Implementation

In this section we'll develop a simple REST API which implementing the OpenAPI specification.

1. Pre-Requisites
   a. Ensure that you've installed Python in your Local. To install, please go to https://www.python.org/downloads/windows/ to download the correct package. It is recommended to use the version 3 instead of version 2.

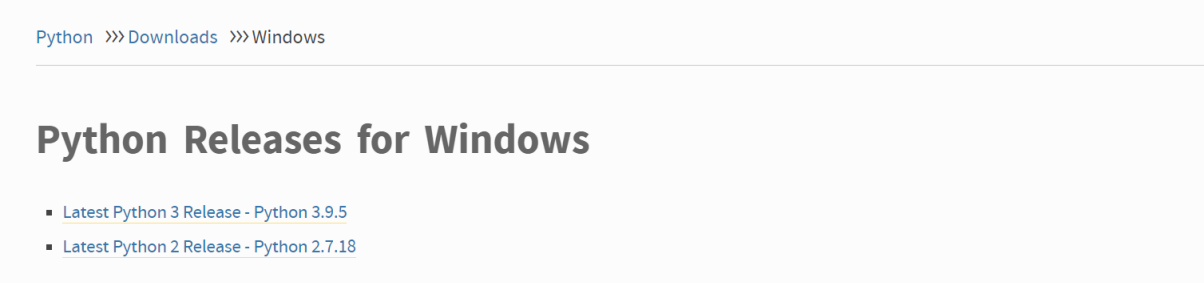

Figure 2 – Python Installer

   b. As IDE, you could use Visual Studio Code, you could download it from https://code.visualstudio.com/

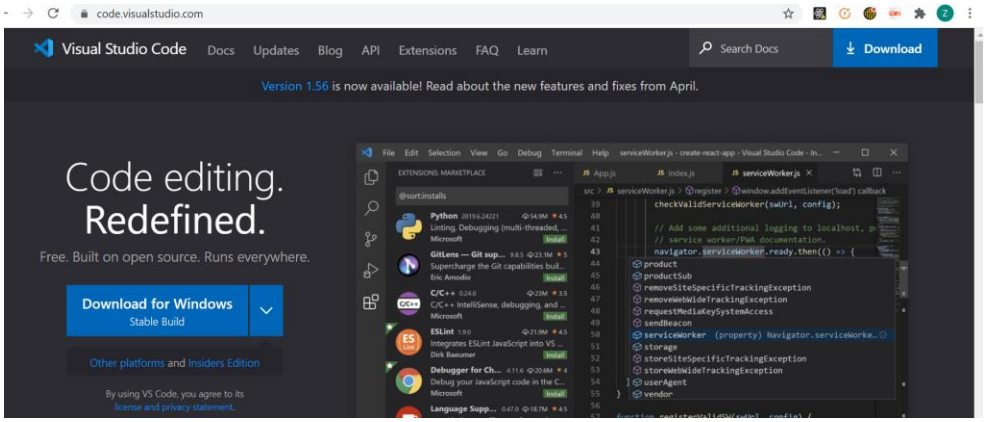

Figure 2 – Visual Studio Code Installer

   c. Ensure that python is successfully installed by typing 'python --version' in command prompt

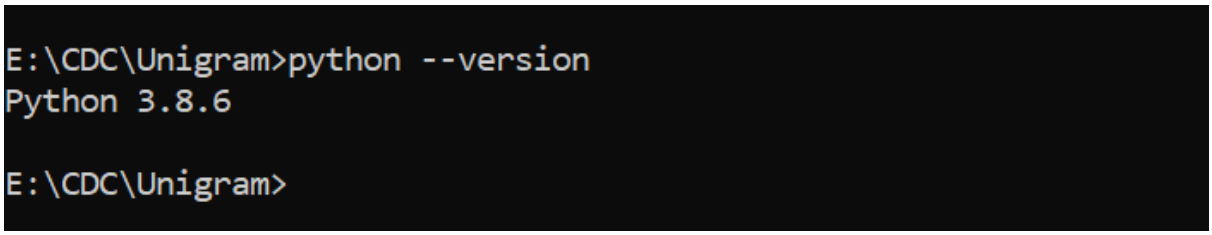

Figure 2 – Python is successfully installed

   d.

