

Virtusa Assignment

1) Web Series:

i) Add Series

POST - /addSeries Request Body: localhost:8080/addwebseries

```
{  
  "id":16,  
  "name":"Webseries 13",  
  "seasons":1,  
  "episodes":8,  
  "rating":10  
}
```

The screenshot shows the Postman application interface. At the top, the URL is set to `localhost:8080/addwebseries`. The method is selected as `POST`. Below the URL, there are tabs for `Params`, `Auth`, `Headers (8)`, `Body` (which is currently selected), `Pre-req.`, `Tests`, and `Settings`. The `Body` tab is further divided into `raw` and `JSON` sections. The JSON section contains the following data:

```
1  {  
2    "id": 16,  
3    "name": "Webseries 13",  
4    "seasons": 1,  
5    "episodes": 8,  
6    "rating": 10  
7  }
```

Below the body editor, the response status is shown as `201 Created` with a timestamp of `67 ms` and a size of `237 B`. There are buttons for `Save Response`, `Pretty`, `Raw`, `Preview`, `Visualize`, and `JSON`. At the bottom right, there are buttons for `Runner` and `Trash`.

ii)Get All Series**GET - localhost:8080/webseries****Result Response:** Json Data

The screenshot shows the Postman application interface. At the top, the URL is set to `localhost:8080/webseries`. The method is selected as `GET`. Below the URL, there are tabs for `Params`, `Auth`, `Headers (8)`, `Body` (which is currently active), `Pre-req.`, `Tests`, and `Settings`. The `Body` tab is expanded, showing the response in `Pretty` format. The response body is a JSON array containing four objects, each representing a web series with fields `id`, `name`, `seasons`, `episodes`, and `rating`.

```
[{"id": 2, "name": "Webseries 22", "seasons": 1, "episodes": 8, "rating": 10}, {"id": 3, "name": "Webseries 22", "seasons": 1, "episodes": 8, "rating": 10}, {"id": 4, "name": "Webseries 4", "seasons": 1, "episodes": 8, "rating": 10}, {"id": 5, "name": "Webseries 13", "seasons": 1, "episodes": 8, "rating": 10}]
```

iii) Get Series By ID

GET - `localhost:8080/webseries/id/10`

Result Response: Json Data

The screenshot shows the Postman application interface. At the top, the URL `localhost:8080/webseries/id/10` is entered into the address bar. Below the address bar, the method is set to **GET**. To the right of the URL, there are buttons for **Save**, **Edit**, and **Send**. The **Send** button is highlighted in blue.

Below the URL, there are tabs for **Params**, **Auth**, **Headers (8)**, **Body** (with a green dot indicating it's selected), **Pre-req.**, **Tests**, and **Settings**. The **Cookies** tab is also visible.

The **Query Params** section contains a table with one row. The columns are **KEY**, **VALUE**, **DESCRIPTION**, and **...**. The row has a single entry: **Key** and **Value**.

At the bottom, the **Body** section is expanded. It shows the **Pretty** tab selected, displaying the JSON response in a readable format. The response is:

```
1  {
2   "id": 10,
3   "name": "Webseries 2",
4   "seasons": 1,
5   "episodes": 8,
6   "rating": 10
7 }
```

Below the JSON, the status code **200 OK**, time **61 ms**, and size **231 B** are displayed. There is also a **Save Response** button.

iv)Update Series

PUT - localhost:8080/updatewebseries/9

```
Body {  
    "id": 9,  
    "name": "Series Name Updated", "seasons": 10,  
    "episodes": 15,  
    "rating": 10  
}
```

The screenshot shows the Postman application interface. The URL in the header is `localhost:8080/updatewebseries/9`. The method is set to `PUT`. The `Body` tab is selected, showing a JSON payload:

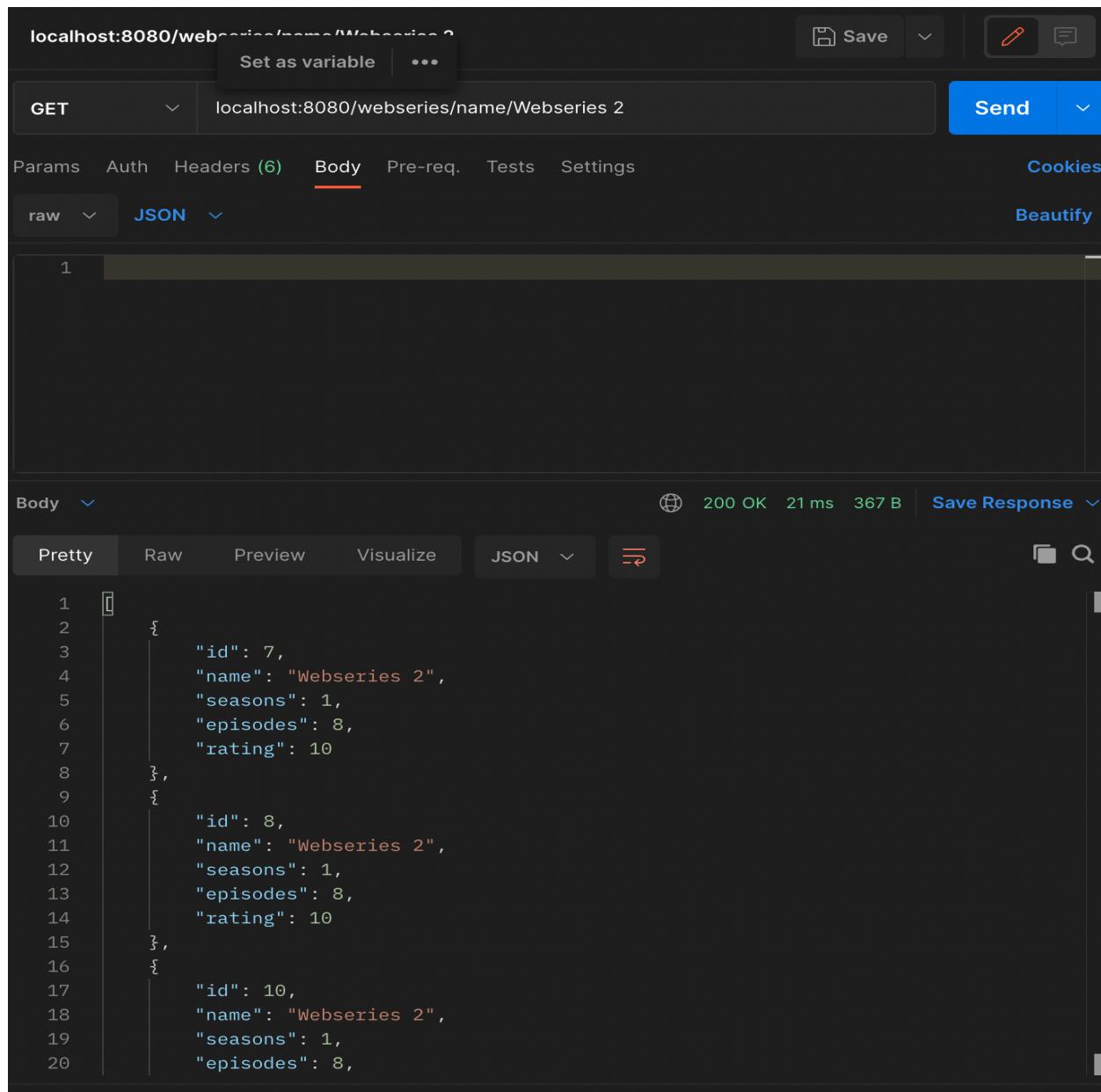
```
1  {  
2      "id": 9,  
3      "name": "Series Name Updated",  
4      "seasons": 10,  
5      "episodes": 15,  
6      "rating": 10  
7  }
```

Below the body, the response is shown as a 200 OK status with 92 ms latency and 240 B size. The response body is identical to the request body.

v)Get Series By Name

GET - `localhost:8080/webseries/name/Webseries 2`

Result Response: Json Data



The screenshot shows the Postman application interface. The URL in the header is `localhost:8080/webseries/name/Webseries 2`. The method is set to **GET**. The **Body** tab is selected, showing the raw JSON response. The response body is a JSON array with three elements, each representing a web series. The JSON is formatted with line numbers and collapsible sections.

```
1 [ { 2   "id": 7, 3   "name": "Webseries 2", 4   "seasons": 1, 5   "episodes": 8, 6   "rating": 10 7 }, 8   { 9     "id": 8, 10    "name": "Webseries 2", 11    "seasons": 1, 12    "episodes": 8, 13    "rating": 10 14 }, 15   { 16     "id": 10, 17     "name": "Webseries 2", 18     "seasons": 1, 19     "episodes": 8, 20   } ]
```

vi) Delete Series

DELETE - `localhost:8080/deletewebseries/14`

Result Response: Series Removed ! {id}

The screenshot shows the Postman application interface. At the top, the URL `localhost:8080/deletewebseries/14` is entered into the address bar, along with a dropdown menu showing "Set as variable" and "Params". To the right are "Save", "Edit", and "Send" buttons. Below the address bar, the method is set to "DELETE" and the URL is again displayed. The "Params" tab is selected, showing a table for "Query Params" with one row: "Key" and "Value". The "Body" tab is selected at the bottom, showing the response status as "200 OK" with "80 ms" and "181 B" response time. The response body is a single line: "1 Series Removed 14". Other tabs like "Raw", "Preview", "Visualize", and "Text" are also visible.

