

Flyaway

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

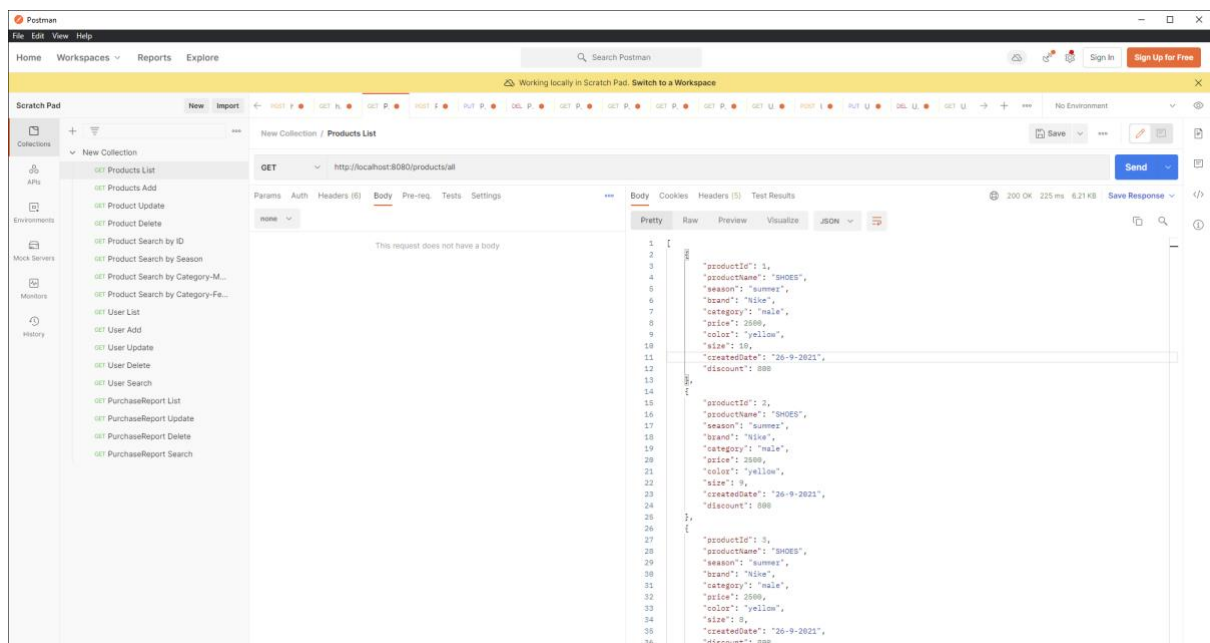
The application has the following capabilities:

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

Postman interface showing a POST request to `http://localhost:8080/products/add`. The request body is a JSON object representing a product:

```
1 {
2   "productName": "Sneakers",
3   "season": "summer",
4   "brand": "Adidas",
5   "category": "male",
6   "price": 2000,
7   "color": "white",
8   "size": 9,
9   "createdDate": "26-9-2021",
10  "discount": 000
11 }
```

The response is a 200 OK status with a 19 ms response time and 337 B body size. The response body is a JSON object:

```
1 {
2   "productId": 39,
3   "productName": "Sneakers",
4   "season": "summer",
5   "brand": "Adidas",
6   "category": "male",
7   "price": 2000,
8   "color": "white",
9   "size": 9,
10  "createdDate": "26-9-2021",
11  "discount": 000
12 }
```

Product Update

Postman interface showing a PUT request to `http://localhost:8080/products/update/2`. The request body is a JSON object representing a product:

```
1 {
2   "productId": 2,
3   "productName": "Hiking",
4   "season": "summer",
5   "brand": "Nike",
6   "category": "male",
7   "price": 6000,
8   "color": "Brown",
9   "size": 9,
10  "createdDate": "26-9-2021",
11  "discount": 000
12 }
```