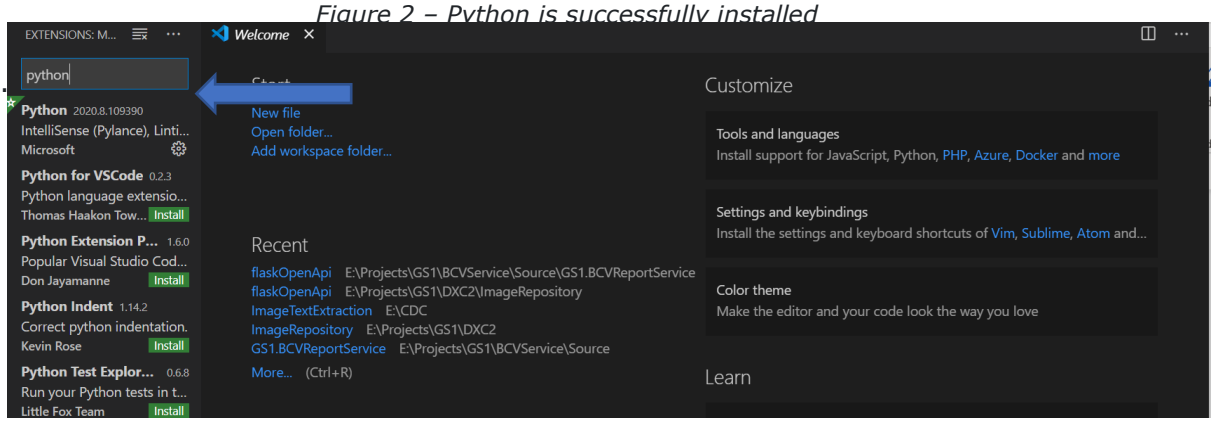

Figure 2 – Visual Studio Code already has python extension installed

2. Development

a. To start the development, you could set up the python package in a specific folder and access it through command prompt using administrator mode as we will need install few python components. We'll use pip to install the component so ensure that it is installed.

Figure 2 – Visual Studio Code already has python extension installed

b. We'll be working from virtual environment so you need to install virtualenv package.

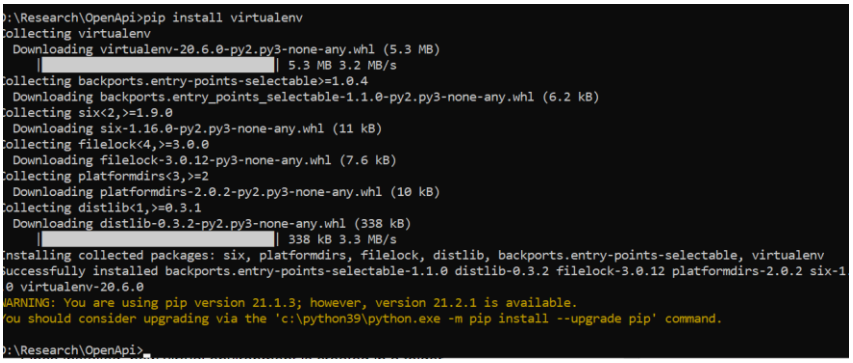

Figure 2 – Visual Studio Code already has python extension installed

c. Create virtual environment then activate it.

