

Sporty Shoes

The code for the project is hosted on: [GitHub](#)

Project Developer: **Vijaysai Yekbote**

This document contains the following sections:

- [Sprint Planning](#)
- [Core Concepts used in project](#)
- [Product capabilities and source code](#)
- [Unique Selling point of the Application](#)
- [Conclusion](#)

Sprint Planning and Task Completion

Note: A separate document for sprint planning is attached to the Zip.

The project is planned to be completed in two Sprints. The task assumed to be completed are:

Sprint 1:

- Flow chart of the application
- Initializing Git repository to track changes of development
- Writing the source code

Sprint 2:

- Testing the program with various user inputs
- Pushing project to GitHub
- Creating documentation

Core Concepts Used in the Project

- Java
- Spring Boot
- Mysql DB
- Postman

Product Capabilities

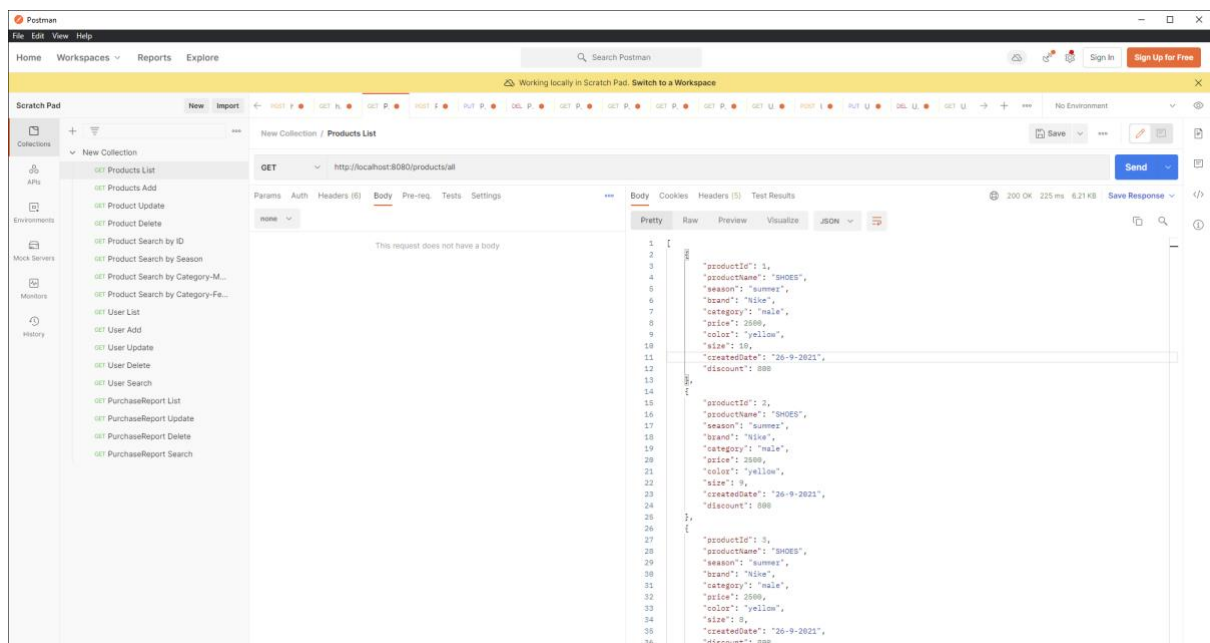
The application has the following capabilities:

1. Products
 - 1.1.Products List
 - 1.2.Product Add
 - 1.3.Product Update
 - 1.4.Product Search by id
2. Users
 - 2.1. User List
 - 2.2.User Update
 - 2.3.User Delete
 - 2.4.User Search
3. Purchase Report
 - 3.1.Purchase Report List
 - 3.2.Purchase repost Update
 - 3.3.Purchase Report Delete
 - 3.4.Purchase Report Search

