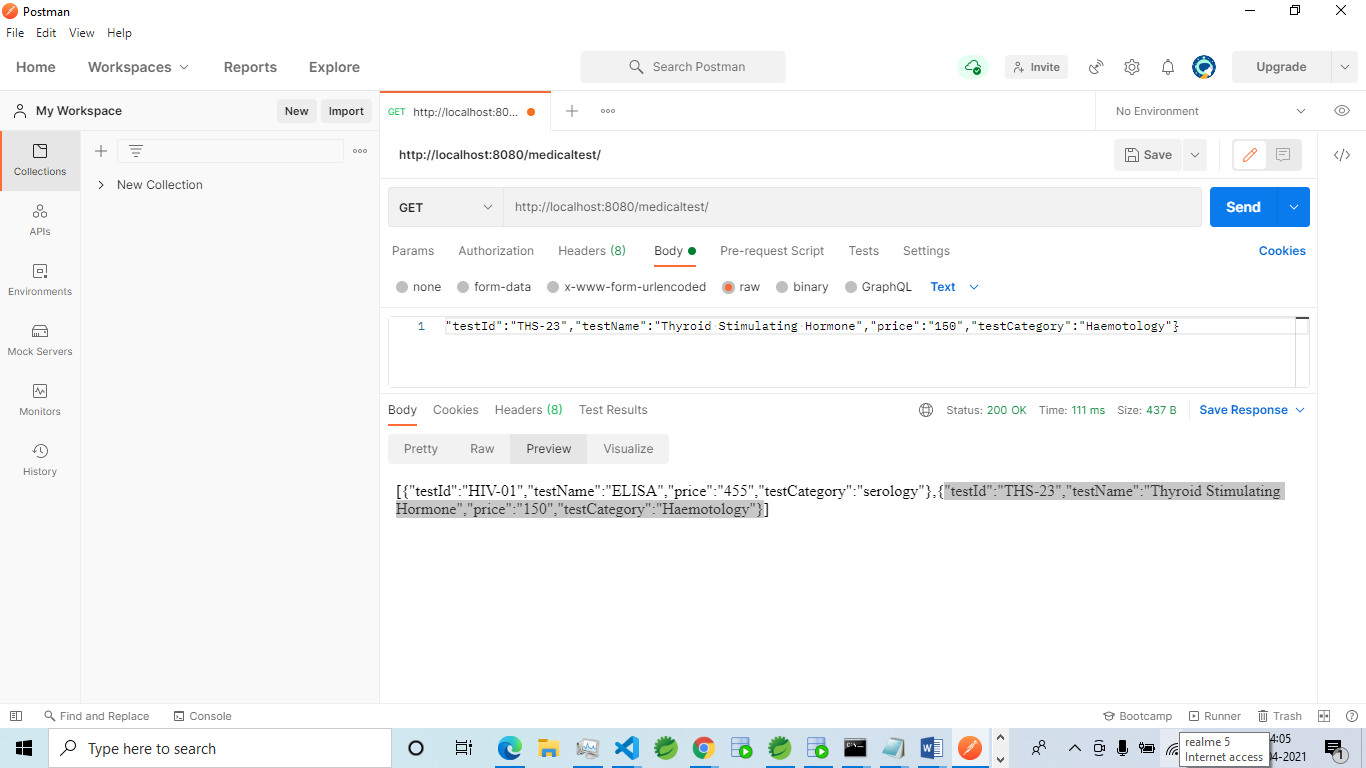
Project4: Medical Diagnostic Laboratory

Back End Development

Note: Back End Development for the project must be done withthe team.