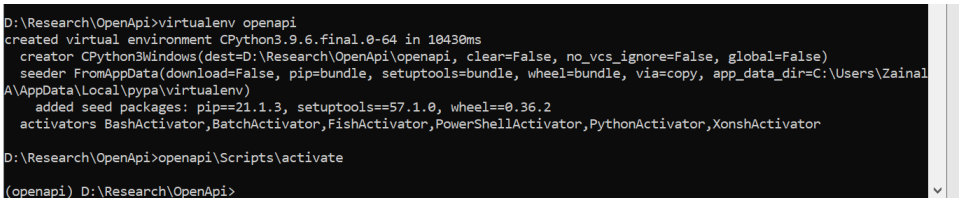

Figure 2 – Visual Studio Code already has python extension installed

d. Install flask using pip

```
(openapi) D:\Research\OpenApi>pip install flask
Collecting flask
  Downloading Flask-2.0.1-py3-none-any.whl (94 kB)
     |████████████████████████████████| 94 kB 746 kB/s
Collecting Werkzeug>=2.0
  Downloading Werkzeug-2.0.1-py3-none-any.whl (288 kB)
     |████████████████████████████████| 288 kB 3.3 MB/s
Collecting Jinja2>=3.0
  Downloading Jinja2-3.0.1-py3-none-any.whl (133 kB)
     |████████████████████████████████| 133 kB 1.7 MB/s
Collecting click>=7.1.2
  Downloading click-8.0.1-py3-none-any.whl (97 kB)
     |████████████████████████████████| 97 kB 1.6 MB/s
Collecting itsdangerous>=2.0
  Downloading itsdangerous-2.0.1-py3-none-any.whl (18 kB)
Collecting colorama
  Downloading colorama-0.4.4-py2.py3-none-any.whl (16 kB)
Collecting MarkupSafe>=2.0
  Downloading MarkupSafe-2.0.1-cp39-cp39-win_amd64.whl (14 kB)
Installing collected packages: MarkupSafe, colorama, Werkzeug, Jinja2, itsdangerous, click, flask
Successfully installed Jinja2-3.0.1 MarkupSafe-2.0.1 Werkzeug-2.0.1 click-8.0.1 colorama-0.4.4 flask-2.0.1 itsdangerous-
2.0.1
WARNING: You are using pip version 21.1.3; however, version 21.2.1 is available.
You should consider upgrading via the 'D:\Research\OpenApi\openapi\Scripts\python.exe -m pip install --upgrade pip' comm
and.

(openapi) D:\Research\OpenApi>
```

Figure 2 – Visual Studio Code already has python extension installed

e. To support the Open API, we need to use an external component called connexion. Install [connexion] using pip.

```
(openapi) D:\Research\OpenApi>pip  install connexion[swagger-ui]
Collecting connexion[swagger-ui]
  Using cached connexion-2.9.0-py2.py3-none-any.whl (84 kB)
Requirement already satisfied: jsonschema<4,>=2.5.1 in d:\research\openapi\openapi\lib\site-packages (from connexion[swa
gger-ui]) (3.2.0)
Requirement already satisfied: inflection<0.6,>=0.3.1 in d:\research\openapi\openapi\lib\site-packages (from connexion[s
wagger-ui]) (0.5.1)
Requirement already satisfied: werkzeug<2.0,>=1.0 in d:\research\openapi\openapi\lib\site-packages (from connexion[swagg
er-ui]) (1.0.1)
```

Figure 2 – Visual Studio Code already has python extension installed

f. Before starting to code, let's collect the requirement file. Use pip freeze command to collect it.

```
(openapi) D:\Research\OpenApi>pip freeze>requirements.txt

(openapi) D:\Research\OpenApi>
```

Figure 2 – Visual Studio Code already has python extension installed

requirements.txt

```
attrs==21.2.0
certifi==2021.5.30
charset-normalizer==2.0.3
click==7.1.2
clickclick==20.10.2
colorama==0.4.4
connexion==2.9.0
Flask==1.1.4
idna==3.2
inflection==0.5.1
isodate==0.6.0
itsdangerous==1.1.0
Jinja2==2.11.3
jsonschema==3.2.0
MarkupSafe==2.0.1
openapi-schema-validator==0.1.5
openapi-spec-validator==0.3.1
pyrsistent==0.18.0
PyYAML==5.4.1
requests==2.26.0
six==1.16.0
swagger-ui-bundle==0.0.8
urllib3==1.26.6
Werkzeug==1.0.1

```

Figure 2 – Visual Studio Code already has python extension installed

g. Create ‘models.py’ file

```
import json

GENDER = ["Male","Female"]
DEPARTMENTS = ["Computer Science","Environmental", "Medical Health"]

class Student:
  def __init__(self, name, gender, subject, department, grade):
    self.name = name
    self.gender = gender
    self.subject = subject
    self.department = department
    self.grade = grade
  def serialize(self):
    return {
              "name": self.name,
              "gender": self.gender,
              "subject": self.subject,
              "department": self.department,
              "grade": self.grade
            }
```

