Hackathon – Python Microservices

Instructions

- Read the problem statement thoroughly.
- You must follow coding standards.
- Naming convention for the project -TourPackageManagement_<CTDT ID>. eg: TourPackageManagement _CT101010101
- To upload in ION, copy the completed project from your workspace, zip the folder (eg. TourPackageManagement _CT101010101.zip) and upload it in ION on time.
- Make sure your project is free from compilation errors.

Note : **The Test Runner App shown in the demo session yesterday is not up due to some technical issues. Hence, please proceed with testing your application in POSTMAN API and submit the zip file in iON.**

Problem Statement

A tour operator company wants to automate its operations to provide a better user experience to its customers. The company wants a portal to manage their offered packages for a better user experience for its customers. Implement the requirement of the company using Flask web framework in Python. You are expected to create the application as a microservice.

Create the below given resource with its db model in your Flask Microservice Application

Tour package:

PackageId(auto-incremented)

city

days

Price

Please note that all the fields, except the packageId of the above resource will be provided in the input payload while running the APIs. The PackageId is to be automatically generated as an auto-increment field.

The APIs to be implemented in this application are given in the below table. Refer the result verification section to know the exact input and output payload format.

API Endpoint	**HTTP method**	**Input**	**Output**
/addPackage	POST	Method takes a TourPackage payload as input and adds the package successfully	Method returns the TourPackage addition status
/viewAllPackages	GET	None	Method returns all the TourPackages from the system as a list of JSON objects
/viewPackageById/<int:id>	GET	None	Method returns the TourPackage with the given id as a JSON object containing the city, days and price.
/viewPackageByCity/<string:city>	GET	None	Method returns all the TourPackages whose city is same as the given city as a list of JSON objects.
/updatePackage	PUT	Method takes a TourPackage payload as input and updates the package if a TourPackage with the given id exists	Method returns the TourPackage updation status
/deletePackage	DELETE	Method takes a TourPackage payload as input and deletes the package if a TourPackage with the given id exists	Method returns the TourPackage deletion status

Sample Test Cases

Below is the list of URLs which can be hit from POSTMAN API after running the above application. Assuming the port number : 5000 .

Application URLS and payloads – For testing it as a standalone application

1. **Add a Tour Package – POST**

   URL : http://localhost:5000/addPackage

   Input Payload:

```
{
  "city":"Delhi",
  "days":3,
  "price":4000
}
```

Output:

Package with id=1 created

2. **View All Tour Packages – GET**

   URL : http://localhost:5000/viewAllPackages

Output :

```
[
 {
  "city": "Delhi",
  "days": 3,
  "id": 1,
  "price": 4000
 }
]
```

3. **View Tour Package by Id – GET**

URL : http://localhost:5000/viewPackageById/1

Output :


```
{
  "city": "Delhi",
  "days": 3,
  "price": 4000
}
```


4. **View Tour Package by Id – GET**

URL : http://localhost:5000/viewPackageById/4

Output :

No package with the given id found

5. **View Tour Package by City – GET**

URL : http://localhost:5000/viewPackageByCity/Delhi

Output :


```
[
  {
    "city": "Delhi",
    "days": 3,
    "id": 1,
    "price": 4000
  }
]
```


6. **View Tour Package by City – GET**

URL : http://localhost:5000/viewPackageByCity/Mumbai

Output :

No package found for given city

7. **Update Tour Package – PUT**

URL : **http://localhost:5000/updatePackage/1**

Input Payload:


```
{
  "city":"Delhi",
  "days":4,
  "price":10000
}
```


Output:

Package with id=1 updated successfully

8. **Update Tour Package – PUT**

URL: **http://localhost:5000/updatePackage/4**

Input Payload:


```
{
  "city": "Delhi",
  "days":4,
  "price":10000
}
```


Output:

Updation failed due to invalid id

9. **Delete Tour Package – DELETE**

URL : **http://localhost:5000/deletePackage/1**

Output:

Package with id=1 deleted successfully

10. **Delete Tour Package – DELETE**

URL : **http://localhost:5000/deletePackage/3**

Output:

Deletion failed due to invalid id