2) Task Management

i) Save a new Task

POST - localhost:8080/savetask

Request Body: JSON

The screenshot shows the Postman application interface. At the top, the URL is set to `localhost:8080/savetask`. Below the URL, the method is selected as `POST`, and the endpoint is also `localhost:8080/savetask`. The `Body` tab is active, and the `JSON` tab is selected under the dropdown. The JSON payload is defined as follows:

```
1 {  
2   "taskId": "12211",  
3   "taskHolderName": "Gowthaman M",  
4   "taskDate": "4/15/2021",  
5   "taskName": "Spring Projects",  
6   "taskStatus": "In Progress"  
7 }
```

At the bottom of the interface, the response status is shown as `201 Created` with a response time of `158 ms` and a size of `297 B`. There is also a `Save Response` button.

ii) Change Task Status

GET - `localhost:8080/changestatus/12211`

The screenshot shows the Postman application interface. At the top, the URL `localhost:8080/changestatus/12211` is entered. Below the URL, the method is set to `PUT`. The `Body` tab is selected, showing the following JSON payload:

```
1 {  
2     "taskId": "12211",  
3     "taskHolderName": "Gowthaman M",  
4     "taskDate": "4/15/2021",  
5     "taskName": "Spring Projects",  
6     "taskStatus": "Completed"  
7 }
```

Below the body, the response status is shown as `200 OK` with a time of `25 ms` and a size of `290 B`.

At the bottom, the response body is displayed in pretty JSON format:

```
1 {  
2     "taskId": "12211",  
3     "taskHolderName": "Gowthaman M",  
4     "taskDate": "4/15/2021",  
5     "taskName": "Spring Projects",  
6     "taskStatus": "Completed"  
7 }
```

iii) Get All Tasks

GET - localhost:8080/tasks/all

The screenshot shows the Postman application interface. At the top, the URL is set to "localhost:8080/tasks/all". Below the URL, the method is selected as "GET". The "Body" tab is active, showing a JSON object with fields: taskId, taskHolderName, taskDate, taskName, and taskStatus. The value for taskStatus is highlighted in yellow. The "Pretty" tab is selected in the preview section at the bottom.

```
1 {
2     "taskId": "12211",
3     "taskHolderName": "Gowthaman M",
4     "taskDate": "4/15/2021",
5     "taskName": "Spring Projects",
6     "taskStatus": "Completed"
7 }
```

Below the preview, the response is displayed in JSON format:

```
1 [
2     {
3         "taskId": "12211",
4         "taskHolderName": "Gowthaman M",
5         "taskDate": "4/15/2021",
6         "taskName": "Spring Projects",
7         "taskStatus": "Completed"
8     },
9     {
10         "taskId": "12212",
11         "taskHolderName": "Gowthaman M",
12         "taskDate": "4/15/2021",
13         "taskName": "Spring Projects",
14         "taskStatus": "Completed"
15     },
16     {
17         "taskId": "12213",
18         "taskHolderName": "Gowthaman M",
19         "taskDate": "4/15/2021",
20         "taskName": "Spring Projects",
21     }
22 ]
```

iv) Get Task by Task HolderName

GET - `localhost:8080/tasks/Gowtham`

The screenshot shows the Postman application interface. At the top, the URL is set to `localhost:8080/tasks/Gowtham`. The method is selected as `GET`. Below the URL, there are tabs for `Params`, `Authorization`, `Headers (8)`, `Body` (which is currently active), `Pre-request Script`, `Tests`, and `Settings`. The `Body` tab has a dropdown menu showing options: `none`, `form-data`, `x-www-form-urlencoded`, `raw` (which is selected), `binary`, `GraphQL`, and `JSON`. To the right of the body dropdown is a `Beautify` button. The main content area displays the raw JSON response:

```
1
2
3 "taskId": "12211",
4 "taskHolderName": "Gowthaman M",
5 "taskDate": "4/15/2021",
6 "taskName": "Spring Projects",
7 "taskStatus": "Completed"
```

Below the response, the status bar indicates `Status: 200 OK Time: 64 ms Size: 288 B` and a `Save Response` button. The bottom section of the interface shows the `Pretty` tab selected, displaying the same JSON response with line numbers and indentation:

```
1 [
2   {
3     "taskId": "12214",
4     "taskHolderName": "Gowtham",
5     "taskDate": "4/15/2021",
6     "taskName": "Spring Projects",
7     "taskStatus": "Completed"
8   }
9 ]
```

v)Delete a Task

GET - localhost:8080/task/delete/12211

The screenshot shows the Postman application interface. At the top, the URL "localhost:8080/task/delete/12211" is entered into the address bar, along with various save and edit icons. Below the address bar, the method is set to "DELETE". The "Body" tab is selected, showing a JSON payload:

```
1 {"taskHolderName": "Gowthaman M", "taskDate": "4/15/2021", "taskName": "Spring Projects", "taskStatus": "Completed"}  
2  
3 "taskId": "12211",  
4 "taskHolderName": "Gowthaman M",  
5 "taskDate": "4/15/2021",  
6 "taskName": "Spring Projects",  
7 "taskStatus": "Completed"
```

Below the body, the response status is shown as "Status: 200 OK Time: 223 ms Size: 184 B". The response body is displayed as "Series Removed 12211".