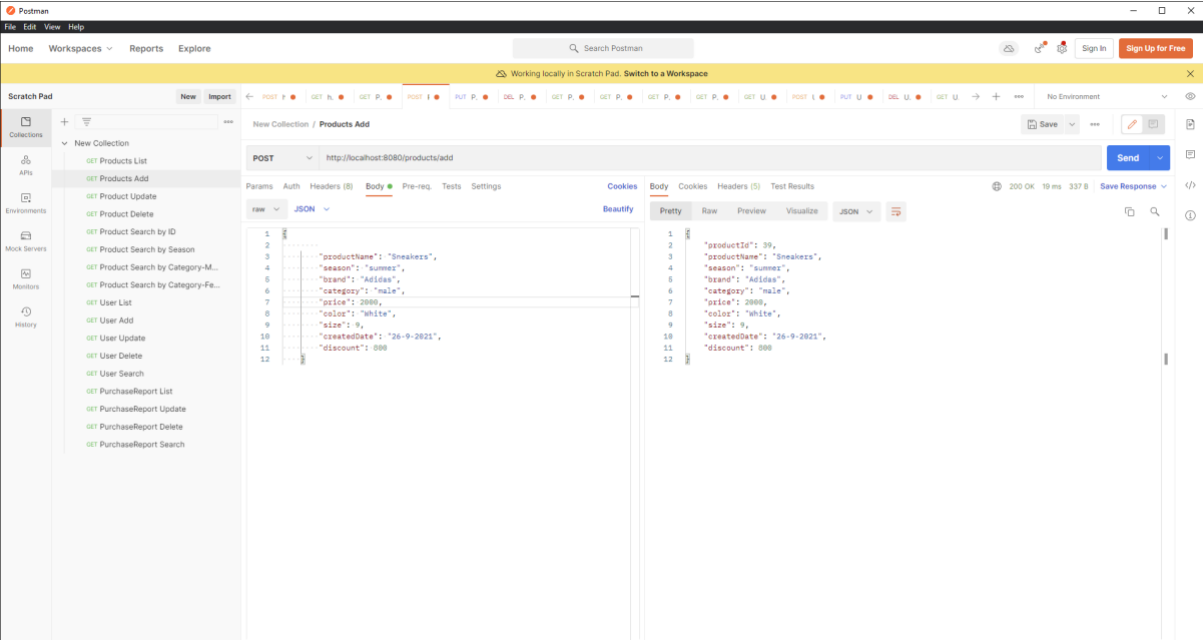
Source code for each of the capabilities with output is shown below:

Note: Complete source code, respective servlets of the app is zipped please check zip/Gitrepo for reference.

