

Ecorus Assignment Notes: Questions and Answers

How long did it take for assignments?

For 1st assignment less than 2 hours (2 different approaches)

For 2nd assignment- took 3-4 days

In between I was not feeling well, so could not submit it

Output:

Python code and console screenshot (printing the objects) after every exercise.

Assignment 1:

Output Assignment 1 (using dictionary):

```
Person Object--> name:eduardo, age:0
Person Object--> name:ganga, age:15
Office Object--> name:ecorus, people_working: {'ganga': 15}
```

Output Assignment 1 (using list):

```
Person Object--> Person(name:eduardo, age:0)
Person Object--> Person(name:ganga, age:15)
Office Object--> Office(name:ecorus, people_working: ['Person(name:ganga, age:15)'])
```

Together outputs:

```
PS C:\Users\Gopi\Documents\Python Scripts\Ecorus\final_ecorus> python .\assignment_1_using_list.py
Person Object--> Person(name:eduardo, age:0)
Person Object--> Person(name:ganga, age:15)
Office Object--> Office(name:ecorus, people_working: ['Person(name:ganga, age:15)'])
PS C:\Users\Gopi\Documents\Python Scripts\Ecorus\final_ecorus> python .\assignment_1_using_dict.py
Person Object--> name:eduardo, age:0
Person Object--> name:ganga, age:15
Office Object--> name:ecorus, people_working: {'ganga': 15}
PS C:\Users\Gopi\Documents\Python Scripts\Ecorus\final_ecorus>
```

2nd Assignment:

Notes:

Used Python Flask Framework, sqlite3 Database, flask_restful api,
Added a component to improve requests handling, used template with HTML pages
to show the data in the front end, used postman to test APIs.

```

PS C:\Users\Gopi\Documents\Python Scripts\Ecorus\final_ecorus_assignemnt2_flask> 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: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 303-398-466
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

```

RESTFUL API End points screenshots

Note: Also attached Excel sheet , please refer

Person RESTFUL APIs				
Sl.no	Endpoints	HTTP Request Method	Description	Database Function
1	http://127.0.0.1:5000/api/persons	GET	Get (show) Persons Data	get_persons()
2	http://127.0.0.1:5000/api/persons/create	POST	Add/Create a new Person by giving following Object data (by passing in Body of postman/ via assignment1): <pre>{ "name": "eduardo", "age": 29 }</pre>	insert_person()
3	http://127.0.0.1:5000/api/persons/change-person-name	PUT	Update/Change Person name for existing/given person id(object_id): <pre>{ "person_id": 1, "name": "new_eduardo", "age": 29 }</pre>	update_person_name()
4	http://127.0.0.1:5000/api/persons/happy-birthday	PUT	Update(increase) Person age age by 1 year) for existing/given person id(object_id): <pre>{ "person_id": 1, "name": "new_eduardo", "age": 29 }</pre>	update_birthday_age

Office RESTFUL APIs				
Sl.no	Endpoints	HTTP Request Method	Description	Database Function
1	http://127.0.0.1:5000/api/office	GET	Show the Office Data - Get list of all office data from database	get_office_data()
2	http://127.0.0.1:5000/api/office/add/start-working-for	POST	Add/Create a new office data like Add person to Office (Start working for) in Json object format: <pre>{ "name": "ecorus", "people_working": { "shiva": 21 } }</pre>	insert_office()
3	http://127.0.0.1:5000/api/office/remove/finished_working_for/1	DELETE	Removes employee from office data corresponding to given last integer value (<employee ID>) in the URL Example: mention number in URL as per the employee id- - "/api/office/remove/finished_working_for/<int>	delete_employee_by_id()

Questions:

1. - What has been the more difficult part?

Was aware about RestAPI, need to research and learn about using Flask RESTFUL API, endpoints, and SQLite3 Database connections.

2. - What part of the system could be improved?

Frontend UI feature with forms could be included for easy interaction with the Application, can enhance code with better Exception handling refactor database implementation, Restful api authentication, including functional tests to test the database, api, app.

3. - How would you scale it, to be able to handle 1K calls per sec? and to handle 1M?

By using load balancer with multiple application servers. By this we can distribute/route the requests across multiple resources/server. Based on network traffic/demand we can add/ remove servers. (This also helps in resilient to failure)

4. - How would you automate the testing?

Using python Pytest module, we can prepare testcases to test App, database, API (using request library), UI etc. It can either be run as an individual tool for testing or it can be attached to Python web development frameworks like Flask, enhancing their unit test capabilities and reliability. By this way we can automate the running of tests and finally obtain test coverage report. Also we have to note that there are many other tools in the market for testing.

5. - How would you implement a continuous development system (pipelines) for this particular case?

We can use Jenkins to design CI/CD pipeline.

Jenkins is a well-known open-source continuous integration and continuous development automation tool

We need following setup to implement end to end CI/CD pipeline

- Python setup : to create code for assignments, unit tests
- API creation using flask
- setting up git repository for the project
- Docker image creation for the API then run it as container
- Jenkins server setup

