LAB 02 – **Fast API in Data Engineering**

Introduction

FastAPI is a high-performing web framework for building APIs with Python 3.7+ based on standard Python type hints. It helps developers build applications quickly and efficiently.

FastAPI is built on top of the Starlette web server and includes features that make building web applications easier, such as automatic data validation, error handling, and interactive API docs.

Please keep in mind that this Hands-on exercise is executed within a Jupyter Notebook environment. If you haven't configured Jupyter Notebook on your system. Please ensure that Jupyter Notebook has been installed/configured as explained/demonstrated during the pre-requisite training.

Let's start the implementation

1) Install the Fast API and UVICorn package

Before starting the demo, you will have to install the package. Run the below command to install the Fast API and Uvicorn package.

```
# install fastapi

pip install fastapi

# install uvicorn

pip install uvicorn
```

** Now that the package is successfully installed, we will start the API development using FAST API web framework.

2) Create a Python file and define all the required entities – (name the file as **Entities.py)**

Import the below packages

```python
from typing import List, Optional
from uuid import UUID, uuid4
from pydantic import BaseModel
from enum import Enum
```

typing package is used to define List and Optional data types in the program.

uuid package is used to generate unique identification number in the program.

pydantic package is the most widely used data validation library for Python using BaseModel class.

enum package is used to define Enum (Enumeration) in the program

Now create an Enum class called **Gender** like shown below.

```python
class Gender(str, Enum):
 male = "Male"
 female = "Female"
```

Additionally create an Enum class called **Role** like shown below.

```python
class Role (str, Enum):
 architect = "Architect"
 manager = "Manager"
 lead     = “Technical Lead”
 developer = "Developer"
```

Now let’s create a class called ‘**Employee’** like shown below and import **BaseModel** class for the data validation purpose.

```python
class Employee(BaseModel):
 id: Optional[UUID] = uuid4()
 first_name: str
 last_name: str
 gender: Gender
 roles: List[Role]
```

4) As we have the entities python module created successfully, lets create a Python file and define all the API methods – (name the file as **Main.py)**

```python
from typing import List
from uuid import UUID,uuid4
from fastapi import FastAPI
from fastapi import HTTPException
from entities import Gender, Role, Employee
```

typing package is used to define List and Optional data types in the program.

uuid package is used to generate unique identification number in the program.

fastapi package is the web framework to develop API's which contains class FastAPI and HTTPException which is used for HTTP exception handling to raise errors

entities package is used to import the custom Enums Gender and Role and Class Employee into the main python file.

First let's create an instance for Fast API like below

```
app = FastAPI()
```

Now let's create a db to have some sample employee records and store it on a list like below

```
db: List [Employee] = [
 Employee (
 id=uuid4(),
 first_name="John",
 last_name="David",
 gender=Gender.male,
 roles=[Role.manager],
 ),
 Employee(
 id=uuid4(),
 first_name="Kamila",
 last_name="Parker",
 gender=Gender.female,
 roles=[Role.developer],
 ),
 Employee(
 id=uuid4(),
 first_name="James",
 last_name="Cameron",
 gender=Gender.male,
 roles=[Role.architect],
 ),
 ]
```

Now lets develop a function to get all the employees from db thru the API.

GET API Method - > Get all the list of employees

@app.get(routing url) - > This decorator enables the python method as Get Method and defines the routing url, in this case /api/v1/employees

```
#get employees
@app.get("/api/v1/employees")
async def get_employees():
    return db
```

Now let’s develop the next API method to create an employee resource on the server

POST API Method -> To add a new employee record to the db

```
# add new employee
@app.post("/api/v1/employees")
async def create_employee(employee: Employee):
 db.append(employee)
 return {"id": employee.id}
```

Now let’s develop a API method to delete an employee resource on the server

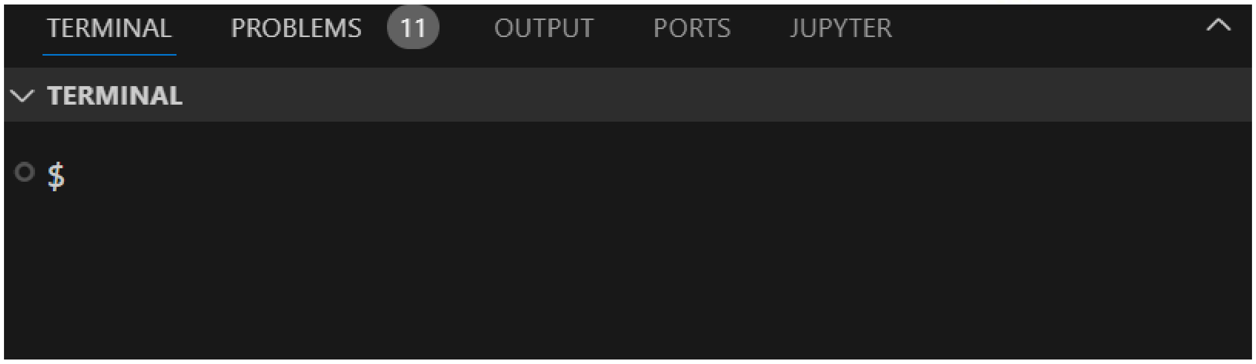
DELETE API Method - > To delete an employee record from the db for a given ID

```
#delete an employee
@app.delete("/api/v1/employees/{id}")
async def delete_employee(id: UUID):
    flag=False
    for employee in db:
        if employee.id == id:
            db.remove(employee)
            flag=True

    if flag==False:
        raise HTTPException(
    status_code=404, detail=f"Delete employee failed, id {id} not found."
    )

    return
```