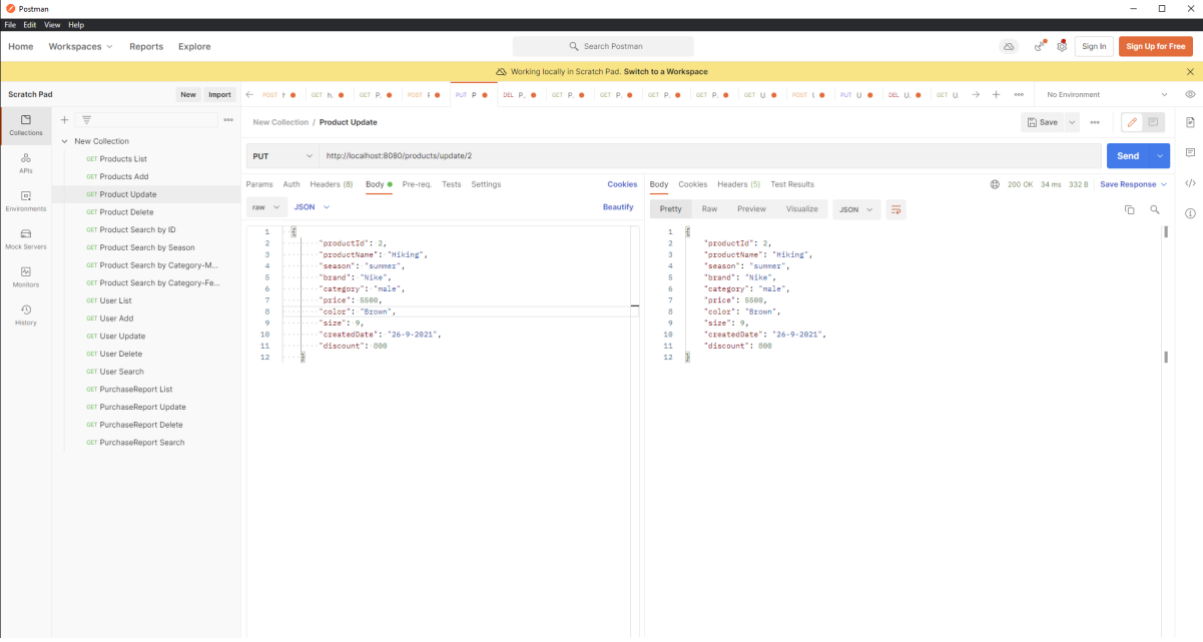
Products List



Product Add



Product Update



Product Delete

Postman interface showing a DELETE request to `http://localhost:8080/products/delete/4`. The response body is displayed in the "Body" tab, showing a list of steps:

```
1 Returns Nothing
2 Check Database For given Id(4)
3 Will have been deleted
```

Product Search by ID

Postman interface showing a GET request to `http://localhost:8080/products/search/5`. The response body is displayed in the "Body" tab, showing a JSON object:

```
{
  "productID": 5,
  "productName": "SHOES",
  "season": "summer",
  "brand": "Nike",
  "category": "Female",
  "color": "black",
  "size": 7,
  "createdDate": "26-9-2021",
  "discount": 800
}
```

Product Search by Season

The screenshot shows the Postman application with a new collection named "Product Search by Category-Male". A GET request is configured to the URL `http://localhost:8080/products/searchbycategory/male`. The response is displayed in the "Body" tab, showing a JSON array of three product objects. The status is 200 OK, and the response time is 20 ms.

KEY	VALUE	DESCRIPTION
Key	Value	Description

```
1
2
3 {
4   "productId": 1,
5   "productName": "SHOES",
6   "season": "summer",
7   "brand": "Nike",
8   "category": "male",
9   "price": 2000,
10  "color": "yellow",
11  "size": 10,
12  "createdAt": "26-9-2021",
13  "discount": 0.00
14 },
15 {
16   "productId": 2,
17   "productName": "Hiking",
18   "season": "summer",
19   "brand": "Nike",
20   "category": "male",
21   "price": 5000,
22   "color": "Brown",
23   "size": 9,
24   "createdAt": "26-9-2021",
25   "discount": 0.00
26 },
27 {
28   "productId": 3,
29   "productName": "SHOES",
30   "season": "summer",
31   "brand": "Nike",
32   "category": "male",
33   "price": 2000,
34   "color": "yellow",
35   "size": 9,
36   "createdAt": "26-9-2021",
37   "discount": 0.00
38 }
```

Passenger Search by Category Male

The screenshot shows the Postman application with a new collection named "Product Search by Category-Male". A GET request is configured to the URL `http://localhost:8080/products/searchbycategory/male`. The response is displayed in the "Body" tab, showing a JSON array of three product objects. The status is 200 OK, and the response time is 20 ms.

KEY	VALUE	DESCRIPTION
Key	Value	Description

```
1
2
3 {
4   "productId": 1,
5   "productName": "SHOES",
6   "season": "summer",
7   "brand": "Nike",
8   "category": "male",
9   "price": 2000,
10  "color": "yellow",
11  "size": 10,
12  "createdAt": "26-9-2021",
13  "discount": 0.00
14 },
15 {
16   "productId": 2,
17   "productName": "Hiking",
18   "season": "summer",
19   "brand": "Nike",
20   "category": "male",
21   "price": 5000,
22   "color": "Brown",
23   "size": 9,
24   "createdAt": "26-9-2021",
25   "discount": 0.00
26 },
27 {
28   "productId": 3,
29   "productName": "SHOES",
30   "season": "summer",
31   "brand": "Nike",
32   "category": "male",
33   "price": 2000,
34   "color": "yellow",
35   "size": 9,
36   "createdAt": "26-9-2021",
37   "discount": 0.00
38 }
```

Passenger Search by Category Female

The screenshot shows the Postman application interface. The 'Scratch Pad' tab is active, displaying a new collection named 'Product Search by Category-Female'. The request is a GET method to the URL 'http://localhost:8080/products/searchbycategory/female'. The response is a JSON array of three product objects, all categorized as 'female'.