Figure 2 – models.py

h. Create ‘app.py’ file

```
import connexion
import json

from connexion.exceptions import OAuthProblem
from models import DEPARTMENTS, GENDER, Student

#token auth
TOKEN_DB = {
    'asdf1234567890': {
        'uid': 100
    }
}

#data setup
results  = dict()
results["Students"] = []
results["Students"].append(Student("Anabelle",GENDER[1],"Data
Analytics",DEPARTMENTS[0],"A"))
results["Students"].append(Student("Deniz",GENDER[0],"Data
Analytics",DEPARTMENTS[0],"A"))
results["Students"].append(Student("Devika",GENDER[1],"Waste
Treatment",DEPARTMENTS[1],"B"))
results["Students"].append(Student("Jeannette",GENDER[1],"Test
Strategy",DEPARTMENTS[0],"A"))
results["Students"].append(Student("Bob",GENDER[0],"Waste
Treatment",DEPARTMENTS[1],"C"))
results["Students"].append(Student("Jane",GENDER[1],"Biology",D
EPARTMENTS[2],"A"))
results["Students"].append(Student("Frans",GENDER[1],"Biology",
DEPARTMENTS[2],"A"))
results["Students"].append(Student("Dave",GENDER[0],"Data
Analytics",DEPARTMENTS[1],"A"))
```

```
results["Students"].append(Student("Jim",GENDER[0],"Biology",DE
PARTMENTS[2],"A"))

#api key authentication
def apikey_auth(token, required_scopes):
    info = TOKEN_DB.get(token, None)
    if not info:
        raise OAuthProblem('Invalid token')
    return info

#get all students
def get_all_students(user) -> str:
    output = []
    for student in results["Students"]:
        output.append(student.serialize())
    result = {'Students': output, 'Total': len(output)}
    return result

#get students by department
def get_students_by_department(department, user) -> str:
    output = []
    filter_result = [x for x in results["Students"] if
x.department == department]
    for student in filter_result:
        output.append(student.serialize())
    result = {'Students': output, 'Total': len(output)}
    return result

#filter set
def filter_set(data, search_string):
    def iterator_func(x):
        for v in x.values():
            if search_string in v:
                return True
        return False
    return filter(iterator_func, data)


if __name__ == '__main__':
    app = connexion.FlaskApp(__name__, port=9090)
    app.app.config['JSON_SORT_KEYS'] = False
    app.add_api('openapi.yaml', arguments={'title': 'Open API
example'})
    app.run()
```

Figure 2 – app.py

i. Create 'openapi.yaml'

```
openapi: "3.0.0"
info:
  title: Open Api Example with api key 'asdf1234567890'
  version: "1.0"
servers:
  - url: http://localhost:9090/v1.0
paths:
  '/get_all_students':
    get:
      summary: Get students
      description: Get List of students
      operationId: app.get_all_students
      responses:
```

```yaml
        '200':
          description: Return All students
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Student'
      security:
        - api_key: []
  '/get_students_by_dept/{department}':
    get:
      summary: Filter students by department
      description: Filter List of students by Department
      operationId: app.get_students_by_department
      parameters:
        - $ref: '#/components/parameters/department'
      responses:
        '200':
          description: Return students with specific department
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Student'
      security:
        - api_key: []
components:
  securitySchemes:
    api_key:
      type: apiKey
      name: X-Auth
      in: header
      x-apikeyInfoFunc: app.apikey_auth
  parameters:
    department:
        name: department
        description: Student's department
        in: path
        required: true
        example: Computer Science,Environmental,Medical Health
        schema:
          type: string
  schemas:
    Student:
      type: object
      properties:
        name:
          type: string
          description: name of student
        gender:
          type: string
          description: gender of student
        subject:
          type: string
          description: subject that students study
        department:
          type: string
          description: student's department
        grade:
          type: string
          description: student's grade
```

Figure 2 – openapi.yaml

j. Open the terminal in visual studio code and print 'python .\app.py'

```
PS D:\Research\OpenApi> python .\app.py
 * Serving Flask app 'app' (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 all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
```

Figure 2 – run the app

k. Then go to browse to 'http://localhost:9090/v1.0/ui/'
l. You could see that there are some API method information and we also could execute our API methods