3)Car Rental Management

i)Add a new Car

POST - localhost:8080/savecar

Request Body:

```
{  
  "carId": "12212",  
  "carModel": "baleno",  
  "carNo": "TN 38 CJ 6636",  
  "status": "available"  
}
```

The screenshot shows the Postman application interface. At the top, the URL 'localhost:8080/savecar' is entered. Below the URL, the method is set to 'POST'. The 'Body' tab is selected, showing the JSON payload: { "carId": "12212", "carModel": "baleno", "carNo": "TN 38 CJ 6636", "status": "available" }. The 'Send' button is highlighted in blue. In the bottom section, the response status is '201 Created' with a response time of '376 ms' and a size of '251 B'. The response body is identical to the request body.

```
localhost:8080/savecar  
POST      POST      localhost:8080/savecar      Send  
Params Auth Headers (8) Body ● Pre-req. Tests Settings Cookies Beautify  
raw JSON  
1  {  
2    "carId": "12212",  
3    "carModel": "baleno",  
4    "carNo": "TN 38 CJ 6636",  
5    "status": "available"  
6  }  
  
Body Cookies Headers (5) Test Results 201 Created 376 ms 251 B Save Response  
Pretty Raw Preview Visualize JSON
```

```
1  {  
2    "carId": "12212",  
3    "carModel": "baleno",  
4    "carNo": "TN 38 CJ 6636",  
5    "status": "available"  
6  }
```

ii) Edit a Car Details

POST - localhost:8080/editcar/12212

Request Body:

```
{  
  "carId": "12212",  
  "carModel": "baleno",  
  "carNo": "TN 38 CJ 6636",  
  "status": "Booked"  
}
```

The screenshot shows the Postman application interface. The URL in the header is `localhost:8080/editcar/12212`. The method is set to `PUT`. The `Body` tab is selected, showing the JSON request body. The response section shows a `200 OK` status with a response time of `118 ms` and a size of `243 B`.

`PUT` `localhost:8080/editcar/12212` `Send`

Params Auth Headers (8) **Body** Pre-req. Tests Settings Cookies

raw **JSON** Beautify

```
1 {  
2   "carId": "12212",  
3   "carModel": "baleno",  
4   "carNo": "TN 38 CJ 6636",  
5   "status": "booked"  
6 }
```

Body Cookies Headers (5) Test Results `200 OK` `118 ms` `243 B` Save Response

Pretty Raw Preview Visualize `JSON` `Raw` `Search`

```
1 {  
2   "carId": "12212",  
3   "carModel": "baleno",  
4   "carNo": "TN 38 CJ 6636",  
5   "status": "booked"  
6 }
```

iii) Get All Cars

GET - localhost:8080/getcars

The screenshot shows the Postman application interface. At the top, the URL is set to "localhost:8080/getcars". The method dropdown shows "GET". Below the URL, there are tabs for "Params", "Auth", "Headers (8)", "Body", "Pre-req.", "Tests", and "Settings". The "Body" tab is currently selected and has a sub-tab "JSON". The JSON content is:

```
1 [  
2   {"carId": "12214",  
3    "carModel": "suv",  
4    "carNo": "TN 38 CJ 6636",  
5    "status": "available"  
6 }]
```

Below the body, the status bar shows "200 OK" and "147 ms". The "Body" tab is also active here, showing the same JSON response. The "Pretty" tab is selected, displaying the JSON with line breaks and indentations.

```
1 [  
2   {  
3     "carId": "12212",  
4     "carModel": "baleno",  
5     "carNo": "TN 38 CJ 6636",  
6     "status": "booked"  
7   },  
8   {  
9     "carId": "12213",  
10    "carModel": "baleno",  
11    "carNo": "TN 38 CJ 6636",  
12    "status": "available"  
13   },  
14   {  
15     "carId": "12214",  
16     "carModel": "suv",  
17     "carNo": "TN 38 CJ 6636",  
18     "status": "available"  
19   }]
```

iv)Get Car By ID

GET - localhost:8080/getcar/12214

The screenshot shows the Postman application interface. At the top, the URL `localhost:8080/getcar/12214` is entered. Below the URL, the method is set to `GET`. The `Body` tab is selected, showing the JSON input:

```
1 {"carId": "12214",  
2 "carModel": "suv",  
3 "carNo": "TN 38 CJ 6636",  
4 "status": "available"}  
5  
6
```

At the bottom of the body section, there are tabs for `Pretty`, `Raw`, `Preview`, and `Visualize`. The `JSON` tab is currently active. The response section shows the status `200 OK`, time `56 ms`, and size `243 B`. The response body is identical to the input:1 {"carId": "12214",
2 "carModel": "suv",
3 "carNo": "TN 38 CJ 6636",
4 "status": "available"}
5
6

v)Delete a Car

GET - localhost:8080/deletecar/12213

The screenshot shows the Postman application interface. At the top, the URL `localhost:8080/deletecar/12213` is entered into the address bar, along with various save and message icons. Below the address bar, the method is set to `DELETE` and the target URL is displayed again. To the right is a large blue `Send` button. Underneath the URL, there are tabs for `Params`, `Auth`, `Headers (8)`, `Body` (which is currently selected and highlighted in red), `Pre-req.`, `Tests`, and `Settings`. The `Body` tab has a dropdown menu showing `raw` and `JSON`, with `JSON` being selected. To the right of the body section is a `Beautify` link. The main content area displays a JSON object with the following structure:

```
1 [{}  
2 "carId": "12214",  
3 "carModel": "suv",  
4 "carNo": "TN 38 CJ 6636",  
5 "status": "available"  
6 ]
```

Below the body content, there are tabs for `Body`, `Cookies`, `Headers (5)`, and `Test Results`. The `Test Results` tab is currently selected. On the right side of the results panel, there is a status indicator showing `200 OK`, `69 ms`, and `184 B`, followed by a `Save Response` button. At the bottom of the results panel, there are buttons for `Pretty`, `Raw`, `Preview`, `Visualize`, and `Text` (with a dropdown arrow), along with a search icon and a refresh icon.