Request Details:

- Method: GET
- URL: http://localhost:8080/products/searchbycategory/female

Response (JSON):

```
[{"productId": 6, "productName": "SHOES", "season": "summer", "brand": "nike", "category": "female", "price": 2500, "color": "yellow", "size": 9, "createdAt": "26-9-2021", "discount": 80}, {"productId": 4, "productName": "SHOES", "season": "summer", "brand": "nike", "category": "female", "price": 2500, "color": "yellow", "size": 9, "createdAt": "26-9-2021", "discount": 80}, {"productId": 7, "productName": "SHOES", "season": "summer", "brand": "nike", "category": "female", "price": 2500, "color": "yellow", "size": 9, "createdAt": "26-9-2021", "discount": 80}]
```

User List

The screenshot shows the Postman application interface. The 'Scratch Pad' tab is active, displaying a new collection named 'User List'. The request is a GET method to the URL 'http://localhost:8080/users/all'. The response is a JSON array of five user objects.

Request Details:

- Method: GET
- URL: http://localhost:8080/users/all

Response (JSON):

```
[{"id": 1, "emailId": "kirti@kirti.com", "username": "kirtichow", "password": "123456789", "mobile": 1234565}, {"id": 2, "emailId": "vijayraj@vijay.com", "username": "vijayraj", "password": "123456789", "mobile": 1234565}, {"id": 3, "emailId": "vishu@vishu.com", "username": "vishu", "password": "123456789", "mobile": 123}, {"id": 4, "emailId": "kapil@kapil.com", "username": "kapilchow", "password": "123789", "mobile": 123456153}, {"id": 5, "emailId": "srujan@srujan.com", "username": "srujan", "password": "1237", "mobile": 1234564253}]
```

User Add

The screenshot shows the Postman application with a new collection named "User Add". A POST request is configured to the endpoint `http://localhost:8080/users/add`. The request body is a JSON object containing user information. The response is a 200 OK status with a JSON body.

Request:

```
POST http://localhost:8080/users/add
{
  "emailid": "ankit@ankit.com",
  "username": "ankitp",
  "password": "1237",
  "mobile": 123464283
}
```

Response:

```
200 OK 17 ms 257 B
{
  "id": 7,
  "emailid": "ankit@ankit.com",
  "username": "ankitp",
  "password": "1237",
  "mobile": 123464283
}
```

User Update

The screenshot shows the Postman application with a new collection named "User Update". A PUT request is configured to the endpoint `http://localhost:8080/users/update/4`. The request body is a JSON object containing updated user information. The response is a 200 OK status with a JSON body. The interface also shows a "Previously" section indicating the previous password was 123788.

Request:

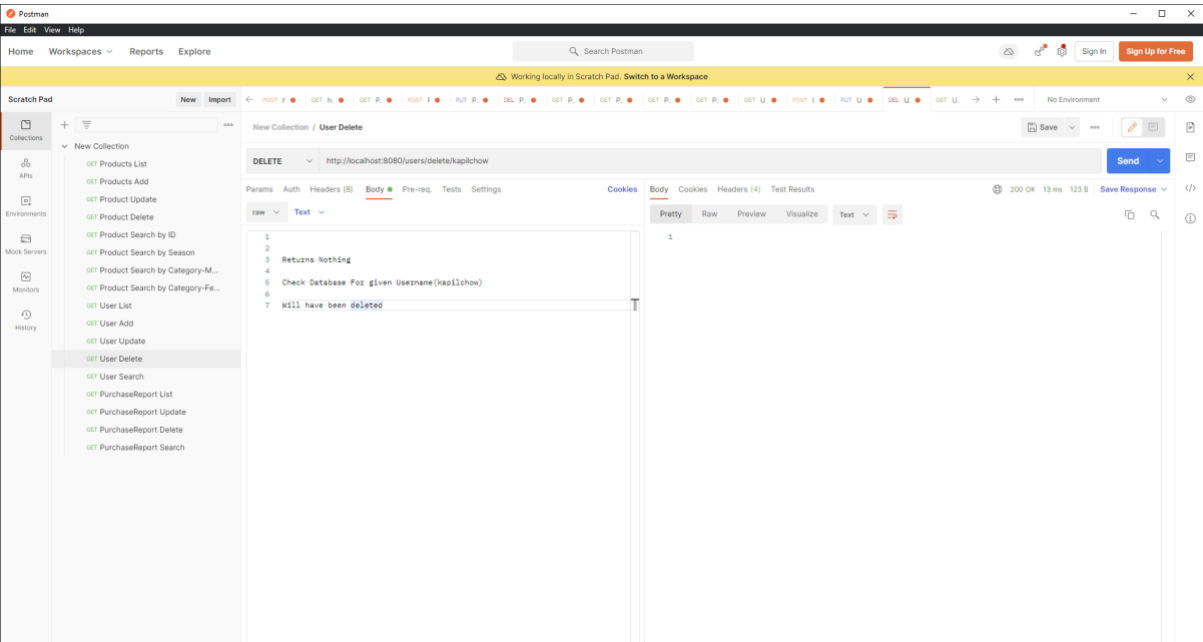
```
PUT http://localhost:8080/users/update/4
{
  "id": 4,
  "emailid": "kavil@kavil.com",
  "username": "kavilchou",
  "password": "123799",
  "mobile": 123464153
}
```

Response:

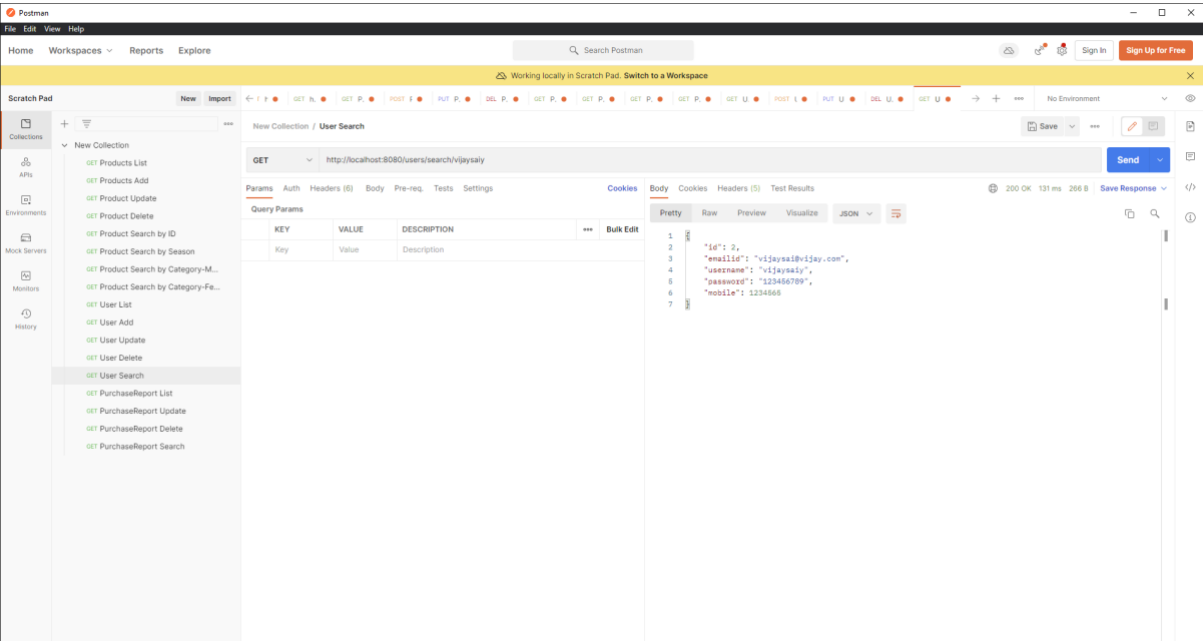
```
200 OK 33 ms 262 B
{
  "id": 4,
  "emailid": "kavil@kavil.com",
  "username": "kavilchou",
  "password": "123799",
  "mobile": 123464153
}
```