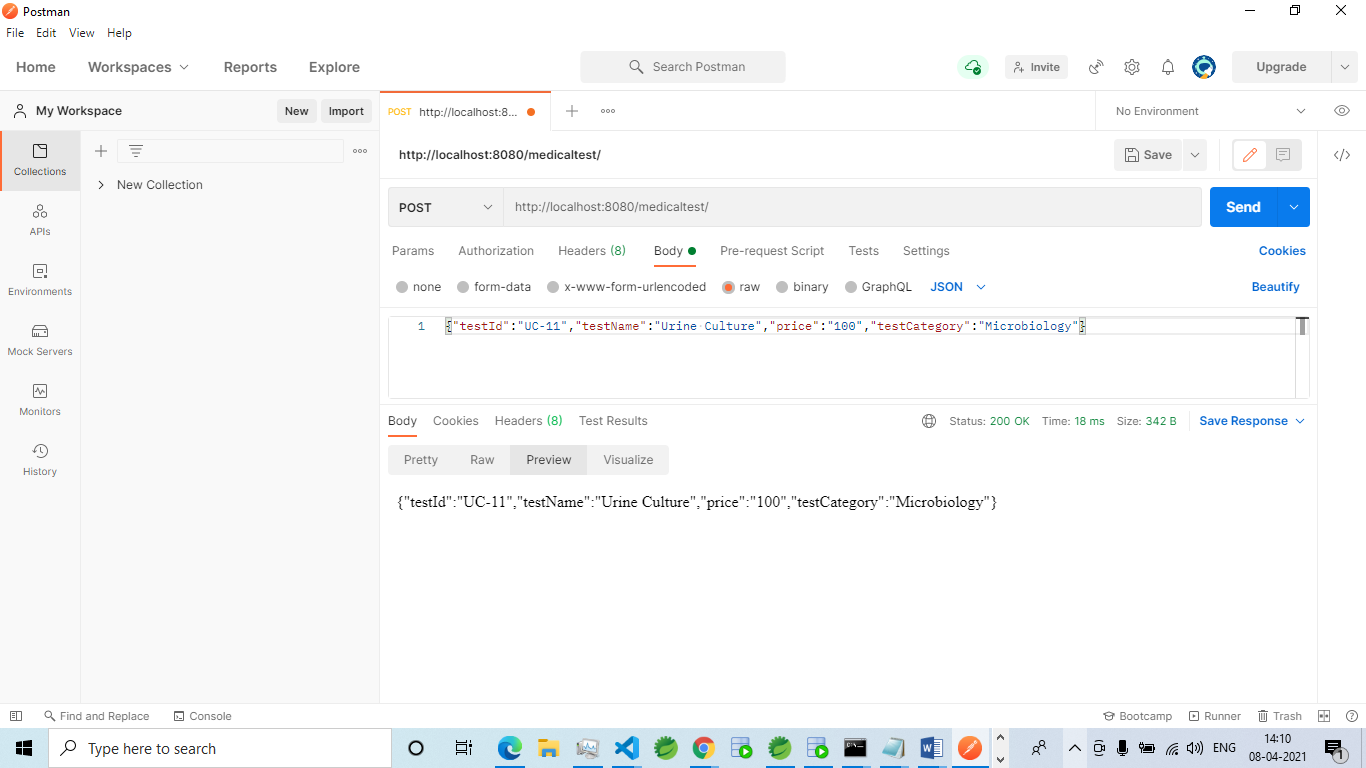
**Review of GET Method using POSTMAN**

Retrieval of allMedicalTest objects is done using GET Method in POSTMAN



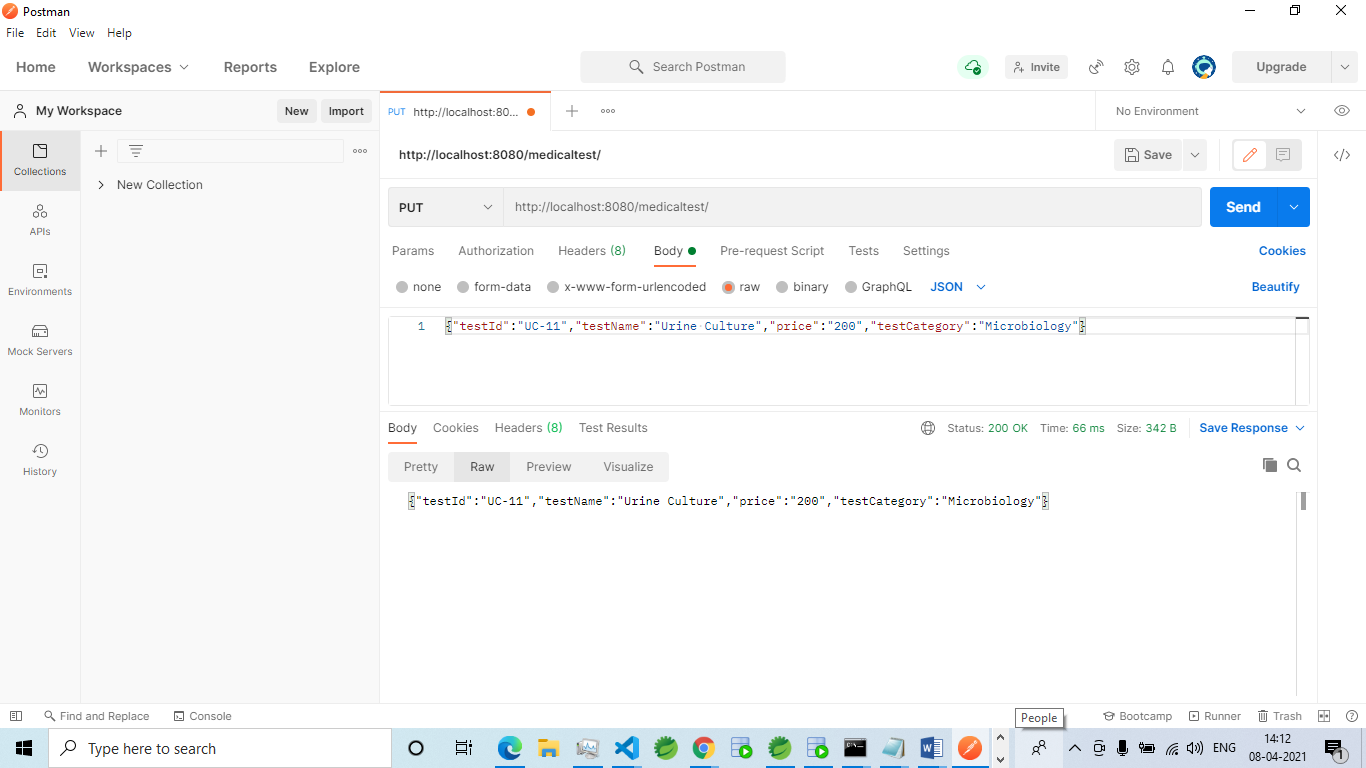
**Review of POST Method using POSTMAN**

Creation of a new object MedicalTest class is done using POST Method in POSTMAN



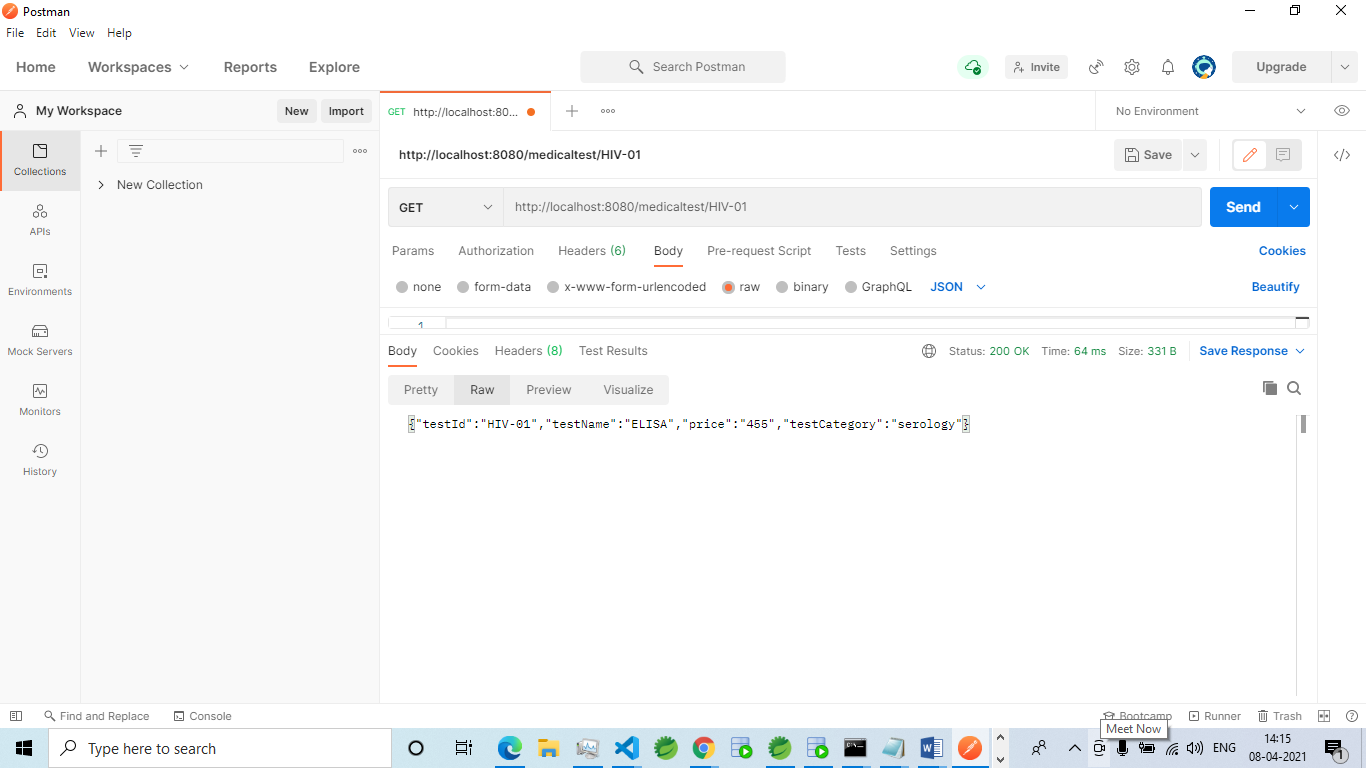
**Review of PUT Method using POSTMAN**

Updation of an object ofMedicalTest class is done using PUT Method in POSTMAN



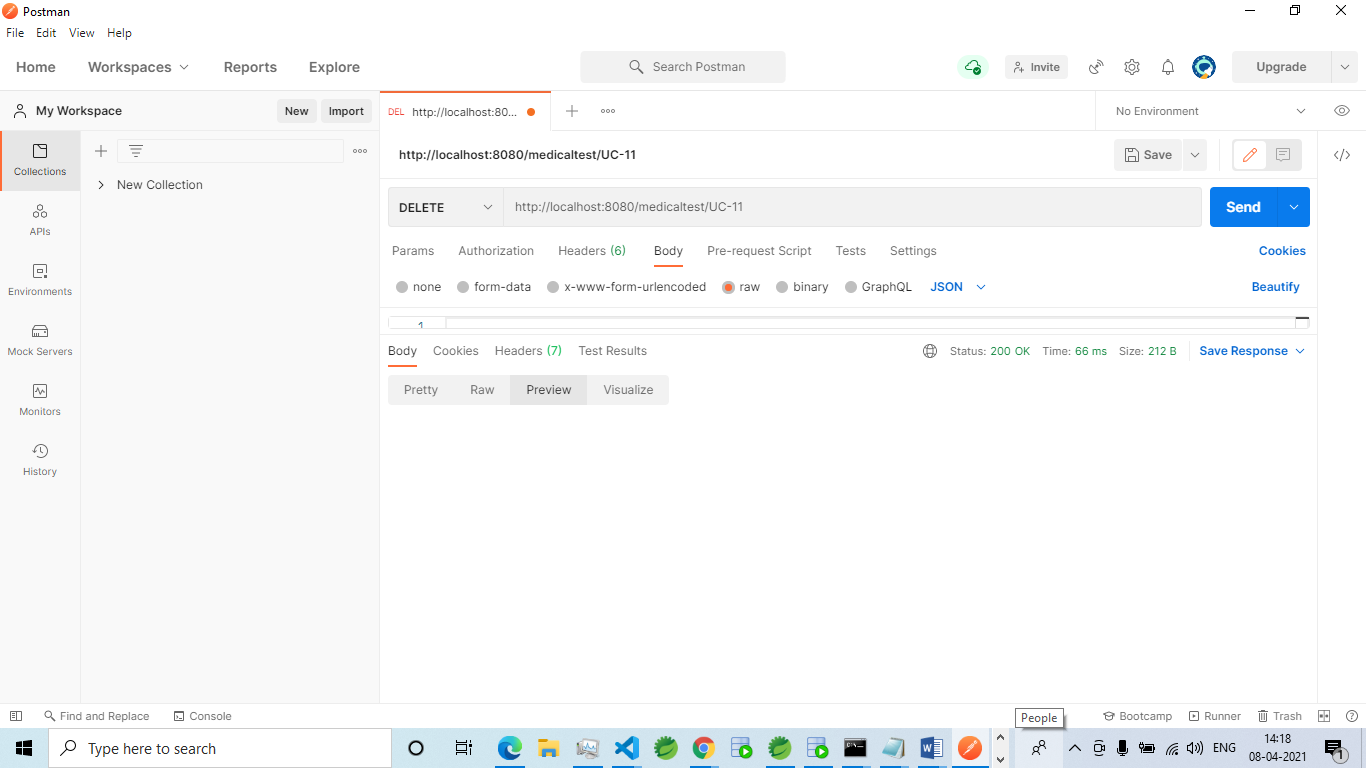
**Review of GET Method by testID using POSTMAN**

Retrieving an object of MedicalTest class by Id is done using GETMethod by testID in POSTMAN



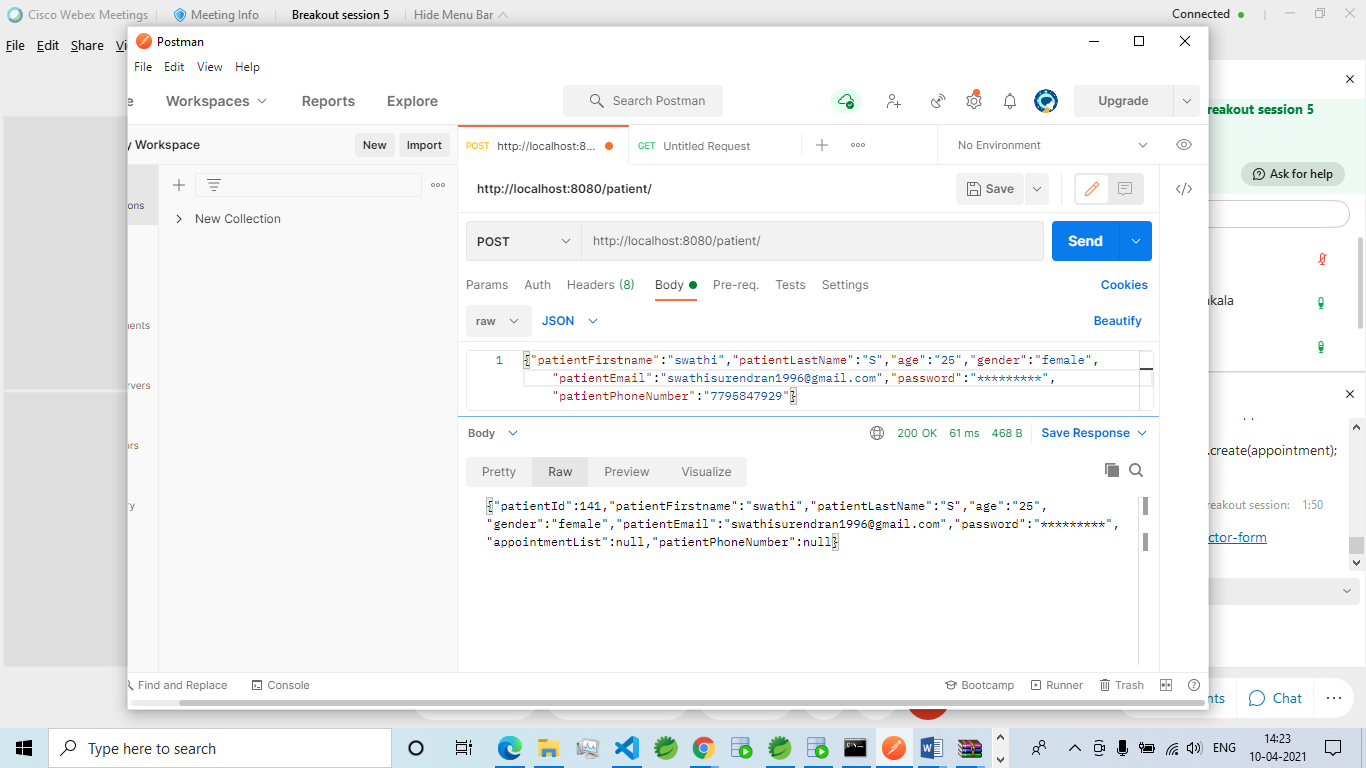
**Review of DELETE Method using POSTMAN**

Deleting of an object of MedicalTest class is done using DELETE Method by testIDin POSTMAN



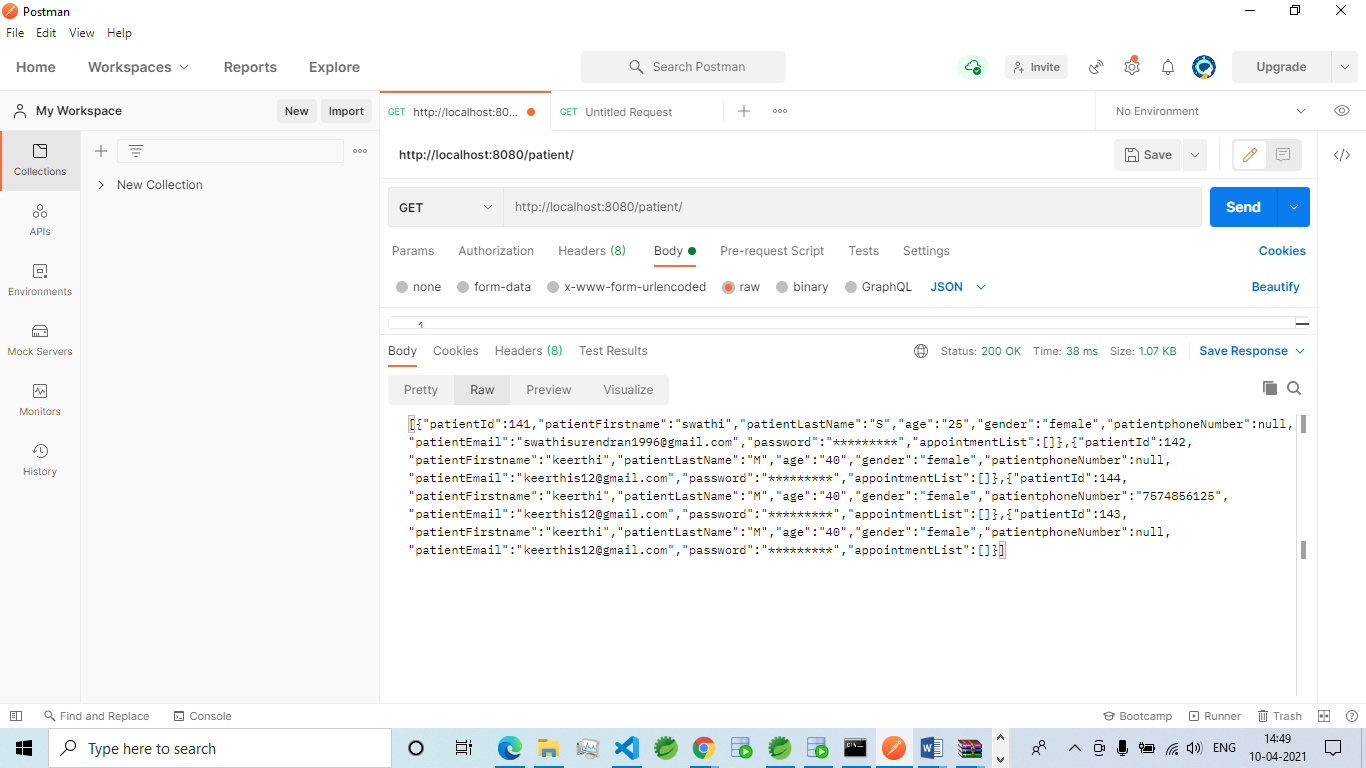
POSTMAN SCREENSHOTS OF PATIENT TABLE:

**Review of POST Method using POSTMAN**

 Creation of Patient object using POST Method in POSTMAN

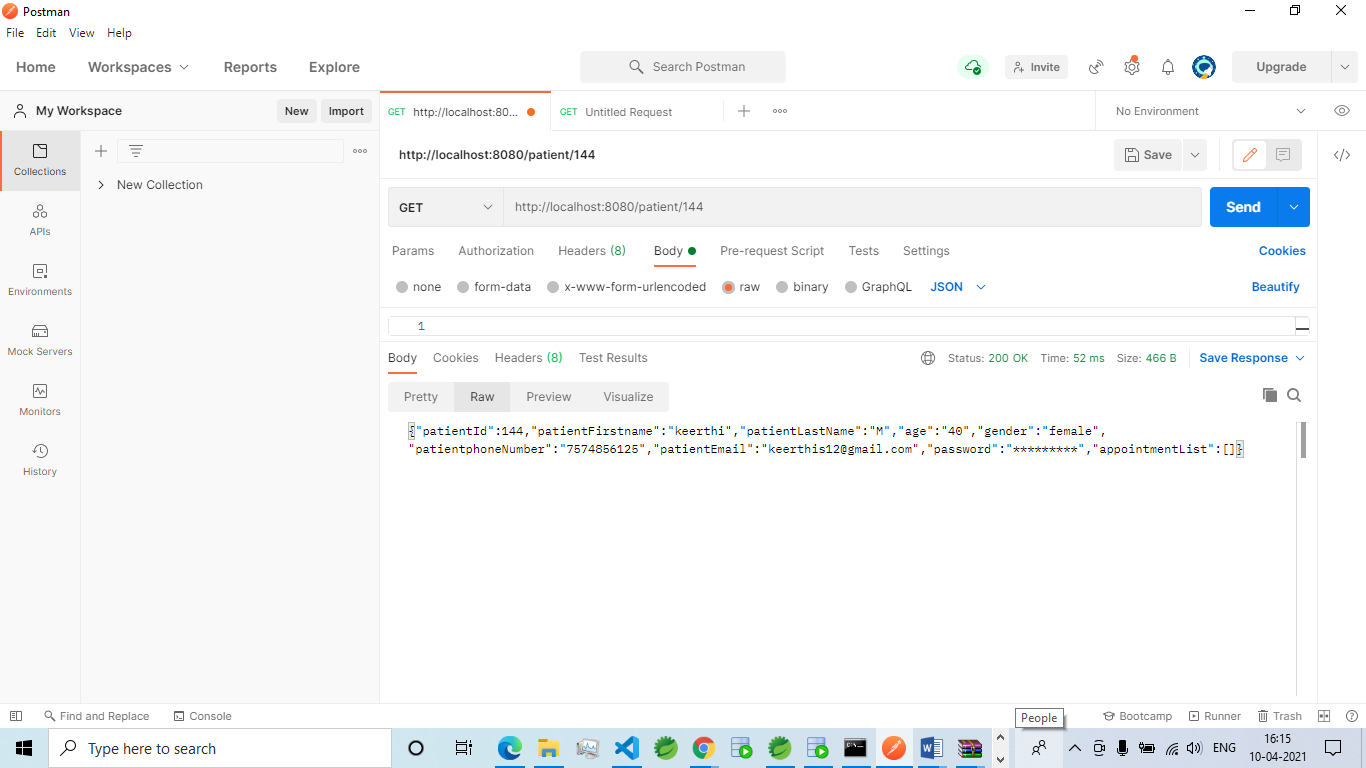
**Review of Get Method using POSTMAN**

Retrieval ofall Patient objects using POST Method in POSTMAN



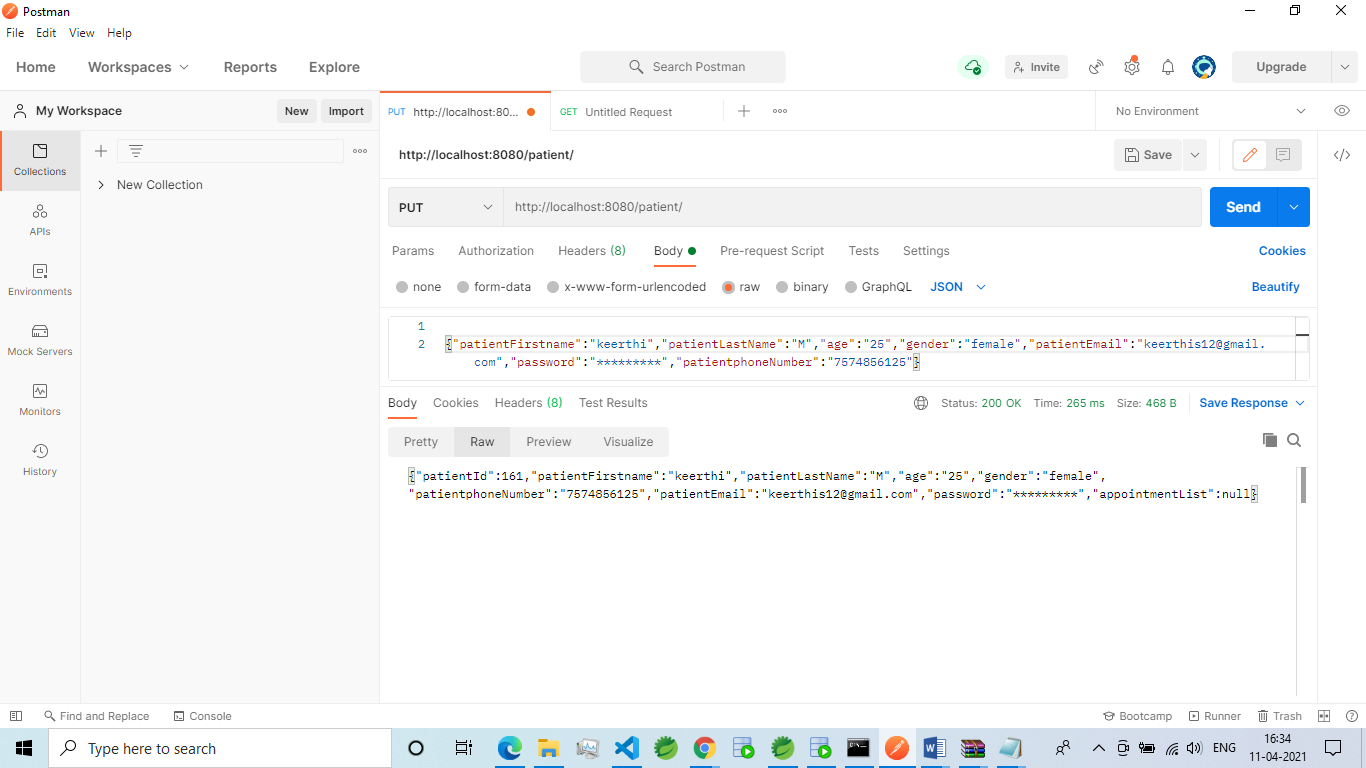
**Review of Get Method bypatientIdusing POSTMAN**

