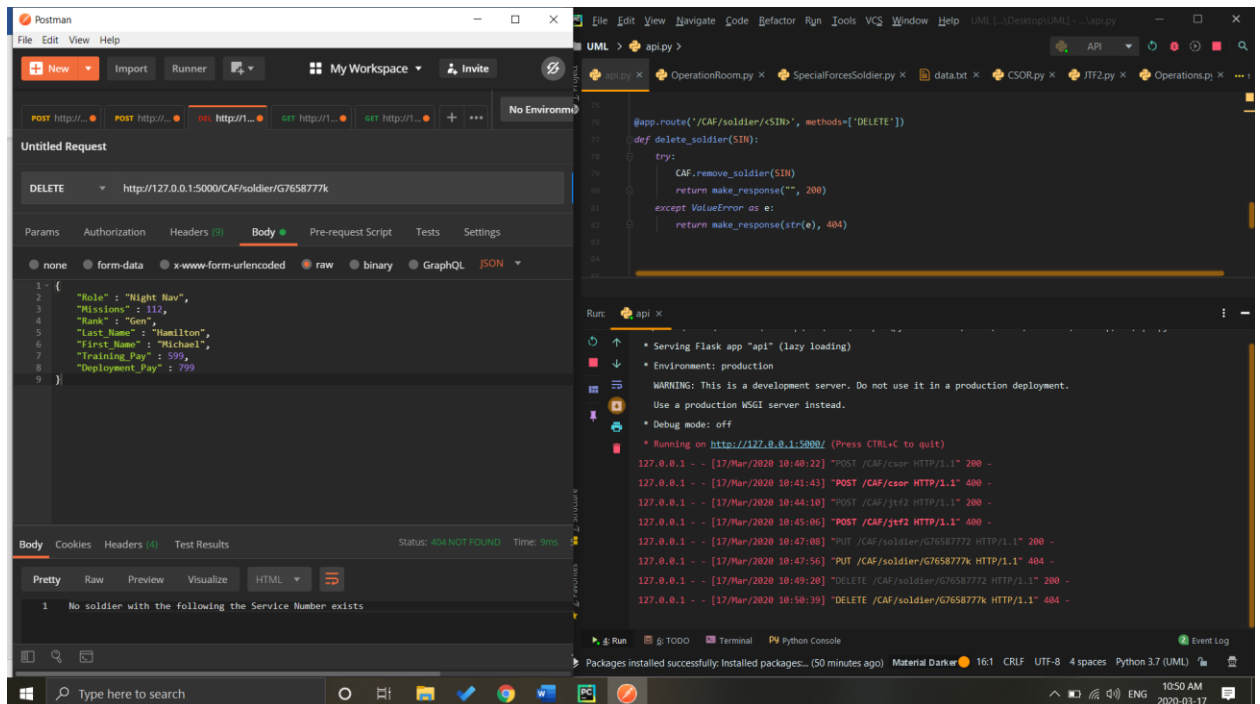
```
def update_soldier(SIN):
    """ Finds a soldier by the service number and updates the rest of the info """
    soldier = CAF.get_soldier_by_ID(SIN)
    data = request.json
    try:
        if not soldier:
            raise ValueError("No soldier with the following the Service Number exists")
        if soldier.get_division() == "CSOR":
            soldier.update_soldier_info(call_sign=data['Call_Sign'], kills=data['Kill_Count'], rank=data['Rank'],
                                       lname=data['Last_Name'], fname=data['First_Name'], tpay=data['Training_Pay'],
                                       dpay=data['Deployment_Pay'])
        else:
            soldier.update_soldier_info(role=data['Role'], missions=data['Missions'], rank=data['Rank'],
                                       lname=data['Last_Name'], fname=data['First_Name'], tpay=data['Training_Pay'],
                                       dpay=data['Deployment_Pay'])
        CAF.write_to_the_file()
        return make_response("", 200)
    except ValueError as e:
        return make_response(str(e), 404)
    except KeyError:
        return make_response("Wrong key name! Maybe you've entered wrong attributes for the soldier type", 400)
```