The response is a 200 OK status with a 34 ms response time and 332 B body size. The response body is a JSON object:

```
1 {
2   "productId": 2,
3   "productName": "Hiking",
4   "season": "summer",
5   "brand": "Nike",
6   "category": "male",
7   "price": 6000,
8   "color": "Brown",
9   "size": 9,
10  "createdDate": "26-9-2021",
11  "discount": 000
12 }
```

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:

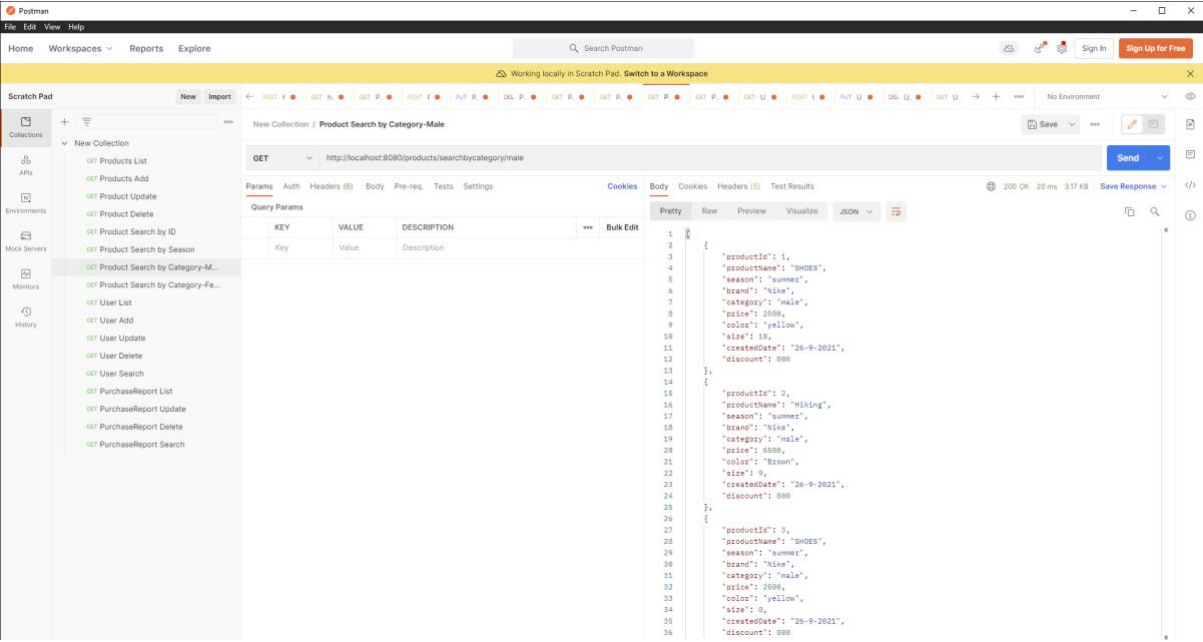
```
1 Returns Nothing
2 Check Database For given Id(4)
3 Will have been deleted
```