Now since the API development is complete in Main.py lets host the API on a http server (Uvicorn). Click on ‘Terminal’ which will open up a terminal screen (preferably bash environment)


TERMINAL PROBLEMS 11 OUTPUT PORTS JUPYTER

TERMINAL

```
$
```

Now run the following command to build and deploy the API

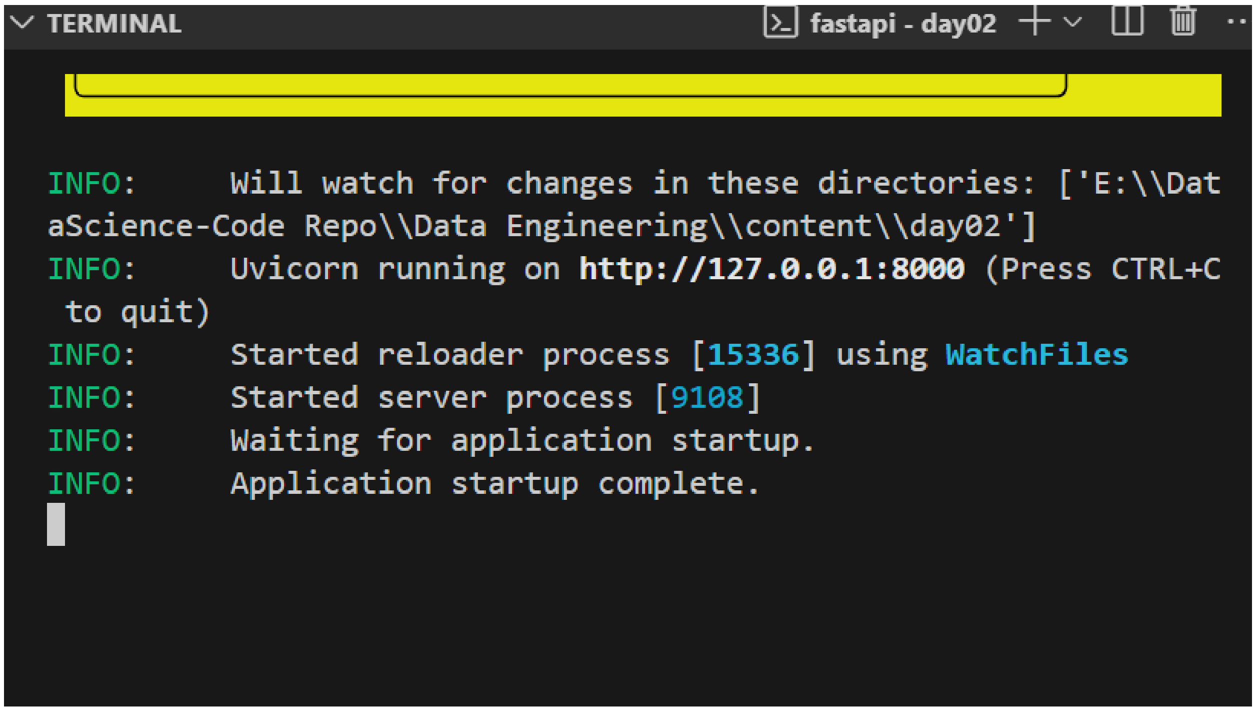
$ fastapi dev main.py

Dev indicates it's a development environment and then specify the main.py filename