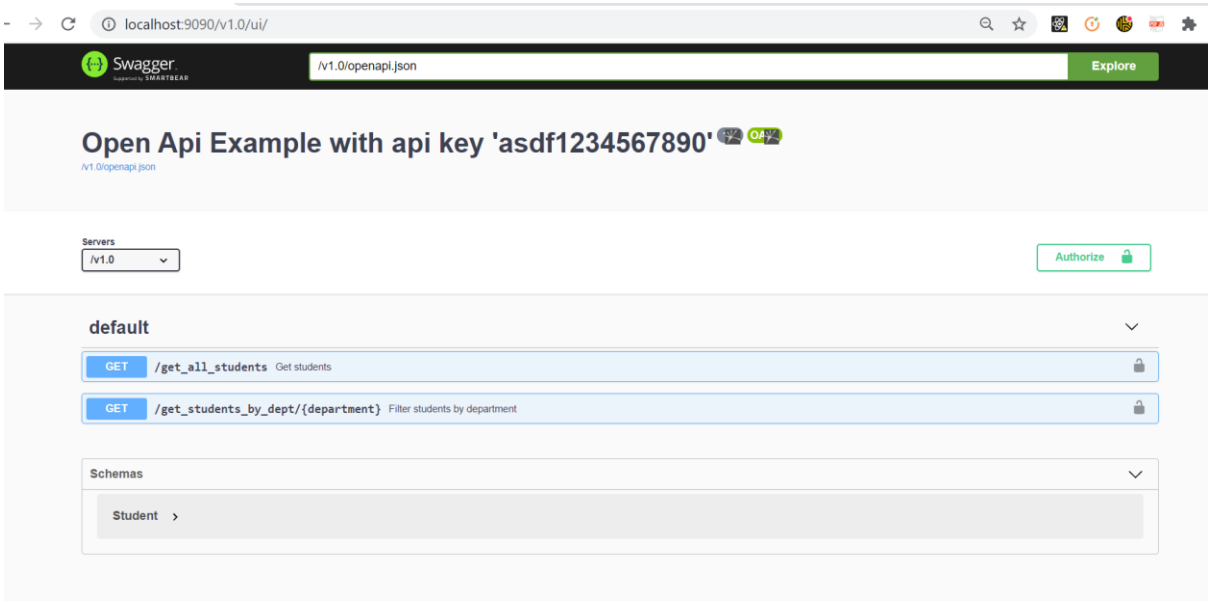

Figure 2 – Default URL page

m. Since we use API Key for authentication then we need to input the API Key by clicking Authorize button then fill the API Key.