1 Series Removed 12213

6)Travel List

i)Add Travel

POST - localhost:8080/addTravel

Request Body:

```
{  
  "id": "tp009",  
  "name": "kashmir",  
  "description": "Your Own Wish"  
}
```

Response:

The screenshot shows the Postman application interface. At the top, the URL 'localhost:8080/addTravel' is entered in the address bar, along with various save and edit icons. Below the address bar, the method 'POST' is selected, and the target URL is again shown. To the right is a large blue 'Send' button. Underneath the URL, there are tabs for 'Params', 'Auth', 'Headers (8)', 'Body' (which is currently active and highlighted in green), 'Pre-req.', 'Tests', and 'Settings'. The 'Body' tab has a dropdown menu showing 'raw' and 'JSON', with 'JSON' being selected. In the JSON editor area, the following object is displayed:

```
1 {  
2   "id": "tp009",  
3   "name": "kashmir",  
4   "description": "Your Own Wish"  
5 }
```

Below the JSON editor, there are tabs for 'Body', 'Cookies', 'Headers (5)', and 'Test Results'. The 'Test Results' tab is active and highlighted in green. It displays the response message '1 Travel Added Successfully'. At the top right of the main window, there are status indicators for '201 Created', '20 ms', '194 B', and a 'Save Response' button.

ii) Get All Travel List

GET - localhost:8080/travel

Result Response: Json Data

The screenshot shows the Postman application interface. At the top, the URL is set to "localhost:8080/travel". Below the URL, the method is selected as "GET". The "Body" tab is active, showing a JSON object with five lines of code. The "Headers" tab shows eight entries. The "Tests" and "Settings" tabs are also visible. In the bottom section, the "Body" tab is active, displaying the JSON response in a pretty-printed format. The response consists of an array of three travel items, each with an id, name, and description. The "Pretty" button is highlighted.

```
1 {
2   "id": "tp009",
3   "name": "kashmir",
4   "description": "Your Own Wish"
5 }
```

```
1 []
2 [
3   {
4     "id": "tp005",
5     "name": "Jammu & Kashmir",
6     "description": "Your Own Wish"
7   },
8   {
9     "id": "tp006",
10    "name": "Jammu",
11    "description": "Your Own Wish"
12  },
13  {
14    "id": "tp009",
15    "name": "kashmir",
16    "description": "Your Own Wish"
17 }
```

iii)Get Single Travel By id

GET - localhost:8080/travel/tp005

Result Response: Json Data

The screenshot shows the Postman application interface. At the top, there is a header bar with a save icon, a pencil icon, and a copy icon. Below the header, the URL 'localhost:8080/travel/tp005' is entered into the 'Send' field. To the left of the URL, there is a dropdown menu with 'Set as variable' and a three-dot menu. The main area shows a 'GET' method selected, and the URL 'localhost:8080/travel/tp005' again. To the right of the URL is a 'Send' button with a dropdown arrow. Below the URL, there are tabs for 'Params', 'Auth', 'Headers (8)', 'Body', 'Pre-req.', 'Tests', and 'Settings'. The 'Body' tab is currently active, indicated by a red underline. Under the 'Body' tab, there are two dropdown menus: 'raw' and 'JSON', with 'JSON' being the selected option. To the right of the body section is a 'Beautify' button. The JSON response is displayed in a code editor-like format:

```
1 {  
2   "id": "tp009",  
3   "name": "kashmir",  
4   "description": "Your Own Wish"  
5 }
```

At the bottom of the interface, there are tabs for 'Body', 'Cookies', 'Headers (5)', and 'Test Results'. The 'Headers (5)' tab is active, indicated by a red underline. To the right of these tabs, there is a status bar showing '200 OK', '60 ms', '233 B', and a 'Save Response' button with a dropdown arrow. Below the status bar, there are tabs for 'Pretty', 'Raw', 'Preview', 'Visualize', and 'JSON', with 'JSON' being the selected option. To the right of these tabs is a search icon. The JSON response is also displayed here in a pretty-printed format:

```
1 {  
2   "id": "tp005",  
3   "name": "Jammu & Kashmir",  
4   "description": "Your Own Wish"  
5 }
```

iv) Update Travel List

PUT -localhost:8080/travel/tp005

Request Body:

```
{  
  "id": "tp005",  
  "name": "Jammu and Kashmir",  
  "description": "My Own Wish Updated"  
}
```

Result Response:

The screenshot shows the Postman application interface. At the top, the URL `localhost:8080/travel/tp005` is entered in the address bar, along with various save and edit icons. Below the address bar, the method is set to `PUT` and the target URL is `localhost:8080/travel/tp005`. The `Body` tab is selected, showing the JSON payload used for the update. The payload is a JSON object with fields `id`, `name`, and `description`. The `Headers` section shows 8 headers. The `Body` section is currently in `JSON` format, with a `Beautify` button available. The response section at the bottom shows a status of `200 OK` with a response time of `86 ms` and a size of `197 B`. The response body contains the message `Travel Updated Successfully tp005`.

v)Delete Travel

DELETE -localhost:8080/travel/tp005

Result Response:

The screenshot shows a Postman request configuration and its resulting response.