Product Search by ID

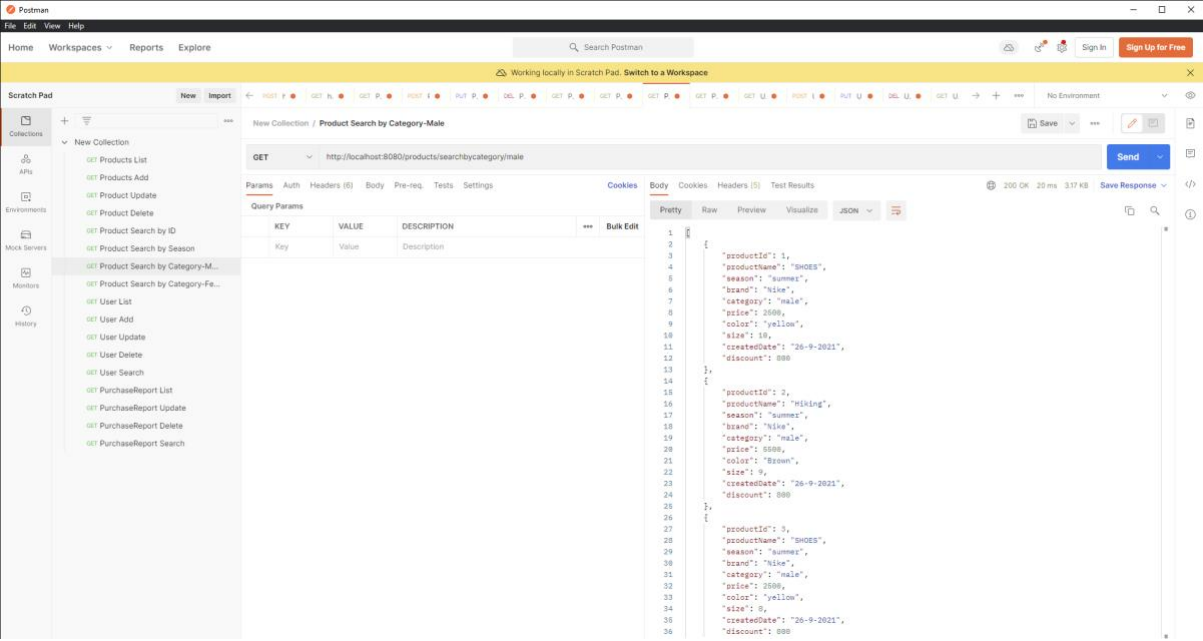
Postman interface showing a GET request to `http://localhost:8080/products/search/5`. The response body is displayed in the "Body" tab, showing a JSON object:

```
{
  "productID": 5,
  "productName": "SHOES",
  "season": "summer",
  "brand": "Nike",
  "category": "Female",
  "color": "2000",
  "size": "yellow",
  "createdDate": "26-9-2021",
  "discount": 800
}
```

Product Search by Season



Passenger Search by Category Male



Passenger Search by Category Female

The screenshot shows the Postman application with a REST client request configured. The request is a GET method to the URL `http://localhost:8080/products/searchbycategory/female`. The response is displayed in the 'Body' tab, showing a JSON array of three product objects. The 'Query Params' tab is also visible, showing a table with columns 'KEY', 'VALUE', and 'DESCRIPTION'.

KEY	VALUE	DESCRIPTION
Key	Value	Description

```
{
  "productId": 6,
  "productName": "SHOES",
  "season": "summer",
  "brand": "nike",
  "category": "female",
  "price": 2000,
  "color": "yellow",
  "size": 9,
  "createdAt": "26-9-2021",
  "discount": 800
},
{
  "productId": 4,
  "productName": "SHOES",
  "season": "summer",
  "brand": "nike",
  "category": "female",
  "price": 2000,
  "color": "yellow",
  "size": 9,
  "createdAt": "26-9-2021",
  "discount": 800
},
{
  "productId": 7,
  "productName": "SHOES",
  "season": "summer",
  "brand": "nike",
  "category": "female",
  "price": 2000,
  "color": "yellow",
  "size": 9,
  "createdAt": "26-9-2021",
  "discount": 800
}
```

User List

The screenshot shows the Postman application with a REST client request configured. The request is a GET method to the URL `http://localhost:8080/users/all`. The response is displayed in the 'Body' tab, showing a JSON array of five user objects. The 'Query Params' tab is also visible, showing a table with columns 'KEY', 'VALUE', and 'DESCRIPTION'.