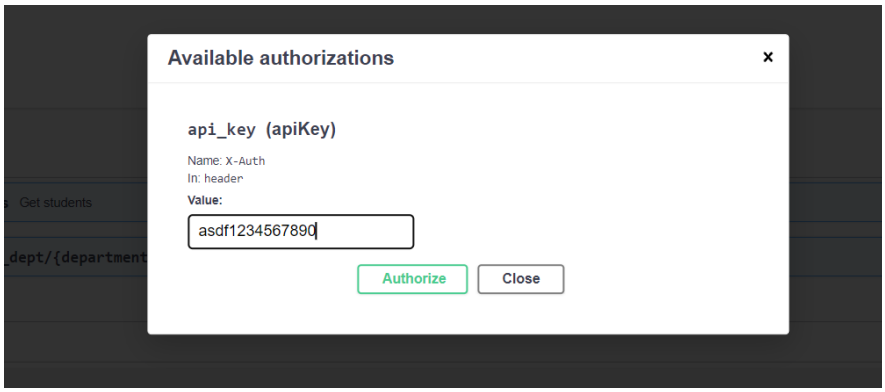

Figure 2 – Input API Key

n. Click Authorize then you are now authorized to access the API

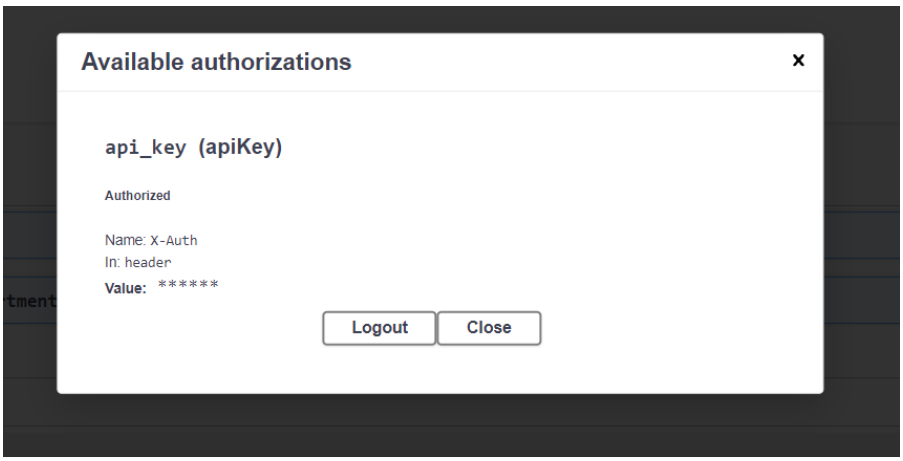

Figure 2 – You're now authorized to access the API

o. For example, you would like to filter students by department, you could use /get_students_by_department/{department} method by clicking Try it out button. Then it will make input text editable.

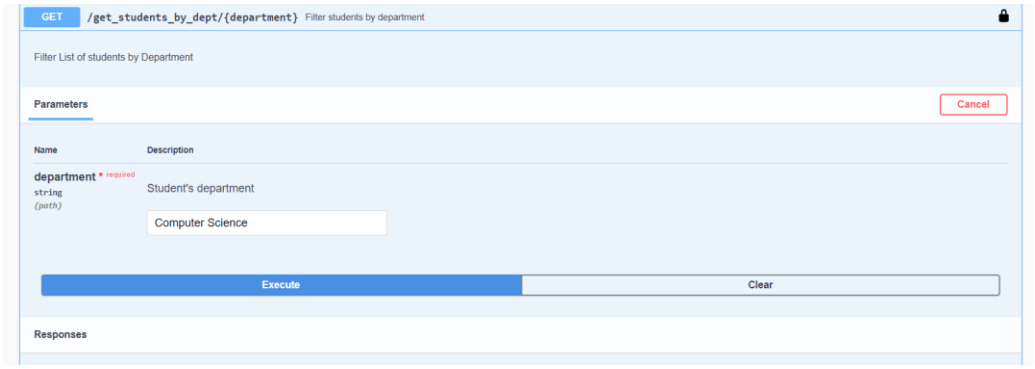

Figure 2 – Filter Students by Department example Computer Science

p. Click Execute button to retrieve the result.

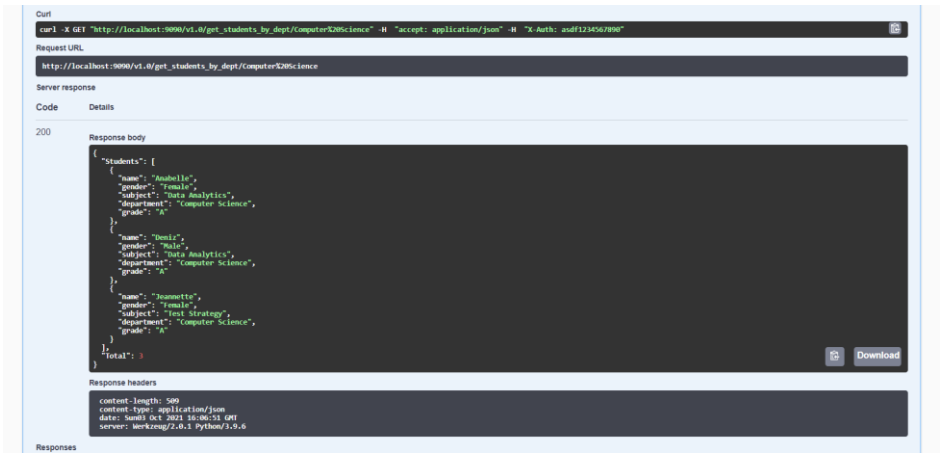

Figure 2 – The List of Computer Science Students are displayed