DELETE /entitymanager/entities/<entity_id>

- Success



- Error



GET /entitymanager/entities/<entity_id>

- Success

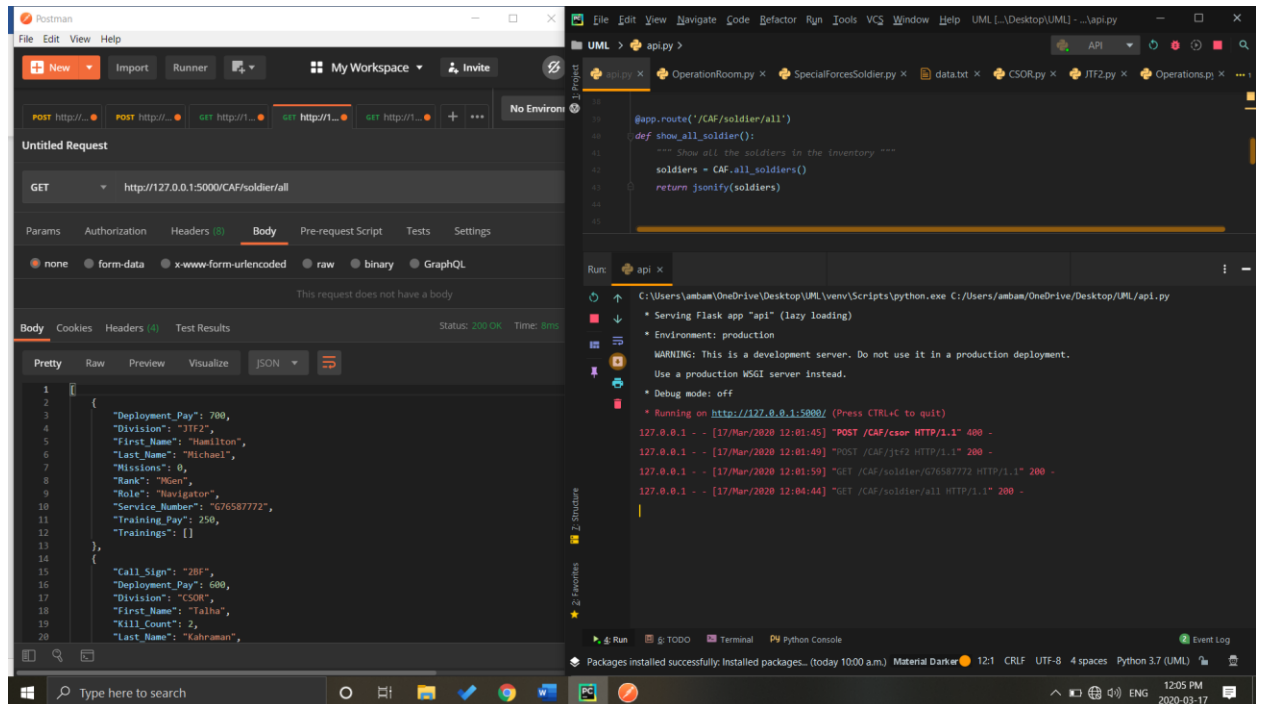
The screenshot shows two windows. On the left is Postman, displaying a GET request to `http://127.0.0.1:5000/CAF/soldier/G76587772`. The response body is a JSON object with details for a soldier named Michael Hamilton, including his role, missions, rank, first name, last name, training pay, deployment pay, and service number. On the right is the UML IDE, showing the Python code for the `find_soldier` function, which uses `CAF.get_soldier_by_ID` to retrieve the soldier's data and returns it as a JSON dictionary. The terminal at the bottom shows the server running on `http://127.0.0.1:5000/` and the successful GET request.

- Error

The screenshot shows two windows. On the left is Postman, displaying a GET request to `http://127.0.0.1:5000/CAF/soldier/G7658777k`. The response status is `404 NOT FOUND` with the message "No soldier with the following the Service Number exists". On the right is the UML IDE, showing the same Python code for the `find_soldier` function. The terminal at the bottom shows the server running on `http://127.0.0.1:5000/` and the error response for the GET request.