Request Configuration:

- Method: **DELETE**
- URL: **localhost:8080/travel/tp005**
- Body (JSON):

```
1 {
2   "id": "tp005",
3   "name": "Jammu and Kashmir",
4   "description": "My Own Wish Updated"
5 }
```

Response:

- Status: 200 OK
- Time: 74 ms
- Size: 197 B
- Body:

```
1 Travel Deleted Successfully tp005
```

5)Employee Management

i)Add Employee

POST - localhost:8080/addEmployee

Request Body:

```
{  
    "employeeId": "12211",  
    "employeeName": "Gowthaman M",  
    "employeeEmail": "gowthaman@iamneo.ai",  
    "dept": "content-team"  
}
```

Response

The screenshot shows the Postman application interface. At the top, the URL is set to `localhost:8080/addEmployee`. Below the URL, the method is selected as `POST`. The `Body` tab is active, showing the JSON payload for the request. The response section at the bottom shows a `201 Created` status with a response time of `345 ms` and a size of `280 B`.

Request Body (JSON):

```
1 [ {  
2     "employeeId": "12211",  
3     "employeeName": "Gowthaman M",  
4     "employeeEmail": "gowthaman@iamneo.ai",  
5     "dept": "content-team"  
6 } ]  
7
```

Response Headers (5):

```
1 [ {  
2     "Content-Type": "application/json",  
3     "Date": "Tue, 03 Oct 2023 10:45:15 GMT",  
4     "Server": "Apache/2.4.41 (Ubuntu)",  
5     "Transfer-Encoding": "chunked",  
6     "X-Powered-By": "PHP/8.1.12"  
7 } ]
```

Response Body (Pretty JSON):

```
1 [ {  
2     "employeeId": "12211",  
3     "employeeName": "Gowthaman M",  
4     "employeeEmail": "gowthaman@iamneo.ai",  
5     "dept": "content-team"  
6 } ]
```

ii) Get All Employee

GET - localhost:8080/getAllEmployee

Response

The screenshot shows the Postman application interface. At the top, the URL is set to "localhost:8080/getAllEmployee". Below the URL, the method is selected as "GET". The "Body" tab is active, showing a JSON object with fields: employeeId, employeeName, employeeEmail, and dept. The "Pretty" tab in the bottom navigation bar is selected, displaying the JSON response in a readable, indented format.

Body (Pretty)

```
1 [ ]  
2 { "employeeId": "12211",  
3   "employeeName": "Gowthaman M",  
4   "employeeEmail": "gowthaman@iamneo.ai",  
5   "dept": "content-team"  
6 }  
7 [ ]  
8 {  
9   "employeeId": "12212",  
10  "employeeName": "Gowthaman M",  
11  "employeeEmail": "gowthaman@iamneo.ai",  
12  "dept": "marketing-team"  
13 }  
14 {  
15   "employeeId": "12213",  
16   "employeeName": "Gowthaman M",  
17   "employeeEmail": "gowthaman@iamneo.ai",  
18   "dept": "sales-team"  
19 }  
20 ]
```

iii) Get Employee By ID

GET - localhost:8080/getEmployee/12211

Response

The screenshot shows the Postman application interface. At the top, the URL is set to `localhost:8080/getEmployee/12211`. Below the URL, the method is selected as `GET`, and the endpoint is also `localhost:8080/getEmployee/12211`. The `Body` tab is active, showing a JSON payload:

```
1 [{}]
2 "employeeId": "12213",
3 "employeeName": "Gowthaman M",
4 "employeeEmail": "gowthaman@iamneo.ai",
5 "dept": "sales-team"
6 []
7
```

Below the body, the response details are shown: `200 OK`, `78 ms`, and `275 B`. The response body is identical to the request body, indicating a self-referencing loop or a placeholder.

```
1 [{}]
2 "employeeId": "12211",
3 "employeeName": "Gowthaman M",
4 "employeeEmail": "gowthaman@iamneo.ai",
5 "dept": "content-team"
6 []
```

iv)Update Employee:

POST- localhost:8080/updateEmployee/12211

RESPONSE

The screenshot shows the Postman application interface. At the top, the URL `localhost:8080/updateEmployee/12211` is entered. Below the URL, the method is set to `PUT` and the endpoint is again specified as `localhost:8080/updateEmployee/12211`. The `Body` tab is selected, showing a JSON payload with the following structure:

```
1 {  
2   "employeeId": "12211",  
3   "employeeName": "Gowthaman M",  
4   "employeeEmail": "gowthaman@iamneo.ai",  
5   "dept": "sales-team"  
6 }  
7
```