KEY	VALUE	DESCRIPTION
Key	Value	Description

```
{
  "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": "sauraj@sauraj.com",
  "username": "sauraj",
  "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 with the following fields:

```
{  "emailid": "ankit@ankit.com",  "username": "ankitp",  "password": "1237",  "mobile": 123464283}
```

The response is a JSON object with the following fields:

```
{  "id": 7,  "emailid": "ankit@ankit.com",  "username": "ankitp",  "password": "1237",  "mobile": 123464283}
```

The status is 200 OK, and the response time is 17 ms. The response size is 257 B.

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 with the following fields:

```
{  "id": 4,  "emailid": "kavil@kavil.com",  "username": "kavilchou",  "password": "1237999",  "mobile": 123464183}
```

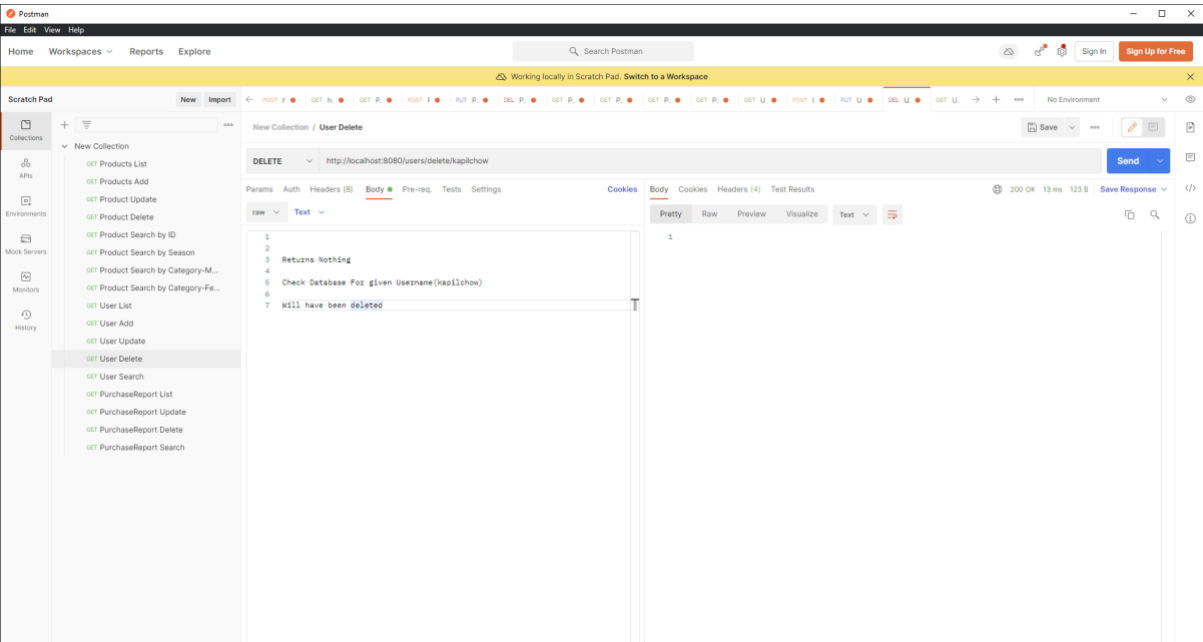
The response is a JSON object with the following fields:

```
{  "id": 4,  "emailid": "kavil@kavil.com",  "username": "kavilchou",  "password": "1237999",  "mobile": 123464183}
```

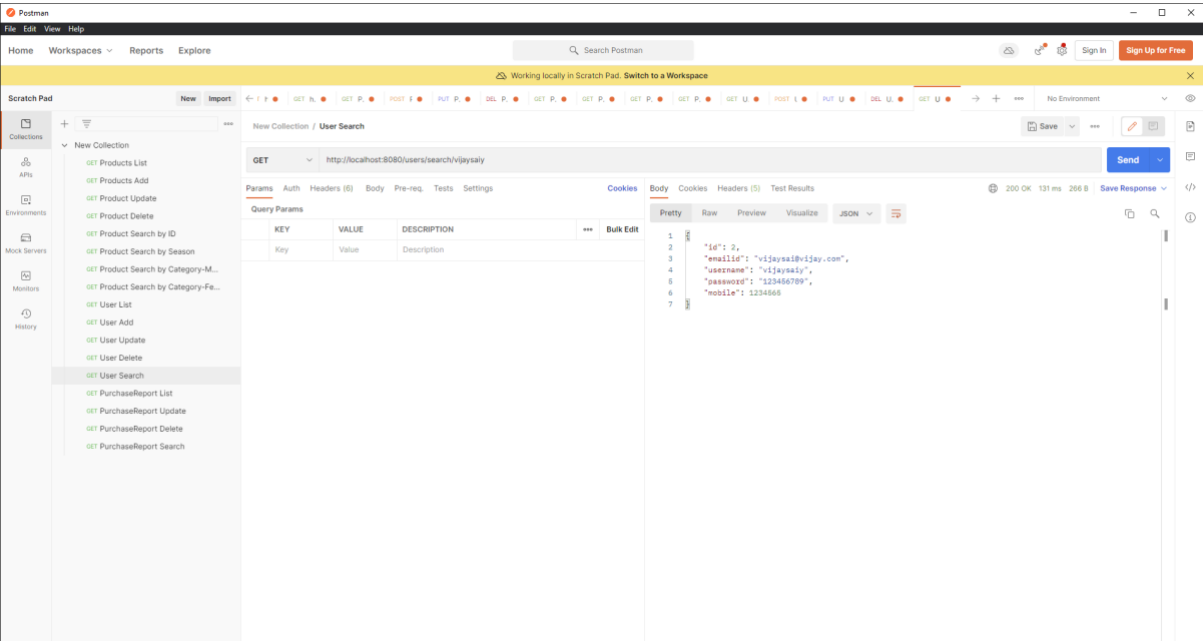
The status is 200 OK, and the response time is 33 ms. The response size is 262 B.

Below the response, there is a note: "Previously The password was 123788".

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 three purchase reports.

KEY	VALUE	DESCRIPTION
Key	Value	Description

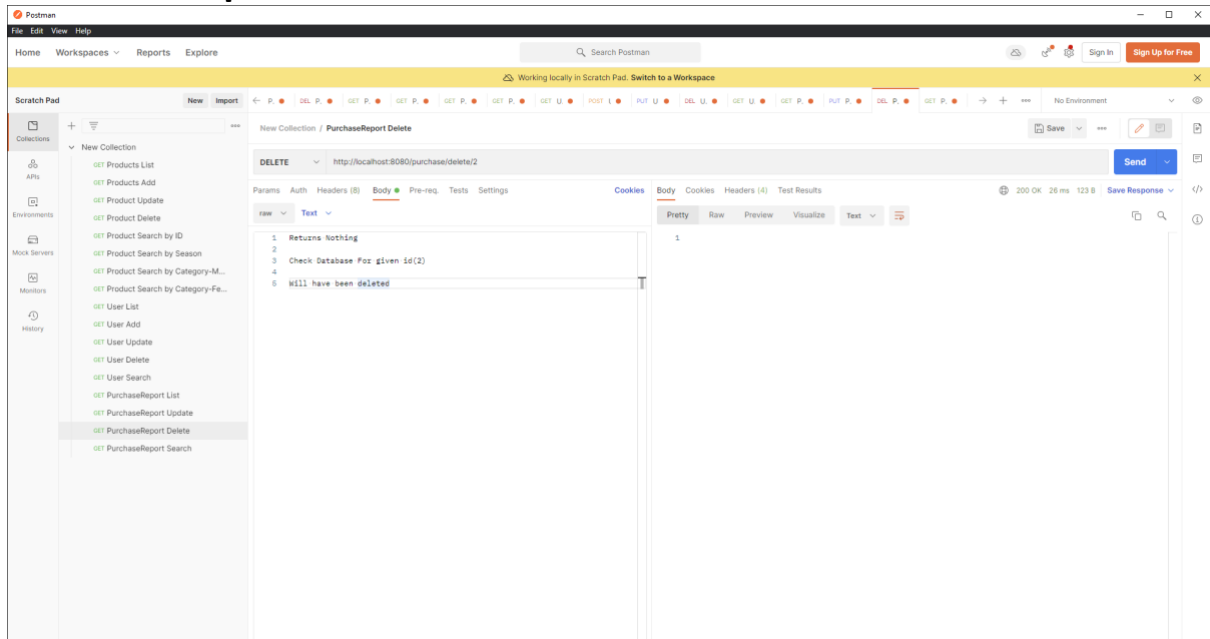
```
1 [
2   {
3     "id": 1,
4     "username": "vijayal",
5     "productName": "shoes",
6     "brand": "adidas",