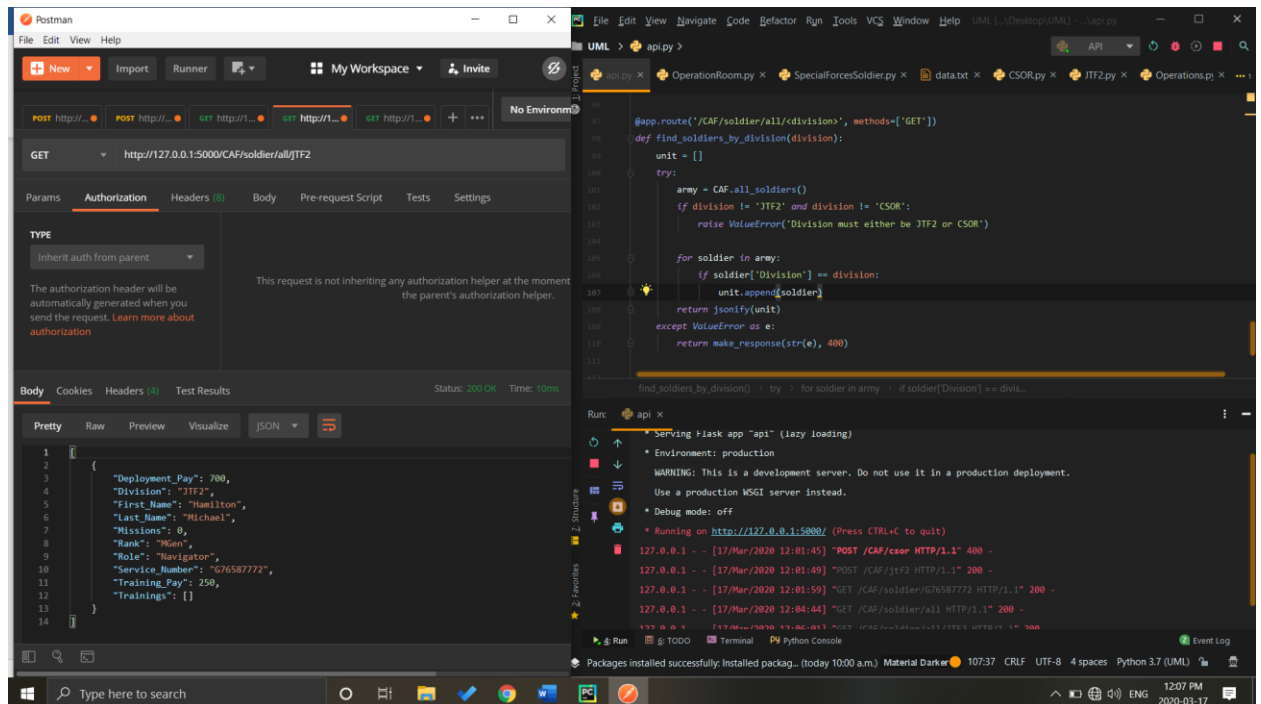
GET /entitymanager/entities/all

- Success



GET /entitymanager/entities/all/<type>

- Success



GET /entitymanager/entities/stats

- Success

The screenshot displays two windows. The left window is Postman, showing a GET request to `http://127.0.0.1:5000/CAF/soldier/stats` with a status of 200 OK and a response body containing JSON data.

The right window is a Python IDE (UML) showing the Flask application code and the terminal output.

Postman Request Details:

- Method: GET
- URL: `http://127.0.0.1:5000/CAF/soldier/stats`
- Status: 200 OK
- Time: 7ms
- Body (JSON):

```
{  "available_CSOR": 1,  "available_JTF2": 1,  "currently_deployed": 0}
```

Python Code (api.py):

```
@app.route('/CAF/soldier/stats', methods=['GET'])def get_stats():    stats = CAF.stats()    return jsonify(stats)
```

Terminal Output:

```
C:\Users\amban\OneDrive\Desktop\UML\venv\Scripts\python.exe C:/Users/amban/OneDrive/Desktop/UML/api.py
* Serving Flask app "api" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [17/Mar/2020 12:01:45] "POST /CAF/csor HTTP/1.1" 400 -
127.0.0.1 - - [17/Mar/2020 12:01:49] "POST /CAF/jtf2 HTTP/1.1" 200 -
127.0.0.1 - - [17/Mar/2020 12:01:59] "GET /CAF/soldier/G76587772 HTTP/1.1" 200 -
127.0.0.1 - - [17/Mar/2020 12:04:44] "GET /CAF/soldier/all HTTP/1.1" 200 -
127.0.0.1 - - [17/Mar/2020 12:06:01] "GET /CAF/soldier/all/JTF2 HTTP/1.1" 200 -
127.0.0.1 - - [17/Mar/2020 12:06:01] "GET /CAF/soldier/stats HTTP/1.1" 200 -
```