Below the body, the response details are shown: a status of `200 OK`, a time of `75 ms`, and a size of `273 B`. The response body is identical to the request body, indicating a successful update.

v)Delete Employee

GET - localhost:8080/deleteEmployee/12211

Response

The screenshot shows the Postman application interface. At the top, the URL `localhost:8080/deleteEmployee/12211` is entered. Below the URL, the method is set to `DELETE`. The `Body` tab is selected, showing a JSON payload:

```
1 {  
2   "employeeId": "12211",  
3   "employeeName": "Gowthaman M",  
4   "employeeEmail": "gowthaman@iamneo.ai",  
5   "dept": "sales-team"  
6 }  
7
```

Below the body, the response section shows a status of `200 OK`, `92 ms`, and `186 B`. The response text is: `Employee Removed 12211`.

4)E-Commerce

i)Add Product

POST - localhost:8080/saveProduct

Request Body:

```
{  
    "productId": "AS002",  
    "productName": "ASUS Zenfone",  
    "description": "!",  
    "quantity": "1",  
    "price": "18000",  
    "type": "mobile"  
}
```

Response

The screenshot shows the Postman application interface. At the top, the URL is set to `localhost:8080/saveProduct`. The method is selected as `POST`. Below the URL, there are tabs for `Params`, `Auth`, `Headers (8)`, `Body` (which is currently active), `Pre-req.`, `Tests`, and `Settings`. Under the `Body` tab, the content type is set to `JSON`. The request body is displayed as follows:

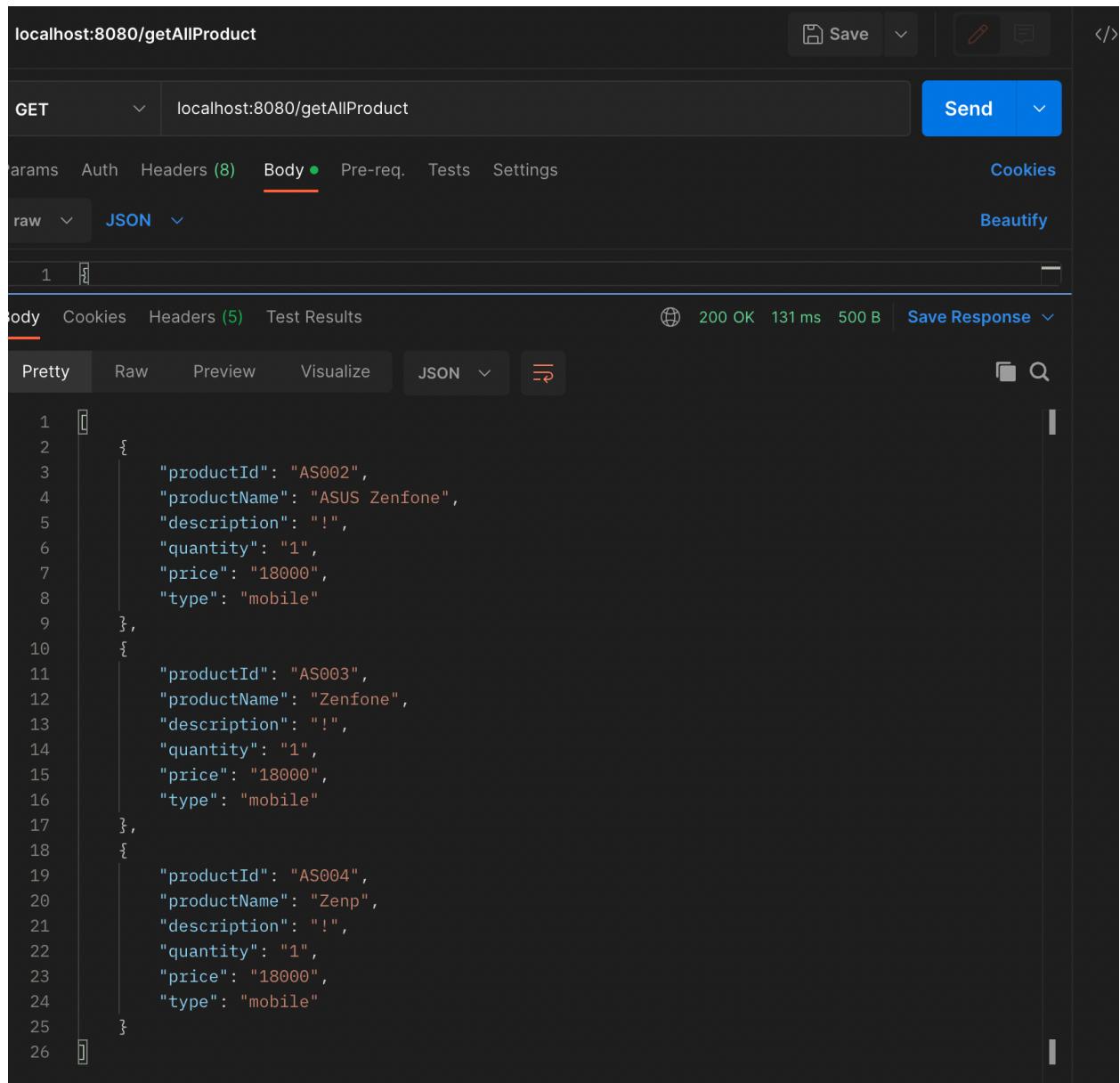
```
1 {"  
2     "productId": "AS002",  
3     "productName": "ASUS Zenfone",  
4     "description": "!",  
5     "quantity": "1",  
6     "price": "18000",  
7     "type": "mobile"  
8 }
```