Retrieval ofa Patient object by patientId using POST Method in POSTMAN



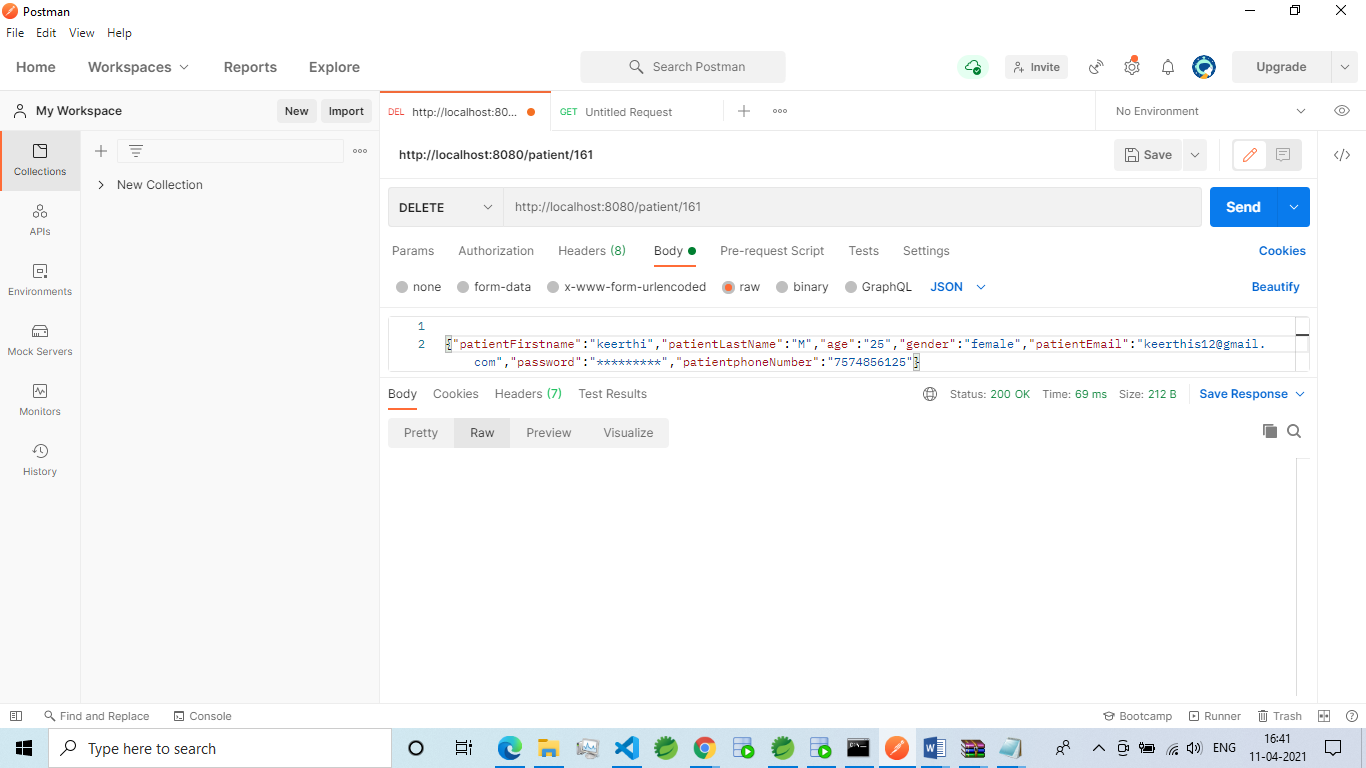
**Review of PUT Method using POSTMAN**

Updation of a Patient object field using PUT Method in POSTMAN



**Review of DELETE Method by Id using POSTMAN**

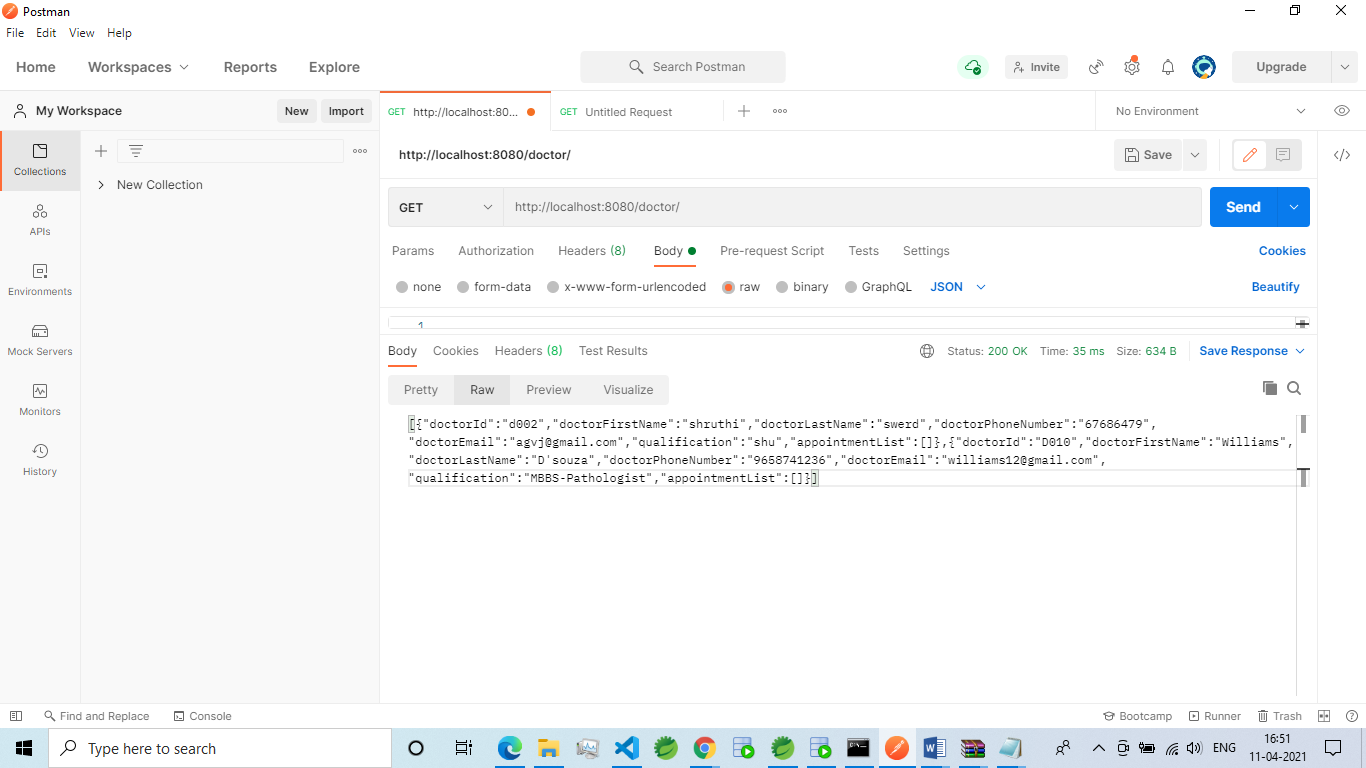
Deletion of a Doctor object by Id using DELETE Method in POSTMAN



DOCTOR:

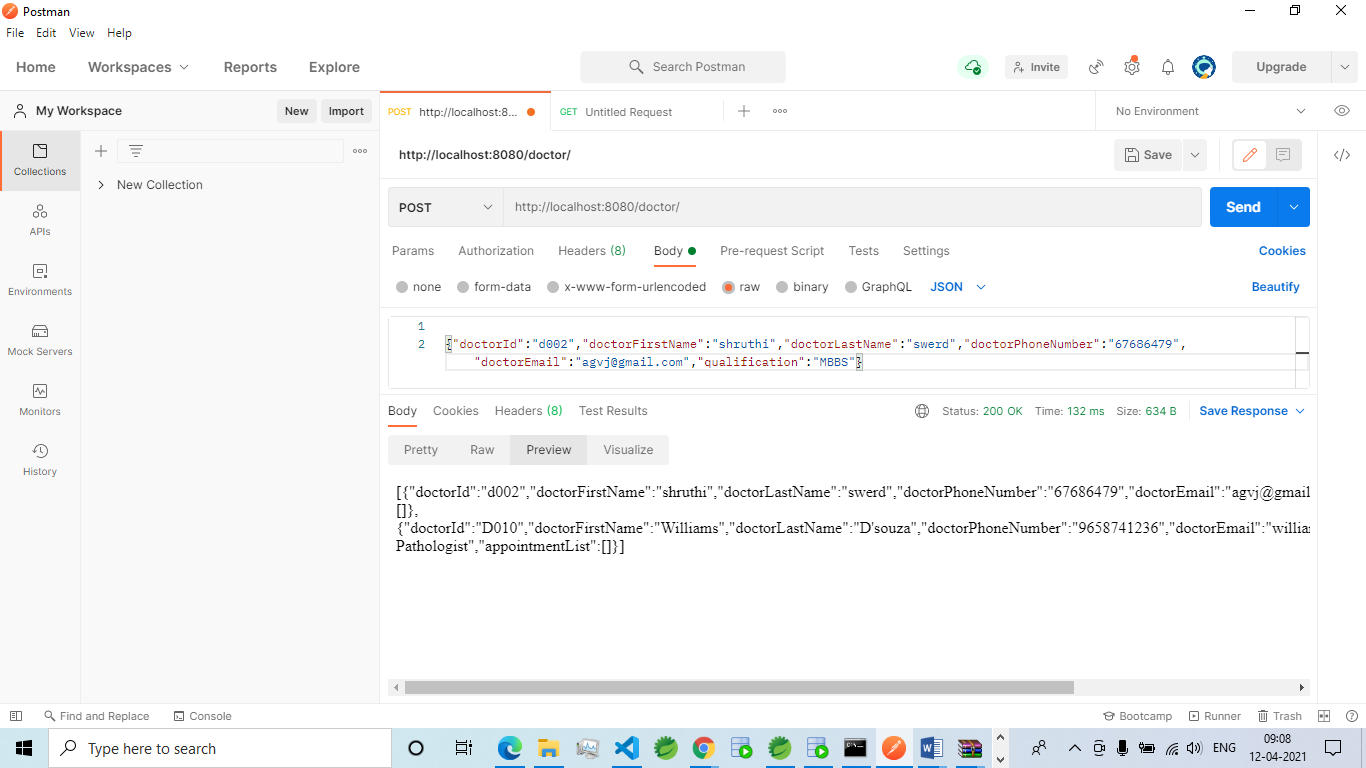
**Review of GET Method using POSTMAN**

Retrieval of a Doctorobject using GET Method in POSTMAN



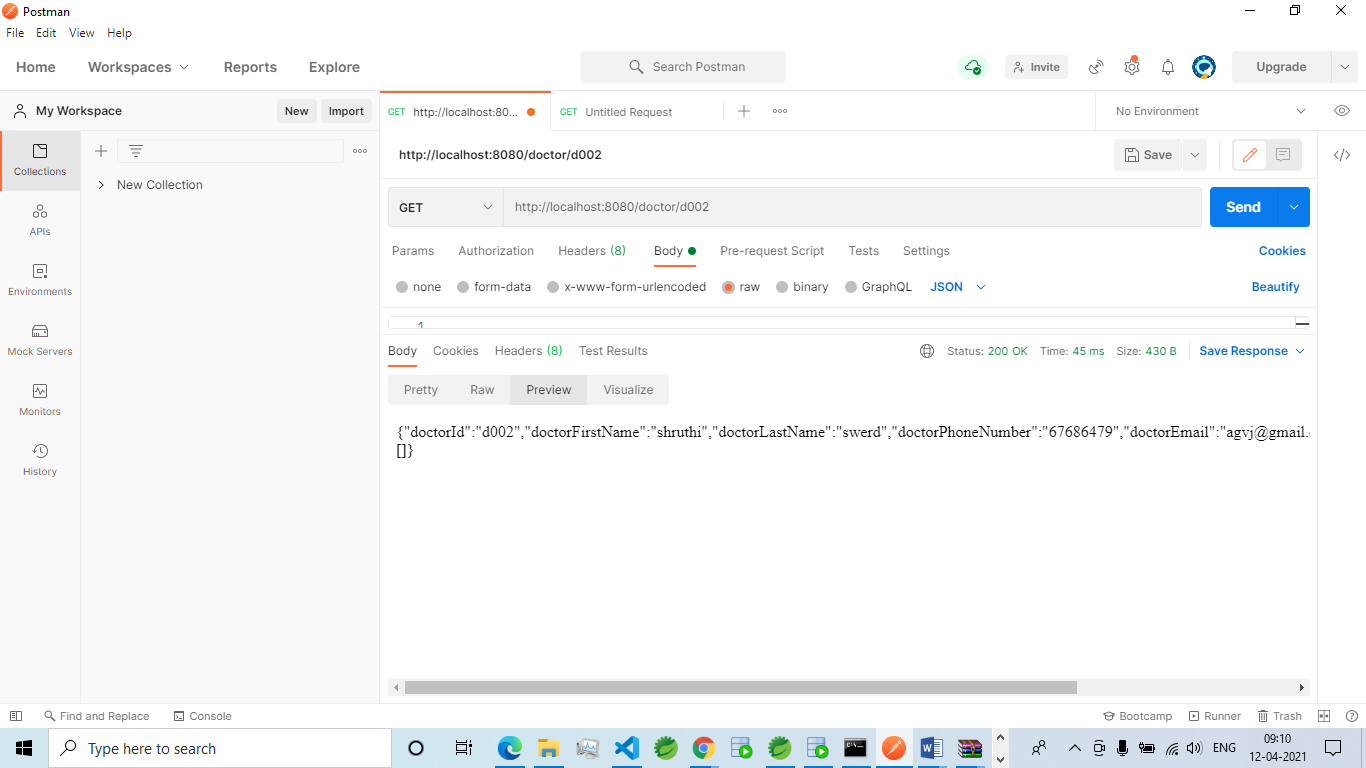
**Review of POST Method using POSTMAN**

Adding of Doctor object using POST Method in POSTMAN



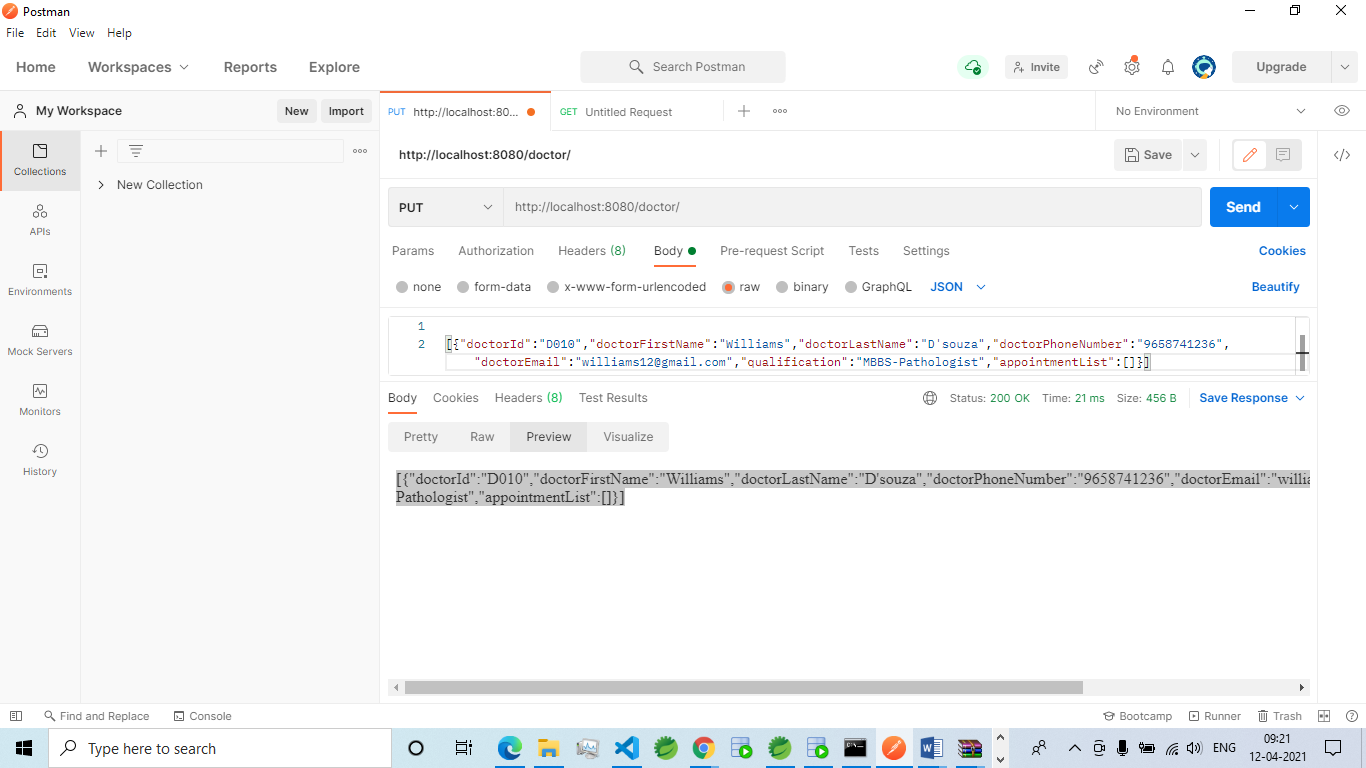
**Review of GET Method by Id using POSTMAN**