7     "colour": "yellow",
8     "size": 10,
9     "mobile": 1234567890,
10    "purchaseDate": "26-9-2021"
11  },
12  {
13    "id": 2,
14    "username": "vijen",
15    "productName": "sportshoes",
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": 1245661512,
30    "purchaseDate": "26-9-2021"
31  }
32 ]
```

Purchase Report Update

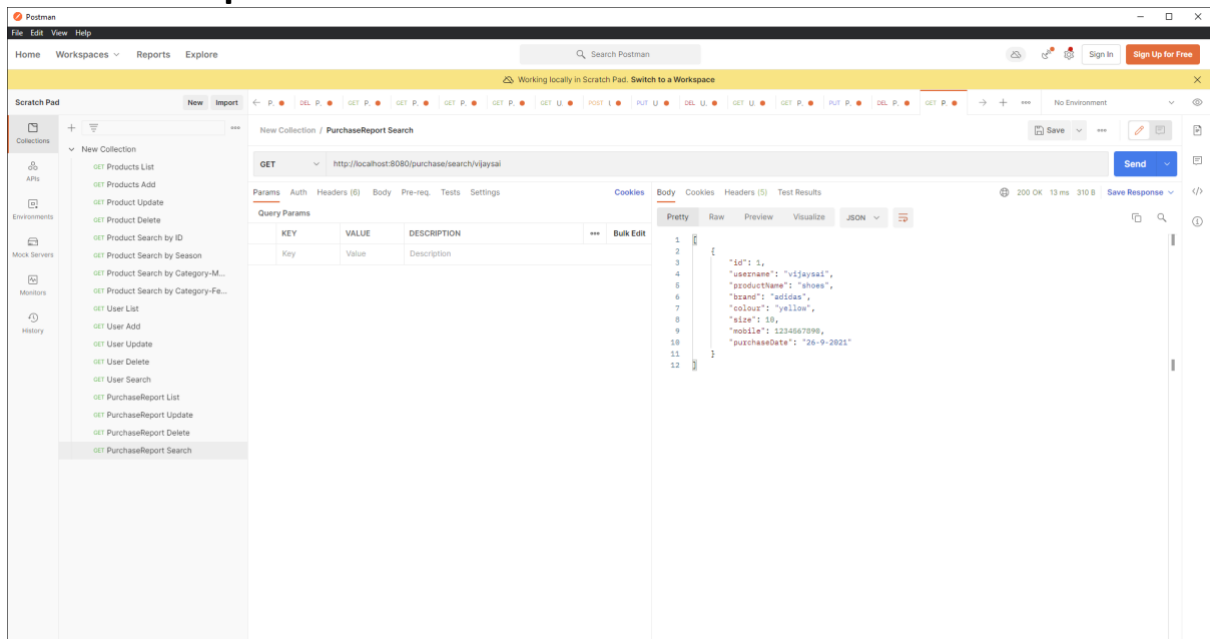
Postman interface showing a PUT request to `http://localhost:8080/purchase/update/2`. The request body is a JSON object for the second purchase report.

```
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.