At the bottom of the interface, the response section shows the status as `201 Created` with a response time of `359 ms` and a size of `284 B`. The response body is identical to the request body, indicating a successful creation of the product.

ii) Get All Products

GET - localhost:8080/getAllProduct

Response: List<productModel>



The screenshot shows a Postman interface with the following details:

- Request URL:** localhost:8080/getAllProduct
- Method:** GET
- Body:** JSON (selected)
- Response Headers:** 200 OK, 131 ms, 500 B
- Response Body (Pretty JSON):**

```
1 [ ]  
2 {  
3     "productId": "AS002",  
4     "productName": "ASUS Zenfone",  
5     "description": "!",  
6     "quantity": "1",  
7     "price": "18000",  
8     "type": "mobile"  
9 },  
10 {  
11     "productId": "AS003",  
12     "productName": "Zenfone",  
13     "description": "!",  
14     "quantity": "1",  
15     "price": "18000",  
16     "type": "mobile"  
17 },  
18 {  
19     "productId": "AS004",  
20     "productName": "Zenp",  
21     "description": "!",  
22     "quantity": "1",  
23     "price": "18000",  
24     "type": "mobile"  
25 }
```

iii)Get product By ProductID

POST - /localhost:8080/getProduct

Request Body:

```
{  
"productId" : "AS004"  
}
```

Response

The screenshot shows the Postman application interface. At the top, the URL is set to `localhost:8080/getProduct`. The method is selected as `GET`. In the `Body` tab, the `JSON` tab is active, displaying the following request body:

```
1 {  
2   "productId" : "AS004"  
3 }
```

Below the request, the response section is visible. The status bar indicates a `200 OK` response with `73 ms` and `271 B` size. The response body is displayed in `Pretty` format:1 {
2 "productId": "AS004",
3 "productName": "Zenp",
4 "description": "!",
5 "quantity": "1",
6 "price": "18000",
7 "type": "mobile"
8 }

iv) Get Products By Type

POST -localhost:8080/getByType

Request Body:

```
{  
  "type": "mobile"  
}
```

Response:

The screenshot shows the Postman interface with the following details:

- URL:** localhost:8080/getByType
- Method:** GET
- Body:** JSON (containing the request body shown above)
- Response Status:** 200 OK (84 ms, 500 B)
- Response Data (Pretty JSON):**

```
1  [  
2   {  
3     "productId": "AS002",  
4     "productName": "ASUS Zenfone",  
5     "description": "!",  
6     "quantity": "1",  
7     "price": "18000",  
8     "type": "mobile"  
9   },  
10  {  
11    "productId": "AS003",  
12    "productName": "Zenfone",  
13    "description": "!",  
14    "quantity": "1",  
15    "price": "18000",  
16    "type": "mobile"  
17  },  
18  {  
19    "productId": "AS004",  
20    "productName": "Zenp",  
21    "description": "!",  
22    "quantity": "1",  
23    "price": "18000",  
24    "type": "mobile"  
25  }]
```

v)Update Product By ProductId

PUT-localhost:8080/updateproduct/AS004

RESPONSE:

The screenshot shows the Postman application interface. At the top, the URL `localhost:8080/updateproduct/AS004` is entered. Below it, a **PUT** method is selected, and the URL is repeated in the request field. To the right, there are buttons for **Save**, **Send**, and other actions.

The **Body** tab is active, showing a JSON payload:

```
1  {
2    "productId": "AS004",
3    "productName": "Zenp",
4    "description": "!",
5    "quantity": "1",
6    "price": "24000",
7    "type": "mobile"
8  }
```

Below the body, the **Headers** tab shows five entries. The **Test Results** tab indicates a successful response:

- Body: **Pretty** (selected), **Raw**, **Preview**, **Visualize**
- Headers: **(5)**
- Test Results: **200 OK**, **90 ms**, **271 B**
- Buttons: **Save Response**, **Copy**, **Find**

vi) Delete Employee

POST - localhost:8080/deleteProduct

Request Body:

```
{  
"productId": "AS004"  
}
```

The screenshot shows the Postman application interface. At the top, the URL `localhost:8080/deleteProduct` is entered. Below it, a `DELETE` button is selected, and the URL `localhost:8080/deleteProduct` is displayed again. The `Body` tab is active, showing the JSON content:

```
1 {"  
2   "productId": "AS004"  
3 }
```

. The `JSON` dropdown is also visible. In the bottom section, the response status is `200 OK`, and the response body is `1 Product Removed AS004`.