Retrieval of a Doctor object by Id using POST Method in POSTMAN



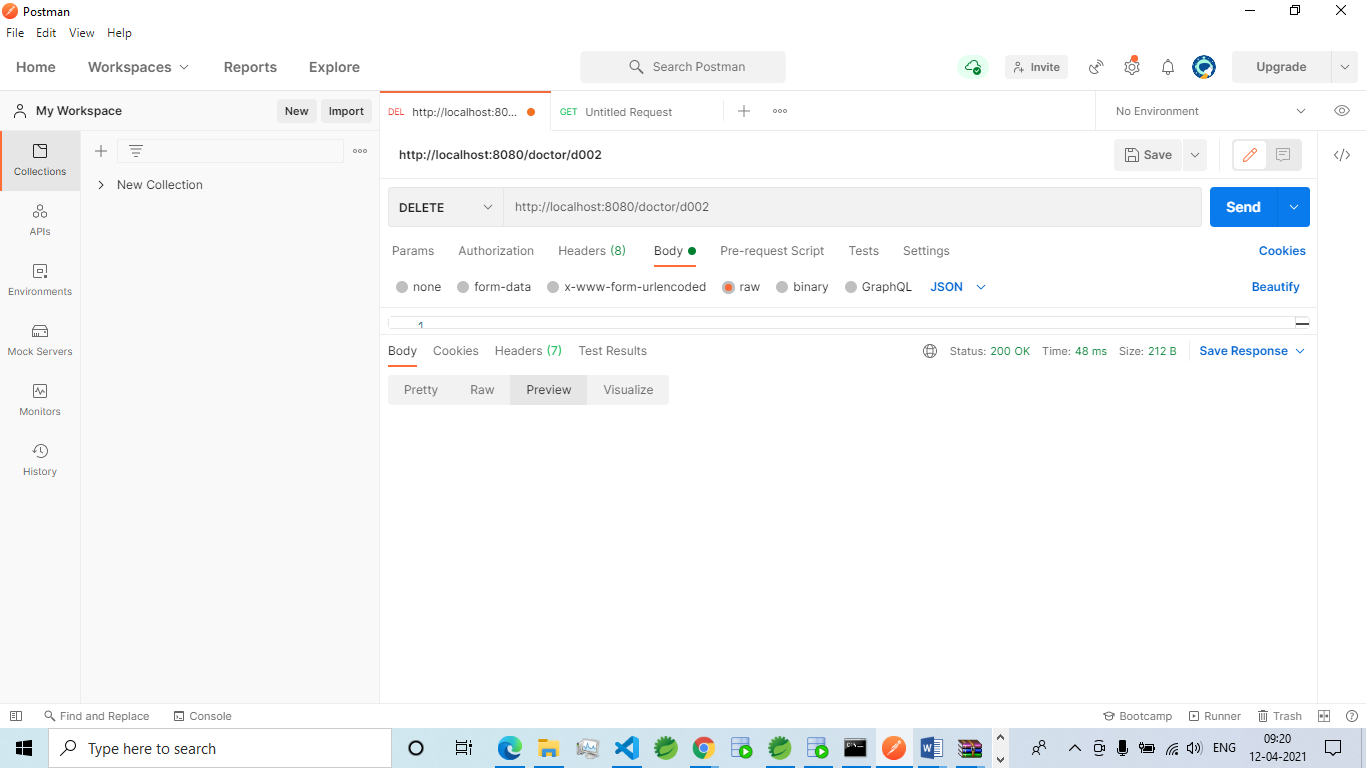
**Review of PUT Method using POSTMAN**

Retrieval of Doctorobject using PUT Method in POSTMAN



**Review of DELETE Method by Id using POSTMAN**

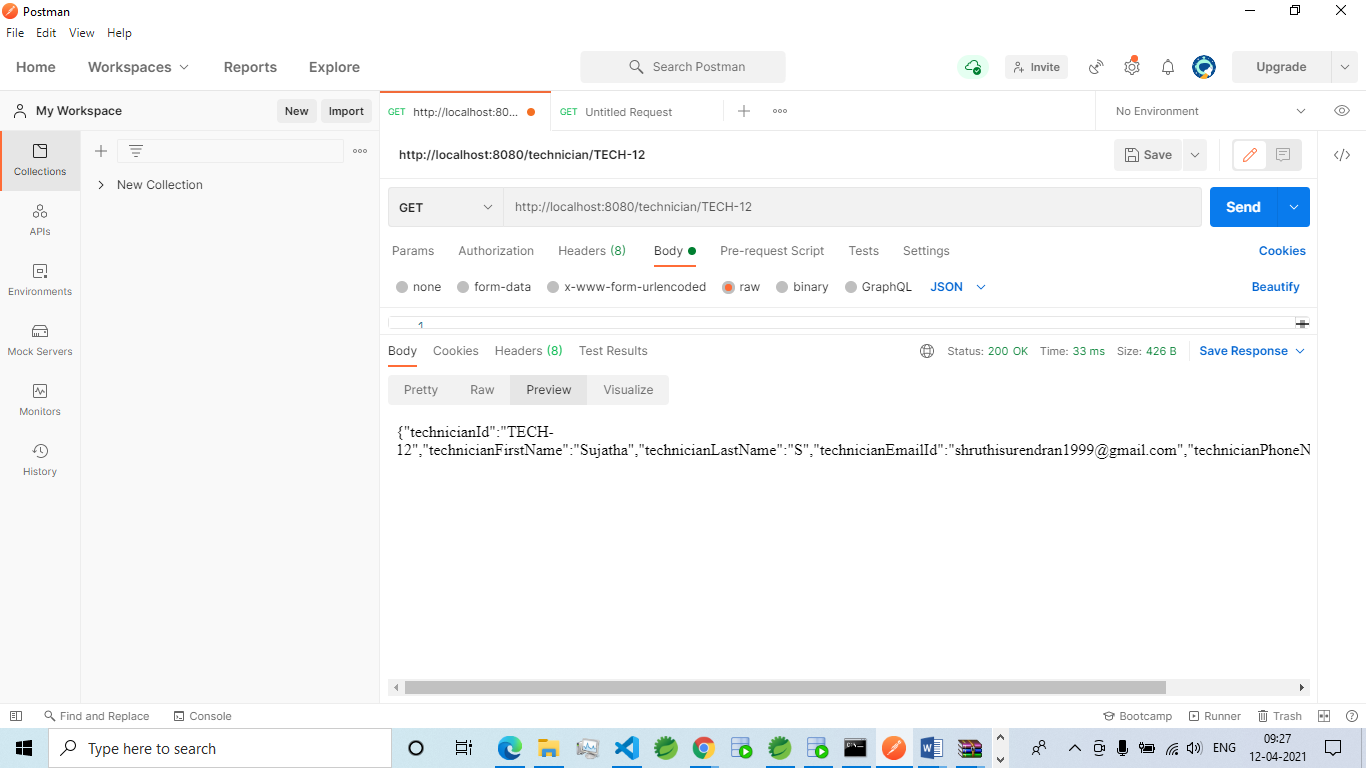
Deletion of a Doctor object by Id using DELETE Method in POSTMAN



TECHNICIAN:

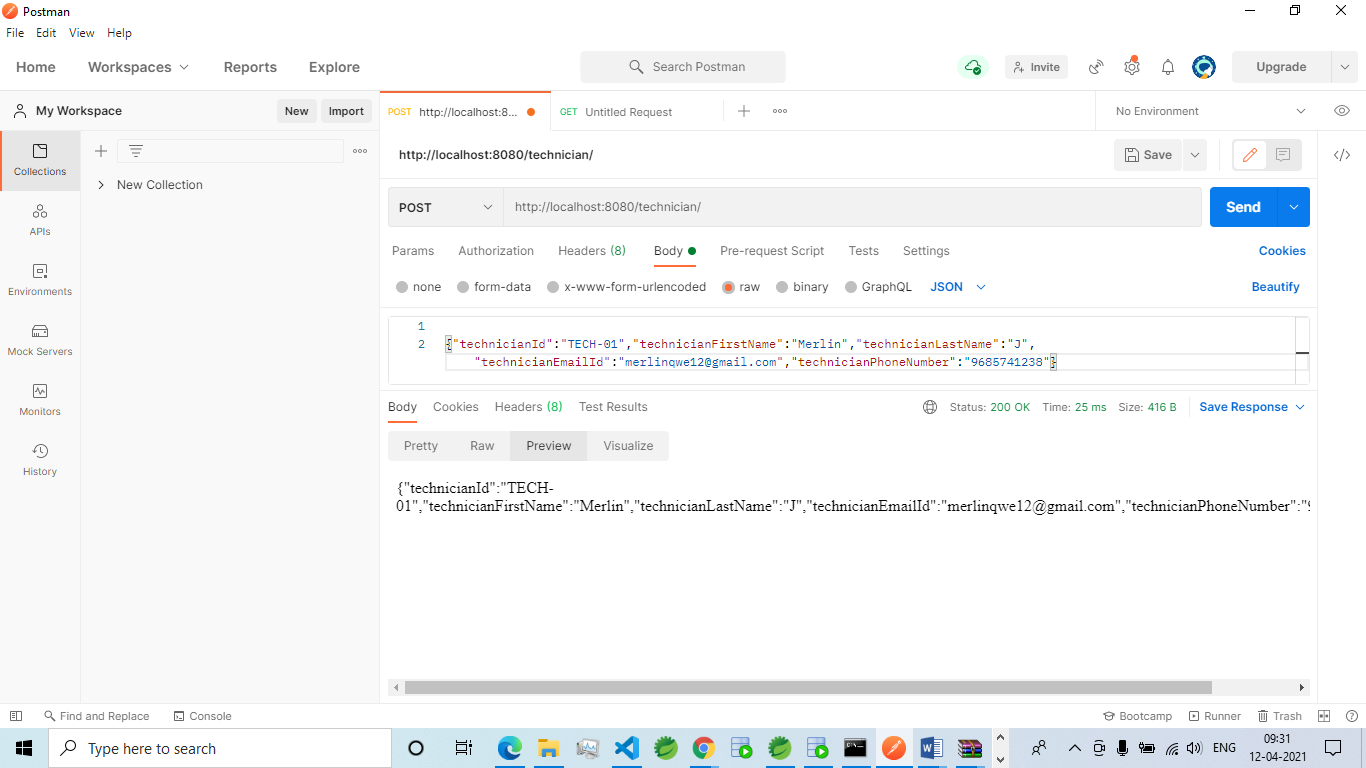
**Review of GET Method using POSTMAN**

Retrieval of a Technicianobject using GET Method in POSTMAN



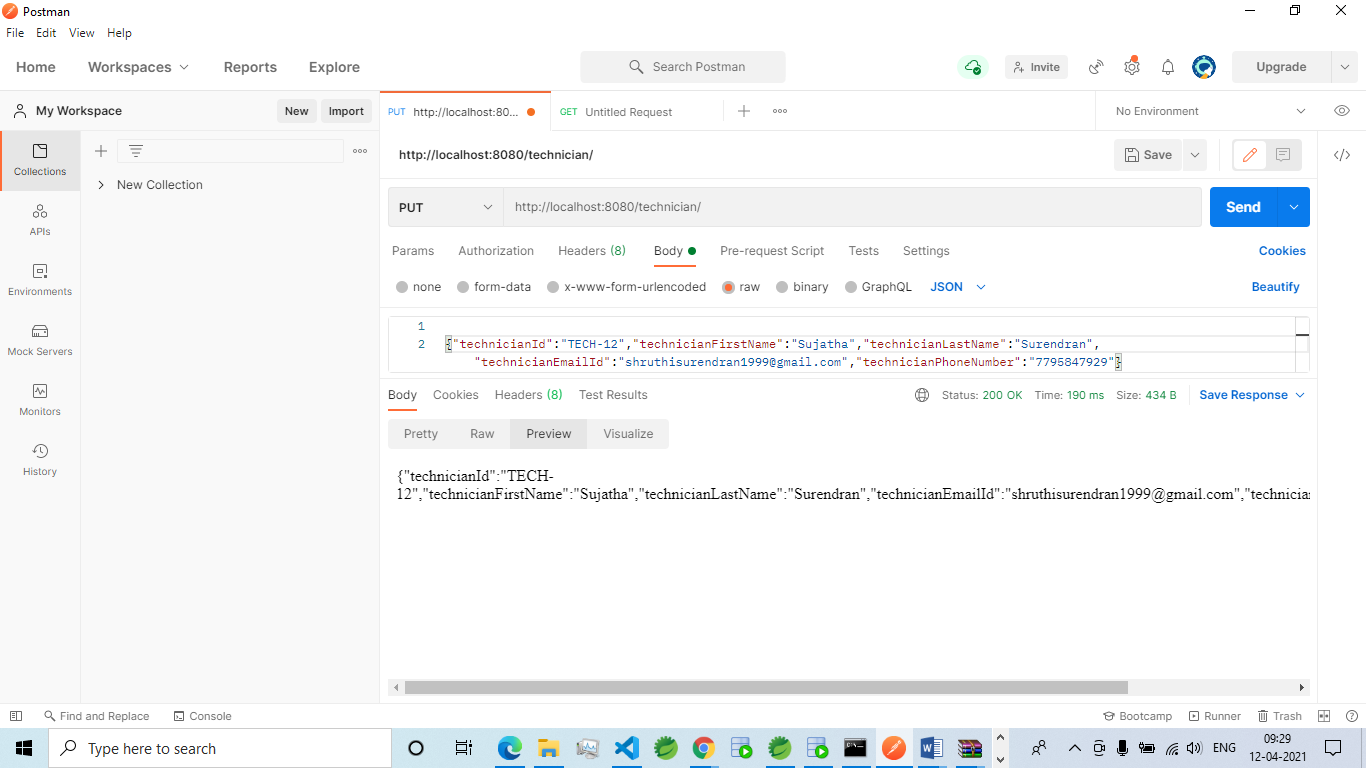
**Review of POST Method using POSTMAN**

Creation of a Technician object using POST Method in POSTMAN



**Review of PUTMethod using POSTMAN**

Updation of a Technician object using PUT Method in POSTMAN

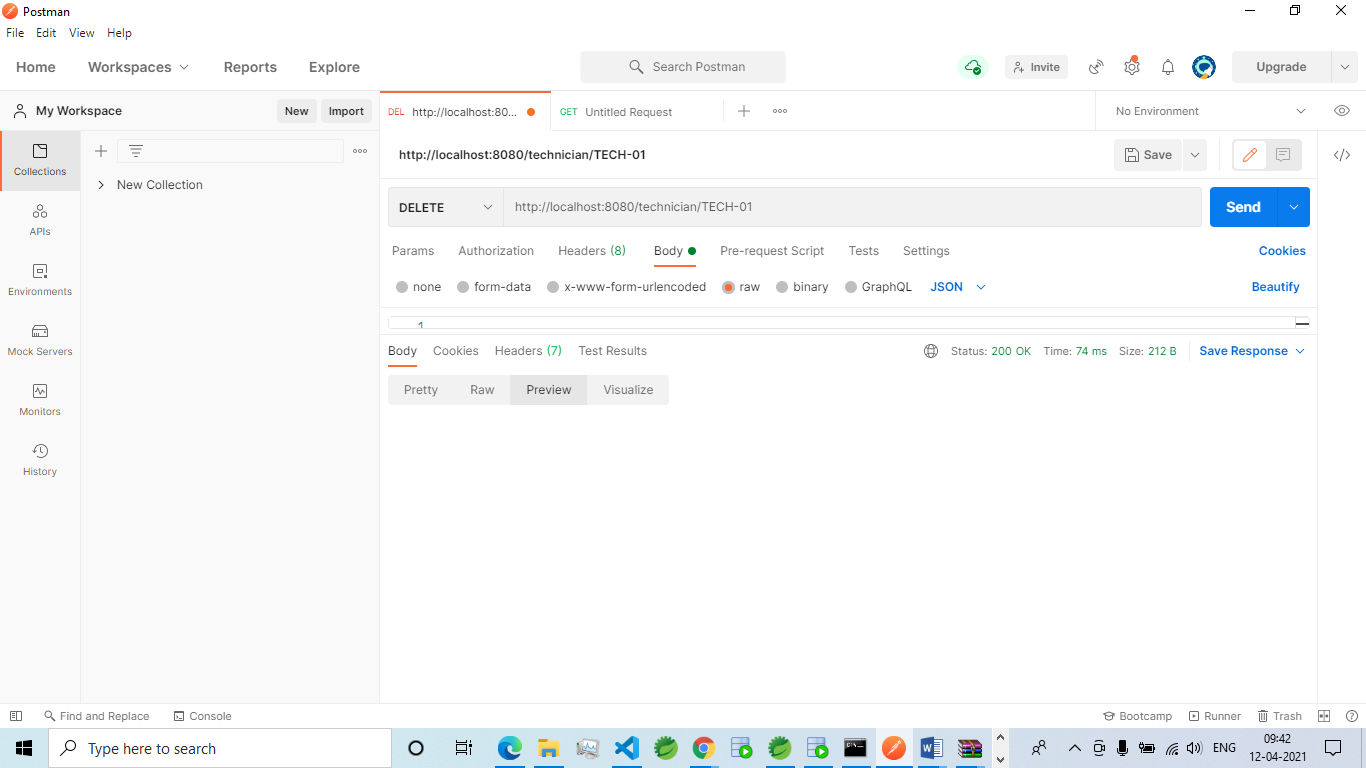
Review

Review of POSTMethod using POSTMAN

Creation of a Technician object using POST Method in POSTMAN

**Review of DELETE Method using POSTMAN**

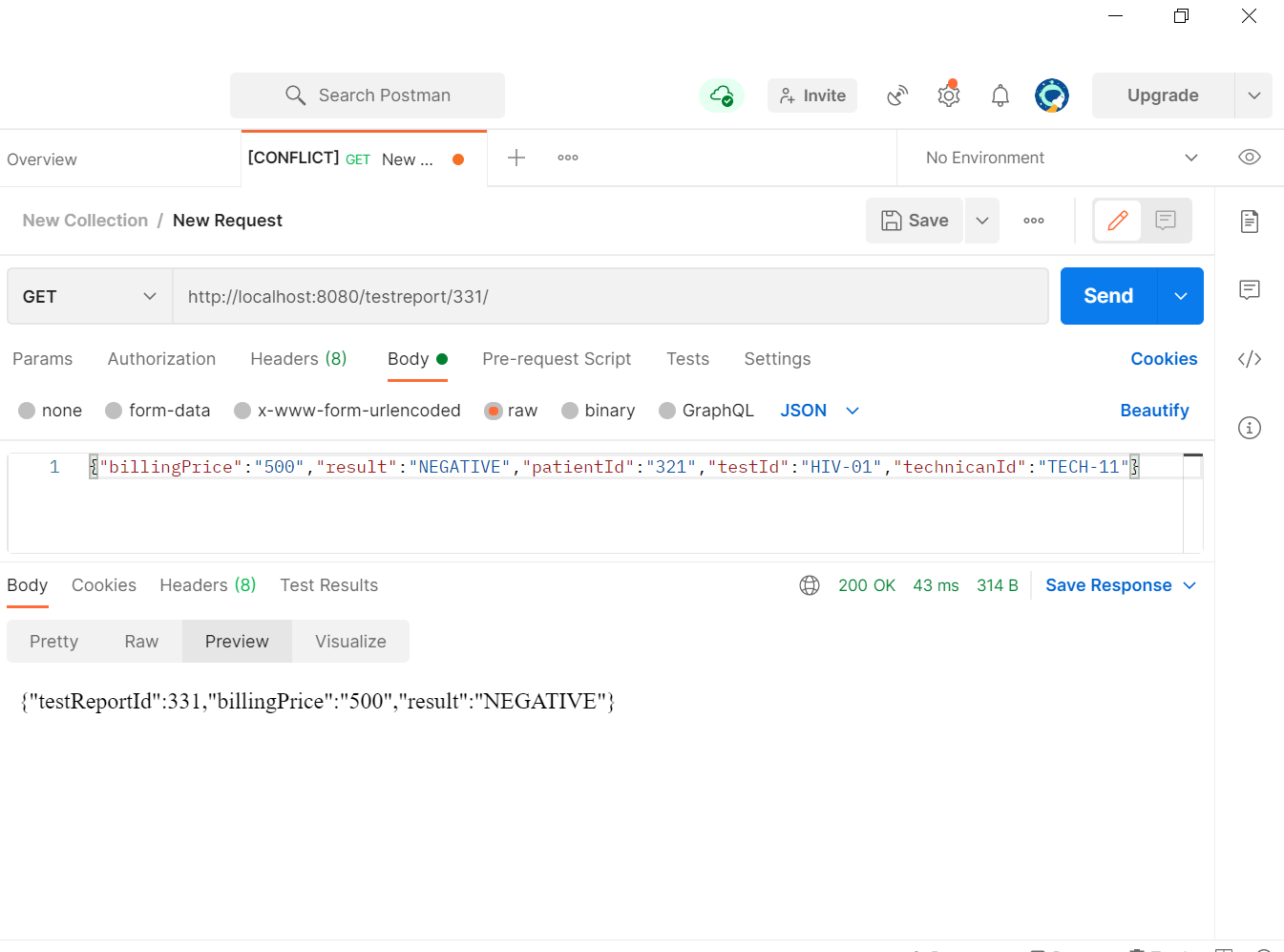
Deletion of a Technician object using DELETE Method in POSTMAN



TESTREPORT:

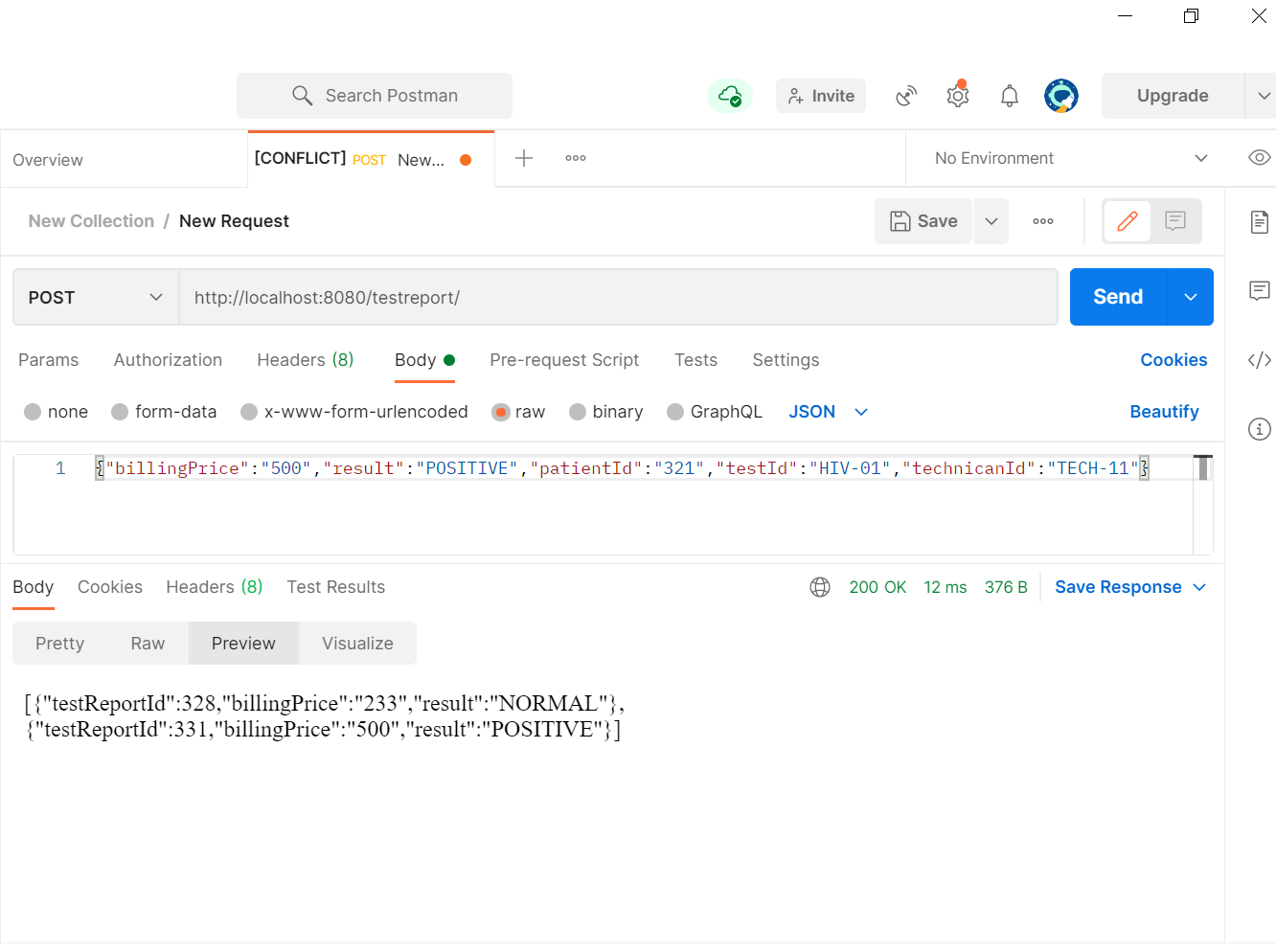
**Review of GET Method using POSTMAN**

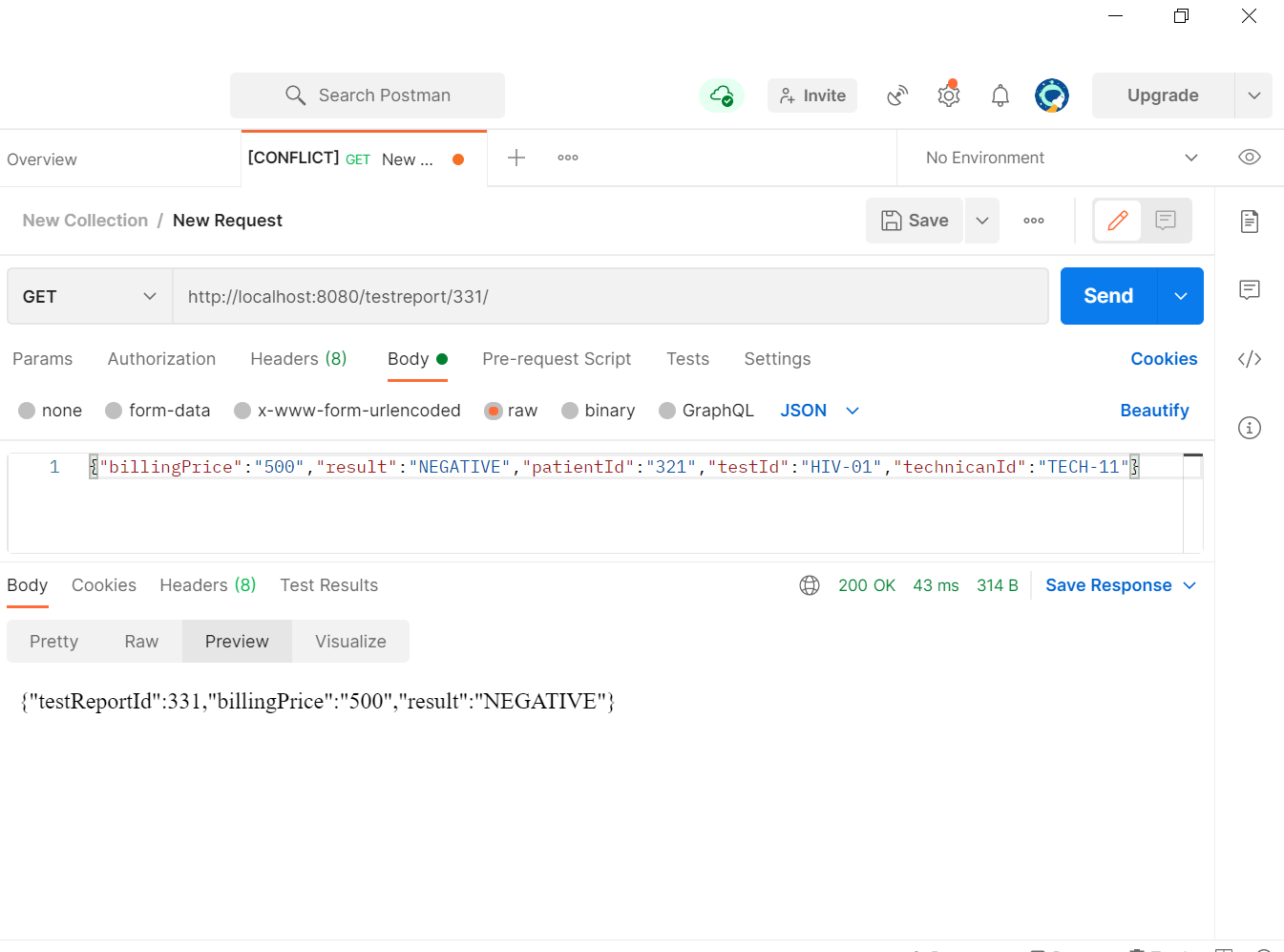
Retrieval of testreport objects is done using GET Method in POSTMAN



**Review of POST Method using POSTMAN**

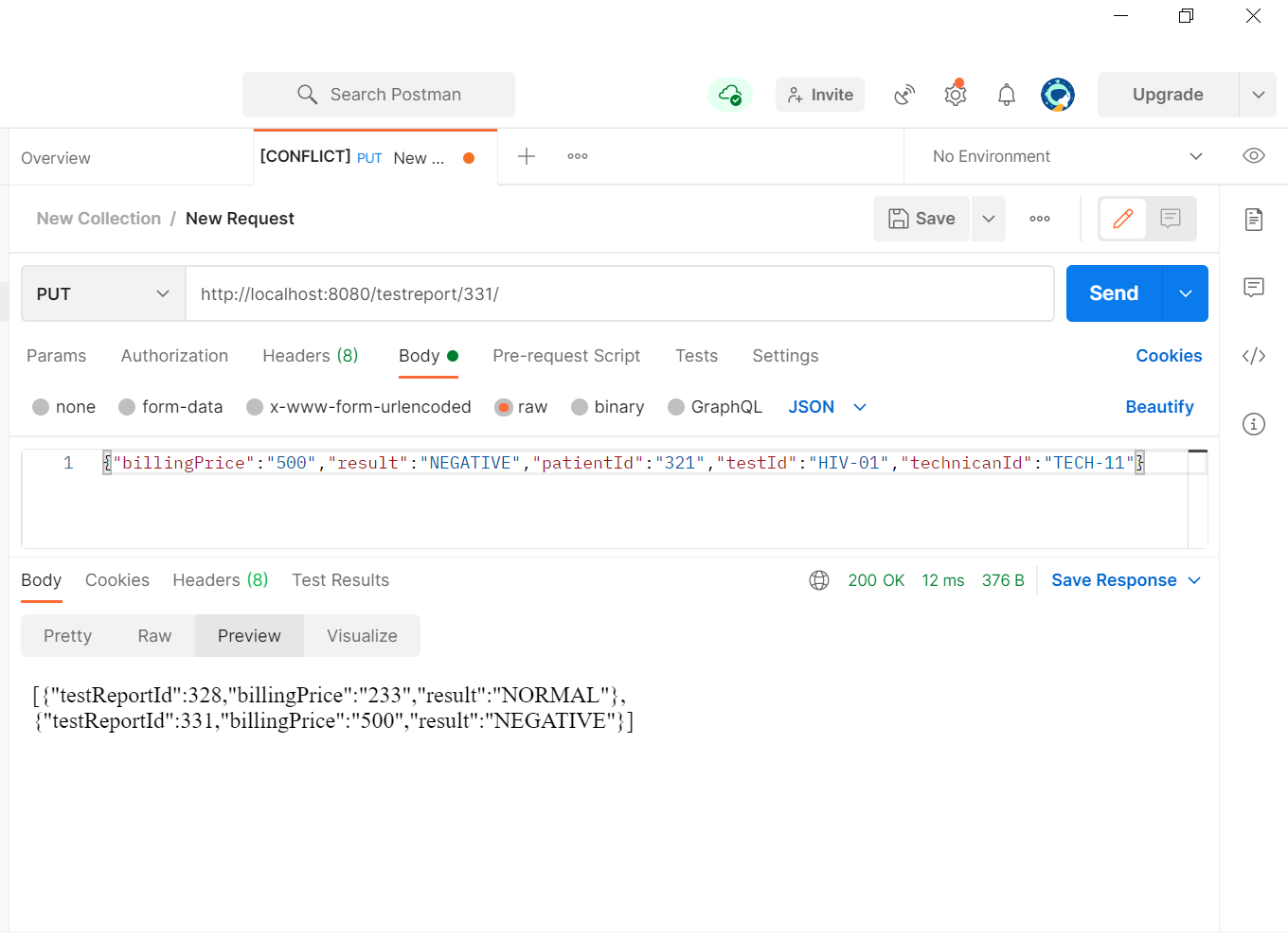
Creation of a new object testreport class is done using POST Method in POSTMAN





**Review of PUT Method using POSTMAN**

Updation of an object of testreport class is done using PUT Method in POSTMAN



|  |  |
| --- | --- |
| **Review of DELETE Method using POSTMAN**  Deleting of an object of testreport class is done using DELETE Method by testID in POSTMAN |  |

**FUNCTIONAL SPECIFICATION**

|  |  |
| --- | --- |
| **ProjectCode:** |  |
| **ProjectName:** | Medical Diagnostic Laboratory |

**FUNCTIONAL SPECIFICATION**

**Table of Contents**

1. INTRODUCTION 3

2. SYSTEM OVERVIEW 3

3. SUB-SYSTEM DETAILS 4

4. DATA ORGANIZATION 5

5. REST APIs to be Built 6

6. ASSUMPTIONS 9

7. EXPECTATIONS 9

8. ACCEPTANCE CRITERIA 9

9. TRACEABILITY TO REQUIREMENTS 9



**FUNCTIONAL SPECIFICATION**

**1 Introduction**

Health Technologies Ltd is a company which builds a software system which is responsible for analyzing testreports of a patient.

Health Technologies Ltd plans to develop "Medical Diagnostic Laboratory " - web application [J2EE Batches - Web Application], where users can register, login, tests are carried out on clinical specimens to obtain information about the health of a patient.

**Scope and Overview:**

The scope of the “Medical Diagnostic Laboratory” will be to provide the functionality as described below. The system will be developed on a Windows operating system using Java/J2EE, Hibernate, Spring.

**2 System Overview**

The “Medical Diagnostic Laboratory” should support basic functionalities (explained in section 2.1) for all below listed users.

* Administrator (A)
* Patient(P)

***2.1 Authentication & Authorization***

***2.1.1 Authentication****:*

Any end-user should be authenticated using a unique patientid and password.