Previously: The password was 123788
Check user: 123

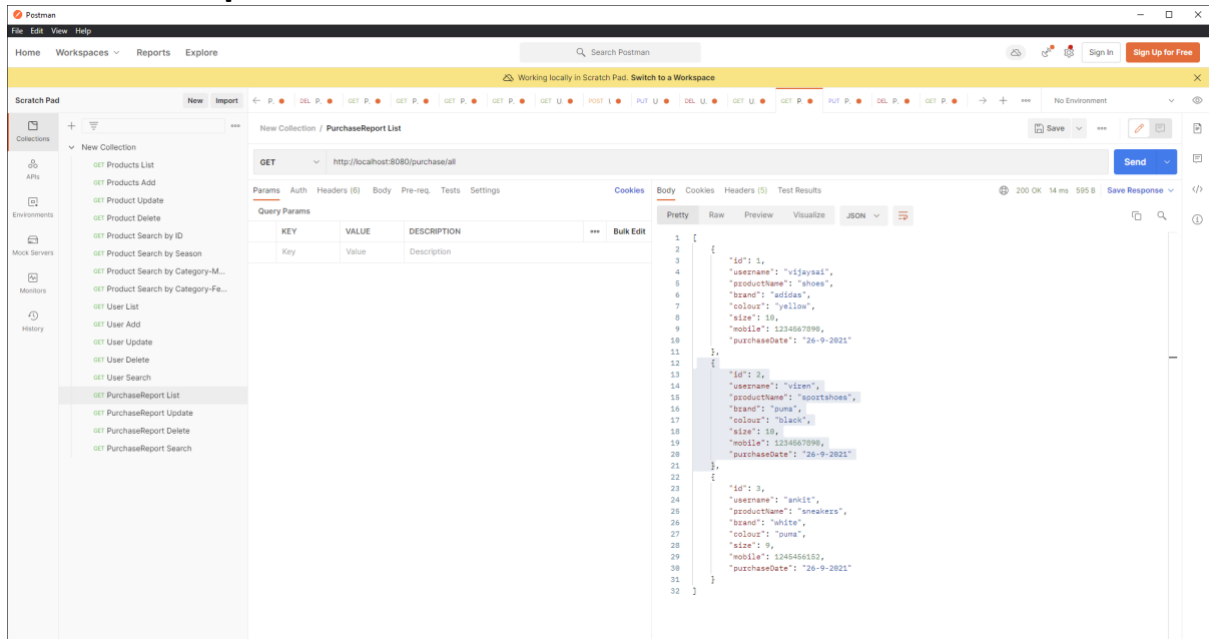
User Delete



User Search



Purchase Report List

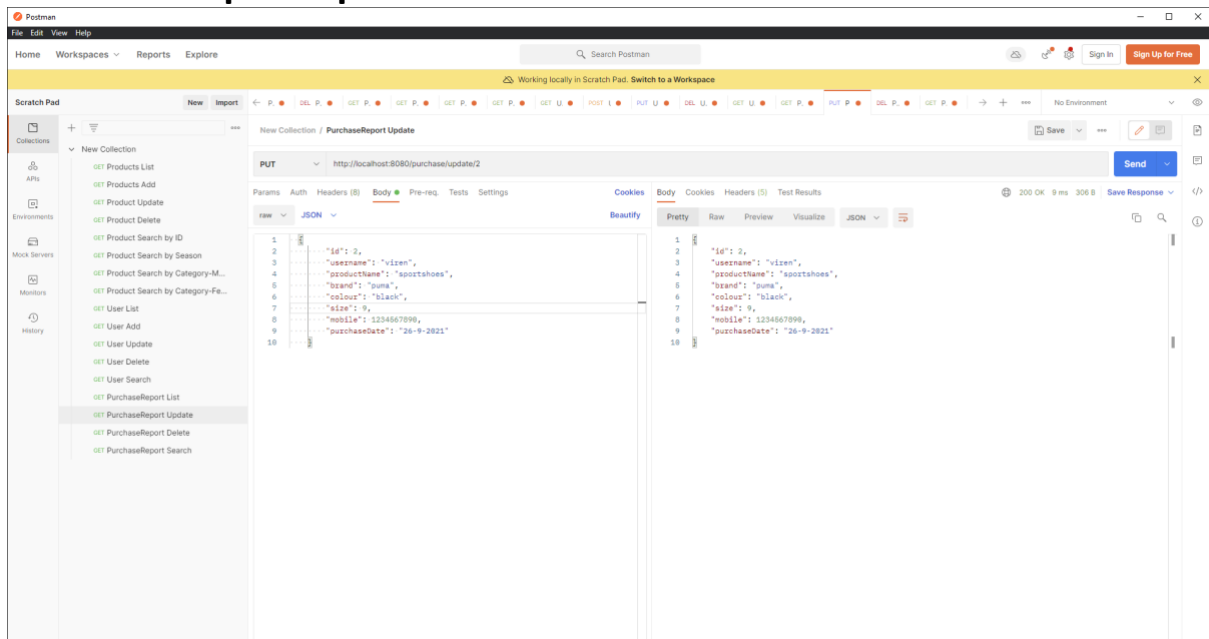


Postman interface showing a GET request to `http://localhost:8080/purchase/all`. The response is a JSON array of 3 items, each representing a purchase report.

KEY	VALUE	DESCRIPTION
Key	Value	Description

```
1 [
2   {
3     "id": 1,
4     "username": "vijayal",
5     "productName": "shoes",
6     "brand": "adidas",
7     "colour": "yellow",
8     "size": 10,
9     "mobile": 1234567890,
10    "purchaseDate": "26-9-2021"
11  },
12  {
13    "id": 2,
14    "username": "vijen",
15    "productName": "sportshoes",
16    "brand": "puma",
17    "colour": "black",
18    "size": 9,
19    "mobile": 1234567890,
20    "purchaseDate": "26-9-2021"
21  },
22  {
23    "id": 3,
24    "username": "anhit",
25    "productName": "sneakers",
26    "brand": "white",
27    "colour": "puma",
28    "size": 9,
29    "mobile": 124565152,
30    "purchaseDate": "26-9-2021"
31  }
32 ]
```