Alternatively, you can the below command as well using Uvicorn

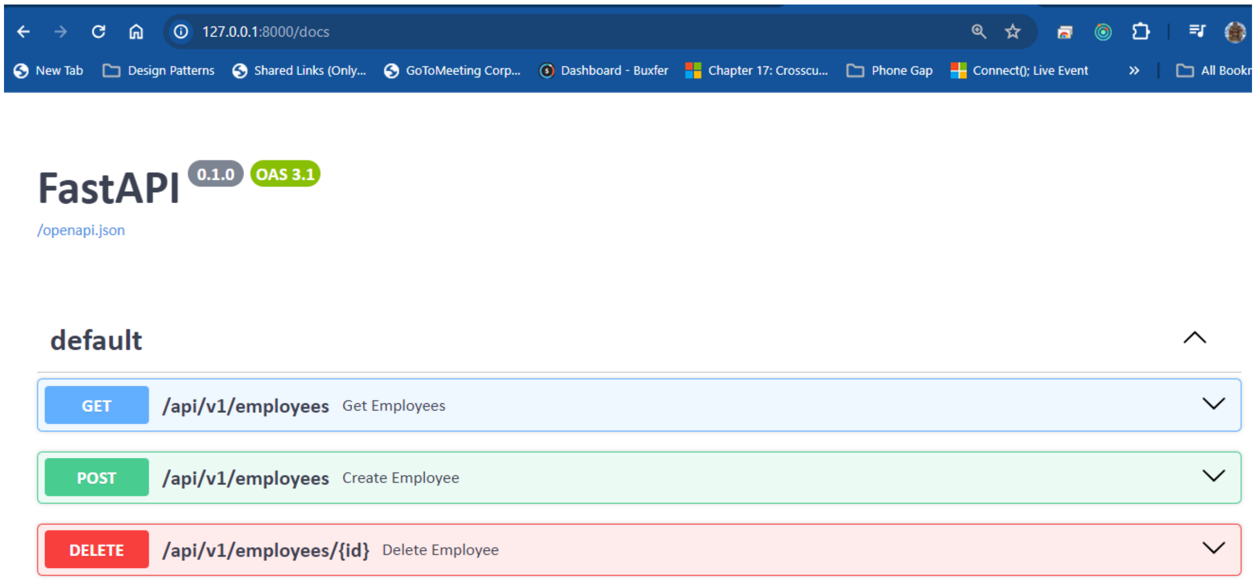
$python -m uvicorn main:app

Once successfully the API server runs, you will see the below screen and the server will be hosted on localhost(127.0.0.1) port no 8000 like http://127.0.0.1:8000


TERMINAL fastapi - day02

```
INFO:     Will watch for changes in these directories: ['E:\\Dat
aScience-Code Repo\\Data Engineering\\content\\day02']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C
 to quit)
INFO:     Started reloader process [15336] using WatchFiles
INFO:     Started server process [9108]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

Now go to browser and type in http://127.0.0.1:8000/Docs to view the swagger API documentation which will show all the API methods which are available in main.py like shown below.

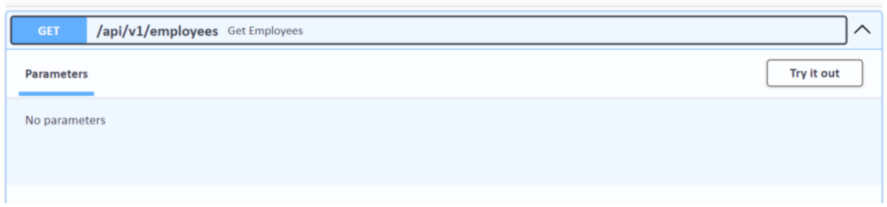


FastAPI 0.1.0 OAS 3.1
/openapi.json

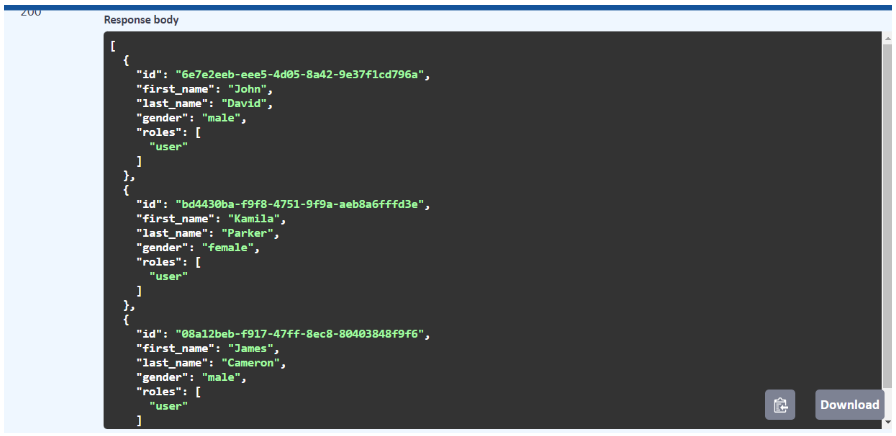
default

GET /api/v1/employees Get Employees
POST /api/v1/employees Create Employee
DELETE /api/v1/employees/{id} Delete Employee

Now click each of the listed API methods and try it out and click execute (blue button) to see the results.



GET /api/v1/employees Get Employees
Parameters
Try it out
No parameters



Response body

```
[
  {
    "id": "6e7e2eeb-eee5-4d05-8a42-9e37f1cd796a",
    "first_name": "John",
    "last_name": "David",
    "gender": "male",
    "roles": [
      "user"
    ]
  },
  {
    "id": "bd4430ba-f9f8-4751-9f9a-aeb8a6fffd3e",
    "first_name": "Kamila",
    "last_name": "Parker",
    "gender": "female",
    "roles": [
      "user"
    ]
  },
  {
    "id": "08a12beb-f917-47ff-8ec8-80403848f9f6",
    "first_name": "James",
    "last_name": "Cameron",
    "gender": "male",
    "roles": [
      "user"
    ]
```

Hurray, you have successfully completed the exercise.