***2.1.2*** ***Authorization***

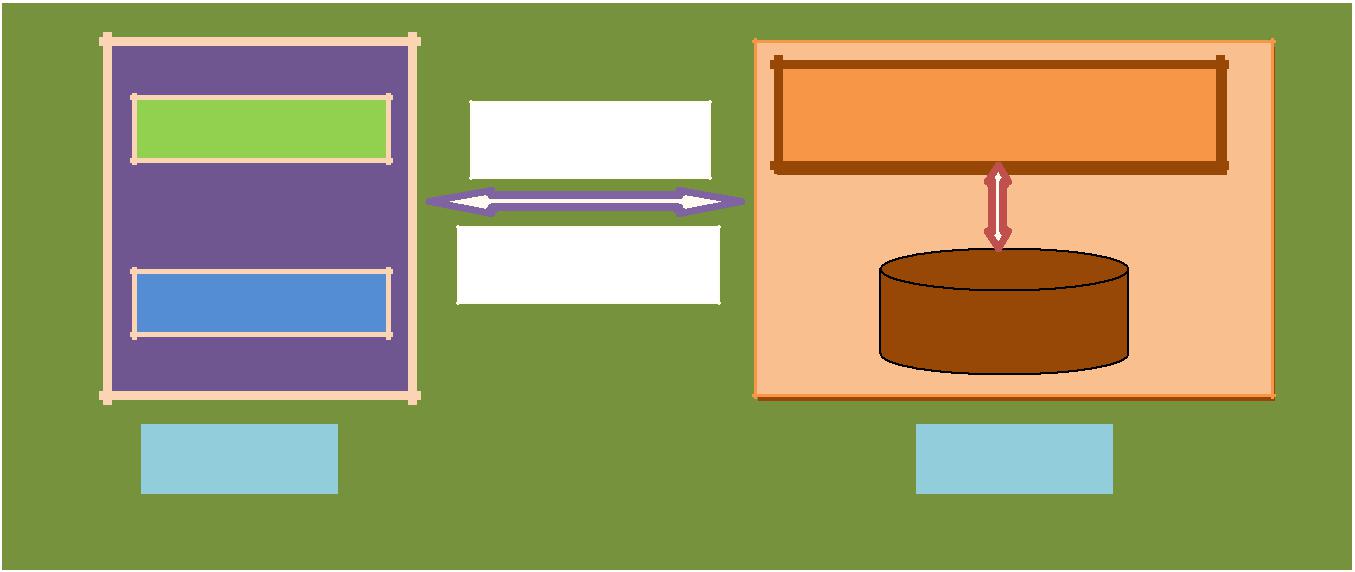
The operations supported and allowed would be based on the patient type. For example, Administrator has the rights to add patient information and view patient details. He can also view testreport and price details of a test.

Whereas Patient has a right to Add, Remove and Clear all the testdetails from testreport.

***2.2 Functional Flow***

The functional flow of the messages across different application components is shown below.

Ex. - Web Application.



|  |  |  |
| --- | --- | --- |
| **Administrator** | **DES Application** |  |
| HTTP/GET |  |
|  | HTTP/Response |  |
| **Customer** | Database |  |
|  |  |
| Client GUI | Server |  |
|  | **Digital Emporium E-Store System** |  |
|  |  |  |
| ***2.3 Environment*** |  |  |

The system will be developed on any Windows OSmachine using J2EE, Hibernate and Spring.

* Intel hardware machine (PC P4-2.26 GHz, 512 MB RAM, 40 GB HDD)
* Server – Apache Tomcat 9
* Database – Oracle 18g
* JRE 8
* Spring Tool Suite



**FUNCTIONAL SPECIFICATION**

**3 Sub-system Details**

Medical Diagnostic Laboratory is defined, wherein all users need to login successfully before performing any of their respective operations.

Find below (section 3.1 & 3.2) tables that provides functionality descriptions for each type of patient / sub-system. Against each requirement, indicative data is listed in column ‘Data to include’. Further, suggested to add/modify more details wherever required with an approval from customer/faculty.

***3.1 Administrator***

The administrator as a user is defined to perform below listed operations after successful login.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Objects | Operations | Data to include | Remarks |
| AD-001  To  AD-004 | Doctor | Add  View  Modify  Delete | doctorId, doctorFirstName,  doctorLastName,doctorPhoneNumber,doctorEmail,qualification |  |
| AD-005  To  AD-0010 | Patient | Add  View  Modify  Delete | patientId,patientFirstName,  patientLastName,age,gender,phoneNumber |  |
| AD – 0011  to  AD - 0013 | Technician | Add  View  Modify  Delete | technicianId,technicianFirstName,technicianLastName,technicianEmailId,technicianPhoneNumber |  |

***3.2 Patient***

The patient as a user is defined to perform below listed operations after successful login.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Objects | Operations | Data to include |  | Remarks | | | | |
| US-001 | Patient | Register  Login | patientId,patientFirstName,  patientLastName,age,gender,phoneNumber |  |  |  |  |  |  |
| US-002 | testreport | View | testreportId,billingPrice,result,patient,medicalTest,technician |  |  |  |  |  |  |
| US-003 | Checkout | ViewDetails  and Price | doctor,patientId and Total Price |  |  |  |  |  |  |

**FUNCTIONAL SPECIFICATION**

***3.3 Login | Logout***

**[Web Application - J2EE, Hibernate, Spring]**

* Go to Registration screen when you click on Register link
* Go to Success screen when you login successfully after entering valid username & password fetched from the database.
* Redirect back to same login screen if username & password are not matching
* Implement Session tracking for all logged in users before allowing access to application features. Anonymous users should be checked, unless explicitly mentioned.

**4 Data Organization**

This section explains the data storage requirements of the Product Order Entry System and **indicative** data description along with suggested table (database) structure. The following section explains few of the tables (fields) with description. However, in similar approach need to be considered for all other tables.

***4.1 Table: User\_Registration\_Details***

The user specific details such as username, email, phone etc. Authentication, and authorization / privileges should be kept in one or more tables, as necessary and applicable.

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| *UserID* | UserID is auto generated after registration and it is used as |
|  | LoginID. |
| *Username* | Username of the Customer |
| *Password* | User Password |
| *Email* | Customer Email Id |
| *Phone Number* | 10-digit contact number of users |
| *State* | State |
| *City* | City |
| *Pin* | Zipcode / Pincode of current location |

***4.2 Table: Product\_Details***

This table contains information related to a product

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| *Product Id* | Unique Product Id, Here Product Id will be Primary Key |
| *Product Name* | Name of the Product e.g. TV, AC etc. |
| *Product Price* | Price of the Product |



***4.3 Table: Product\_Cart\_Details***

This table contains information related to cart details

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| *CartId* | Unique Cart ID Auto Generated |
| *UserId* | UserId corresponding to logged in user |
| *Product Price* | Price of the Product |
| *Quantity* | Quantity of the product in e.g. 2 or 3 number of products |
| *Total Price* | Total price of the purchased products |
| *Status* | Product Availability Status, Example: Pending or Placed. |

***4.4 Table: Checkout\_Details***

This table contains information related to final checkout details

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| *CartId* | Unique Cart Id, this field must be foreign key |
| *UserId* | Registered User Id, this field should be foreign key |
| *Total Price* | Total Price of the purchased products |



**FUNCTIONAL SPECIFICATION**

1. **REST APIsto be Built**

Create following REST resources which are required in the application,

1. Creating **User** Entity: Create Spring Boot with Microservices Application with Spring Data JPA

**Technology stack:**

* Spring Boot
* Spring REST
* Spring Data JPA

Here will have multiple layers into the application:

1. Create an Entity: User
2. Create a UserRepository interface and will make use of Spring Data JPA
3. Will have findByUserName method
4. Add the User details
5. Create a UserService class and will expose all these services
6. Finally, create a UserRestController will have the following Uri’s:

|  |  |  |  |
| --- | --- | --- | --- |
| URI | METHODS | Description | Format |
| /users/userName | GET | Give a single user description searched based on username | JSON |
| /users/userid | GET | Give a single user description searched based on user id | String |
| /users | POST | Add the user details | JSON |
| /users | PUT | Update the user details | JSON |
| /users/userid | DELETE | Delete user by id | String |

1. Creating **Product** Entity:

Build a RESTful resource for **Product** manipulations, where CRUD operations to be carried out. Here will have multiple layers into the application:

1. Create an Entity: Product
2. Create a ProductRepository interface and will make use of Spring Data JPA
3. Will have findByProductName method
4. Add the Product details method
5. Will have deleteProductById method
6. Will have findAllProducts method
7. Create a ProductService class and will expose all these services
8. Finally, create a ProductRestController will have the following Uri’s:

|  |  |  |  |
| --- | --- | --- | --- |
| URI | METHODS | Description | Format |
| /products | GET | Get all the products | JSON |
| /products/productName | GET | Give a single product description searched based on product name | JSON |
| /products | POST | Add the product details | JSON |
| /products/productId | DELETE | Delete a Product based on product id | JSON |

1. Creating **ProductCart** Entity:

Build a RESTful resource for **ProductCart** manipulations, where following operations to be carried out. Here will have multiple layers into the application:

1. Create an Entity: ProductCart
2. Create a ProductCartRepository interface and will make use of Spring Data JPA
   1. Add the ProductCart details
   2. Will have deleteProductById method to remove item with specific product Id from cart.
   3. Will have deleteAllProducts method to remove all items from the cart.
   4. Will have changeProductStatus to change the status from available to sold-out and vice-versa.
3. Create a ProductCartService class and will expose all these services
4. Finally, create a ProductCartRestController will have the following Uri’s:

|  |  |  |  |
| --- | --- | --- | --- |
| URI | METHODS | Description | Format |
| /productCart | POST | Add the product details with updated status. | JSON |
| /productCart/productId | DELETE | Delete a Product based on product id | JSON |
| /productCart | DELETE | Delete all the products from the cart | JSON |
| /productCart/changeStatus | UPDATE/ PATCH | Change the status of the cart to pending or placed | JSON |

1. Creating **Checkout** Entity:

Build a RESTful resource for **Checkout** manipulations, where following operations to be carried out. Here will have multiple layers into the application:

1. Create an Entity: Checkout
2. Create a CheckoutRepository interface and will make use of Spring Data JPA
   1. Add the checkout details
3. Create a CheckoutService class and will expose all these services
4. Finally, create a CheckoutRestController will have the following Uri’s:

|  |  |  |  |
| --- | --- | --- | --- |
| URI | METHODS | Description | Format |
| /checkout | POST | Add the user details with total price. | JSON |

**FUNCTIONAL SPECIFICATION**

1. **Assumptions**

* User Interface: The type of client interface (front-end) to be supported - Angular based
* The administrator can add and remove products into the database on a weekly basis.
* You must not allow user to add same product twice.
* When you add product into cart the No. of Products selected will be incremented.
* If you remove the product from the cart, the counter will be decremented.
* The clear will remove all the products so that the No. of products will be zero
* The total amount will be calculated based on the product, accordingly, change the product counter & total amount.

1. **General Expectations**

* Participants must create the **Class Diagram, Sequence Diagram and ER Diagram**.
* Participants must do **Unit testing and Functional Testing using POSTMAN tool**
* Integration of Angular and Spring Boot with Microservices should be done, referring**Project2 -Frond End Development Project.**
  + The server should be a concurrent server servicing multiple client.
  + Database can be implemented using Oracle 11g or above.
  + To begin with, the application should support at least 1 admin and 2 customers.
  + Compilation and Build should be done using Eclipse IDE or STS
  + Source-code and all documents must be maintained (checked-in) in configuration management system (subversion)
  + Coding standards (for Java) should be followed

NOTE:

1. **Validation of user Data**

* Spring MVC using JSR-303 annotations
* AJAX validation without forcing the page to reload (Wherever applicable)
* JavaScript validation (if necessary)

1. **UI Design – (for Web Application) Use DIV/CSS or Semantic Elements to control the style and layout**
2. **Create at least one SQL DML-statement inside PL/SQL blocks**

**8 Acceptance Criteria**

All P1 requirements must be mandatorily implemented

**9 Traceability to Requirements**

Appropriate requirements from RS and FS are mapped here.

**Document Reference ID & Description: (Doc ID from which this document is derived)**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Reference document: RS** | **Current document: FS** |
|  | **Requirement/Feature (Section ID/Name)** | **Location (Section ID/Name)** |

1.

2.

**10 Acronyms and Glossary**

Acronym and glossary for this document mentioned in the below table.

|  |  |
| --- | --- |
| **Abbreviation** | **Remark** |
| DES | Digital Emporium E-Store System |
| RS | Requirement Specification |
| FS | Functional Specification |

1. Validations should be performed at all levels of application appropriately.