Purchase Report Update

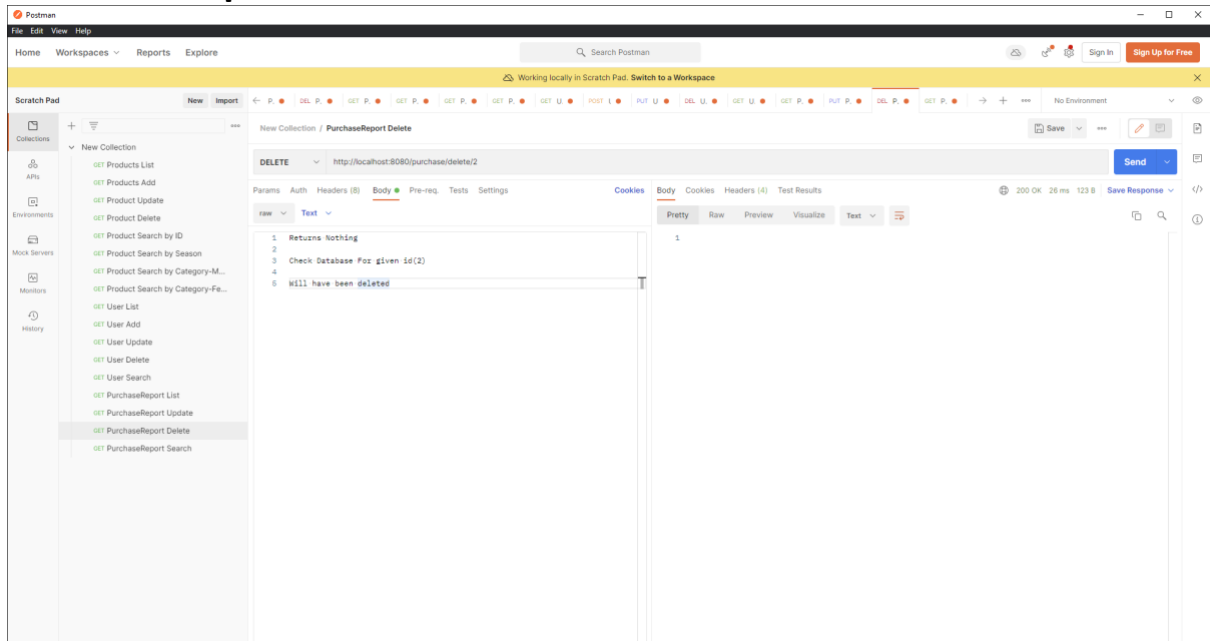


Postman interface showing a PUT request to `http://localhost:8080/purchase/update/2`. The request body is a JSON object for the second item in the list, and the response is the same JSON object.

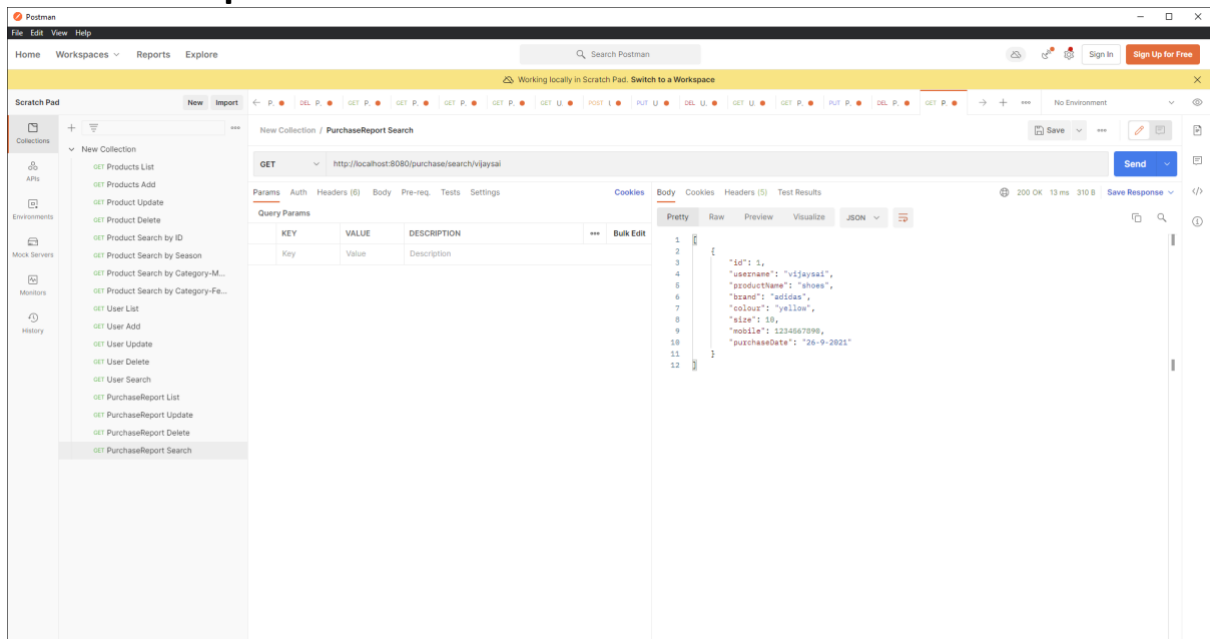
```
1 {
2   "id": 2,
3   "username": "vijen",
4   "productName": "sportshoes",
5   "brand": "puma",
6   "colour": "black",
7   "size": 9,
8   "mobile": 1234567890,
9   "purchaseDate": "26-9-2021"
10 }
```

```
1 {
2   "id": 2,
3   "username": "vijen",
4   "productName": "sportshoes",
5   "brand": "puma",
6   "colour": "black",
7   "size": 9,
8   "mobile": 1234567890,
9   "purchaseDate": "26-9-2021"
10 }
```

Purchase Report Delete



Purchase Report Search



Unique Selling Points of the Application:

- 1.** The application is designed to keep on running and taking user inputs even after exceptions occur.
- 2.** Can search for product by different parameters such as Brand, Season, Gender

Conclusion:

Further improvements to the application can be done which may include:

- Develop UI
- Develop Front End
- Option to book round tickets.