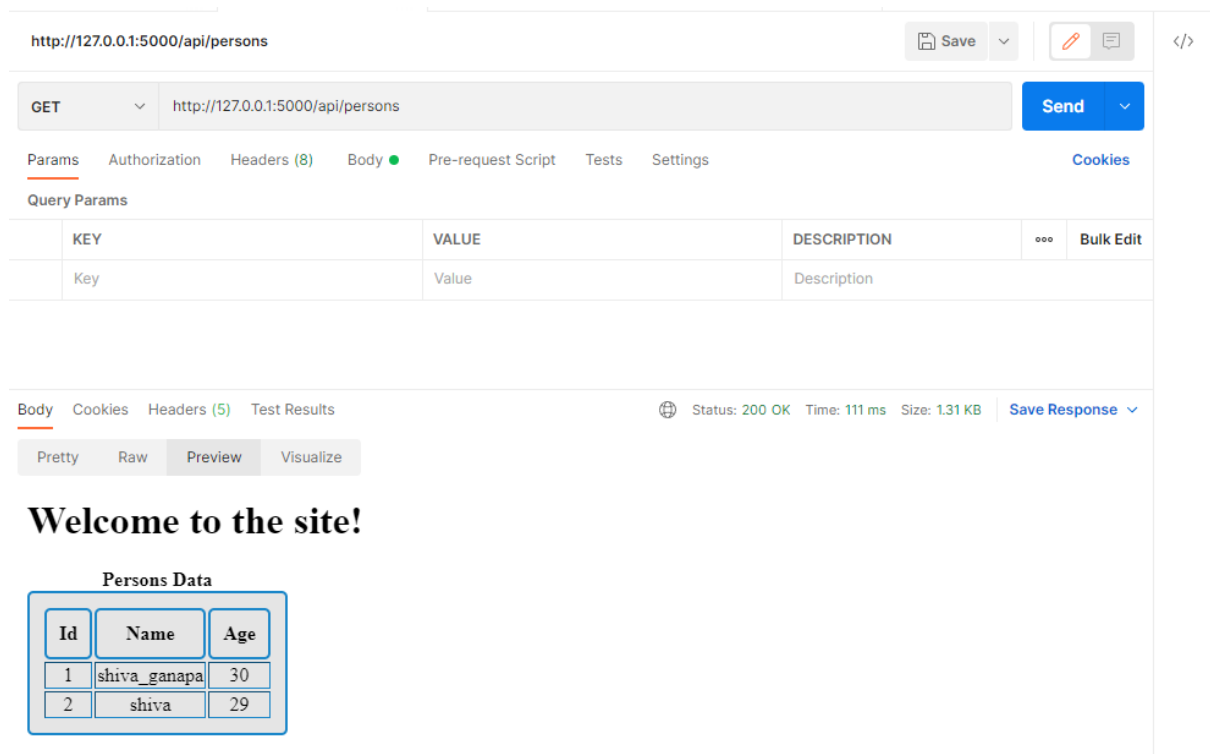
- Creating CI pipeline in Jenkins and Running it
- Testing the end to end flow

Please note that there are other tools like Azure, Travis CI etc.

OUTPUT Screenshots of Assignment 2 -

Falsk Restful API Functional Testing using Postman:

Person APIs:



http://127.0.0.1:5000/api/persons

GET http://127.0.0.1:5000/api/persons

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 111 ms Size: 1.31 KB Save Response

Pretty Raw Preview Visualize

Welcome to the site!

Persons Data

Id	Name	Age
1	shiva_ganapa	30
2	shiva	29

POST

http://127.0.0.1:5000/api/persons/create

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

1

2

3

4

5

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

Status: 200 OK

Time: 247 ms

Size: 1.15 KB

Save Response

Successfully added Person data!

Persons Data		
Id	Name	Age
3	eduardo	29

http://127.0.0.1:5000/api/persons/change-person-name

Save

PUT

http://127.0.0.1:5000/api/persons/change-person-name

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

1

2

3

4

5

6

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

Status: 200 OK

Time: 161 ms

Size: 1.17 KB

Save Response

Successfully Changed/Updated Person Name!

Persons Data		
Id	Name	Age
3	new_eduardo	29

http://127.0.0.1:5000/api/persons/happy-birthday

Save



PUT http://127.0.0.1:5000/api/persons/happy-birthday

Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Beautify

```
1 {
2   ... "person_id" : 3,
3   ... "name" : "new_eduardo",
4   ... "age" : 29
5 }
6 }
```

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 226 ms Size: 1.15 KB

Save Response

Pretty Raw Preview Visualize

Happy Birthday with new age!

Persons Data

Id	Name	Age
3	new_eduardo	30

http://127.0.0.1:5000/api/persons

Save



GET http://127.0.0.1:5000/api/persons

Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Beautify

```
1 {
2   ... "person_id" : 3,
3   ... "name" : "new_eduardo",
4   ... "age" : 29
5 }
6 }
```

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 34 ms Size: 1.47 KB

Save Response

Pretty Raw Preview Visualize

Welcome to the site!

Persons Data

Id	Name	Age
1	shiva_ganapa	30
2	shiva	29
3	new_eduardo	30

OFFICE APIS:

http://127.0.0.1:5000/api/office

GET http://127.0.0.1:5000/api/office Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results Status: 200 OK Time: 83 ms Size: 1.25 KB Save Response

Pretty Raw Preview Visualize

Welcome to the Site!

Office Data

Employee Id	Office Name	Employee Name	Employee Age
1	ecorus	shiva	21

http://127.0.0.1:5000/api/office/add/start-working-for

POST http://127.0.0.1:5000/api/office/add/start-working-for Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "name": "ecorus",
3   "people_working": {
4     "eduardo": 21
5   }
6 }
7
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 314 ms Size: 1.26 KB Save Response

Pretty Raw Preview Visualize

Successfully added office data!!

Office Data

Employee Id	Office Name	Employee Name	Employee Age
2	ecorus	eduardo	21

http://127.0.0.1:5000/api/office/remove/finished_working_for/2

Save

DELETE

http://127.0.0.1:5000/api/office/remove/finished_working_for/2

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

KEY	VALUE	DESCRIPTION		Bulk Edit
Key	Value	Description		

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Time: 323 ms

Size: 1.25 KB

Save Response

Pretty

Raw

Preview

Visualize

Welcome to the Site!

Office Data

Employee Id	Office Name	Employee Name	Employee Age
1	ecorus	shiva	21