

**Name: Muhammad Ryan Marcellino**

## **Mini Project of Backend Server Development**

### **Requirement of Project:**

#### Milestone 1

- A merchant service should be written with Node.js and using MySQL DB as a persistent storage
- A merchant service will expose several APIs with these functions
  - A merchant could register itself/create an account in the merchant service
  - A merchant could remove its data/delete its account in the merchant service
  - A merchant could add products in the merchant service
  - A merchant could delete a product in the merchant service
  - A merchant could update a product in the merchant service
  - A merchant could get the list of its products from the merchant service
  - A validation is needed to validate the data based on their respective format
  - Merchant information that a merchant service needed is
    - id
    - name (required, min: 3, max 50)
    - email (required, email, min: 10)
    - password (required, min: 6)
    - address (required)
    - phone\_number (required, numeric)
  - Product information that is needed is
    - id
    - merchant\_id
    - name (required, min: 3, max: 50)
    - quantity (required, min: 1, numeric)
    - price (required, min: 10000, numeric)
- Commit the working code to git

#### Milestone 2

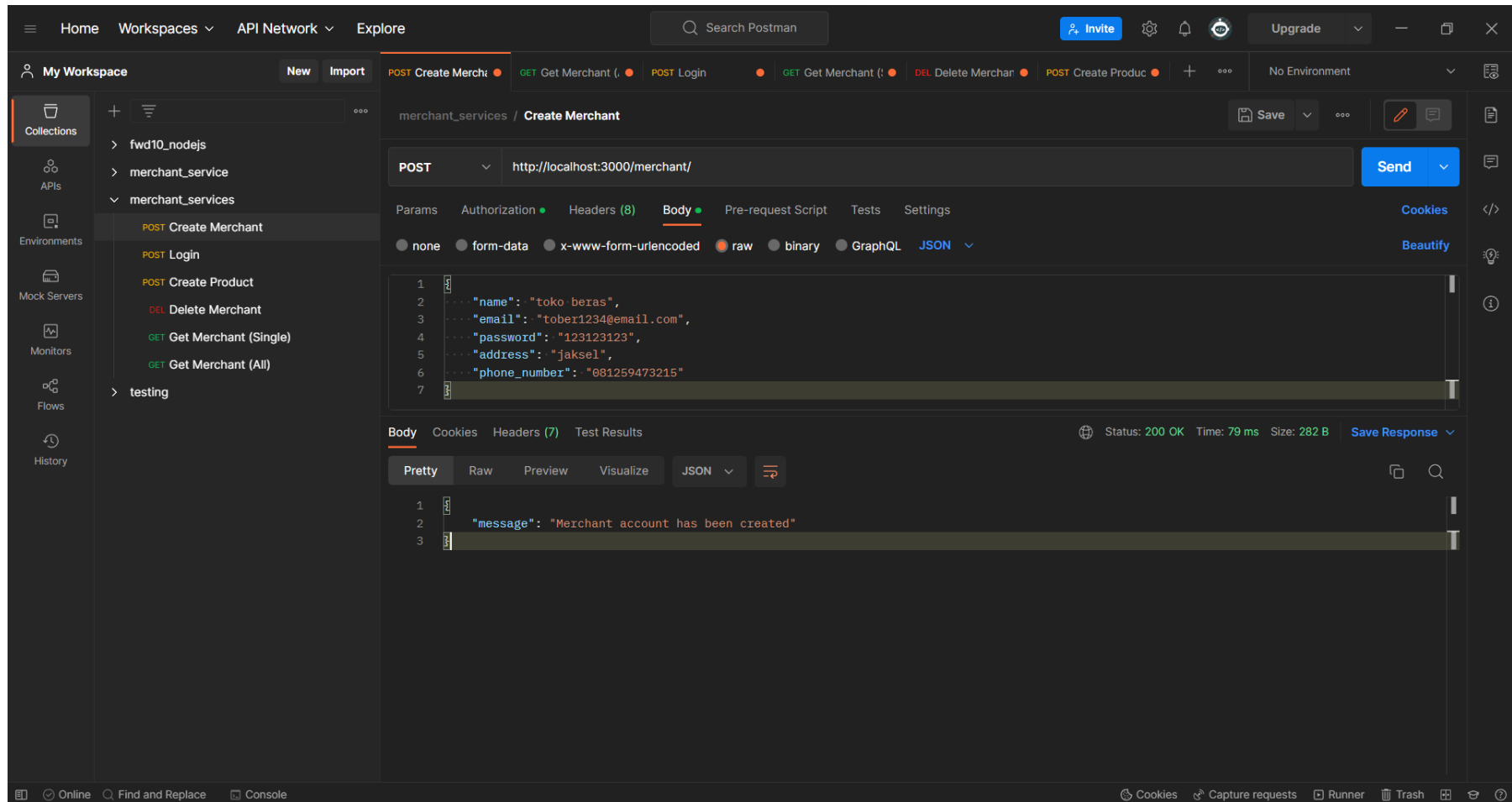
- Add a [/login](#) api to authenticate user
- Authentication using Basic Auth (username and password)
- Commit the working code to git

#### Milestone 3

- Add JWT authorization as [access\\_token](#) to access resources after user logged in
- JWT is passed in Auth header as Bearer token
- Commit the working code to git

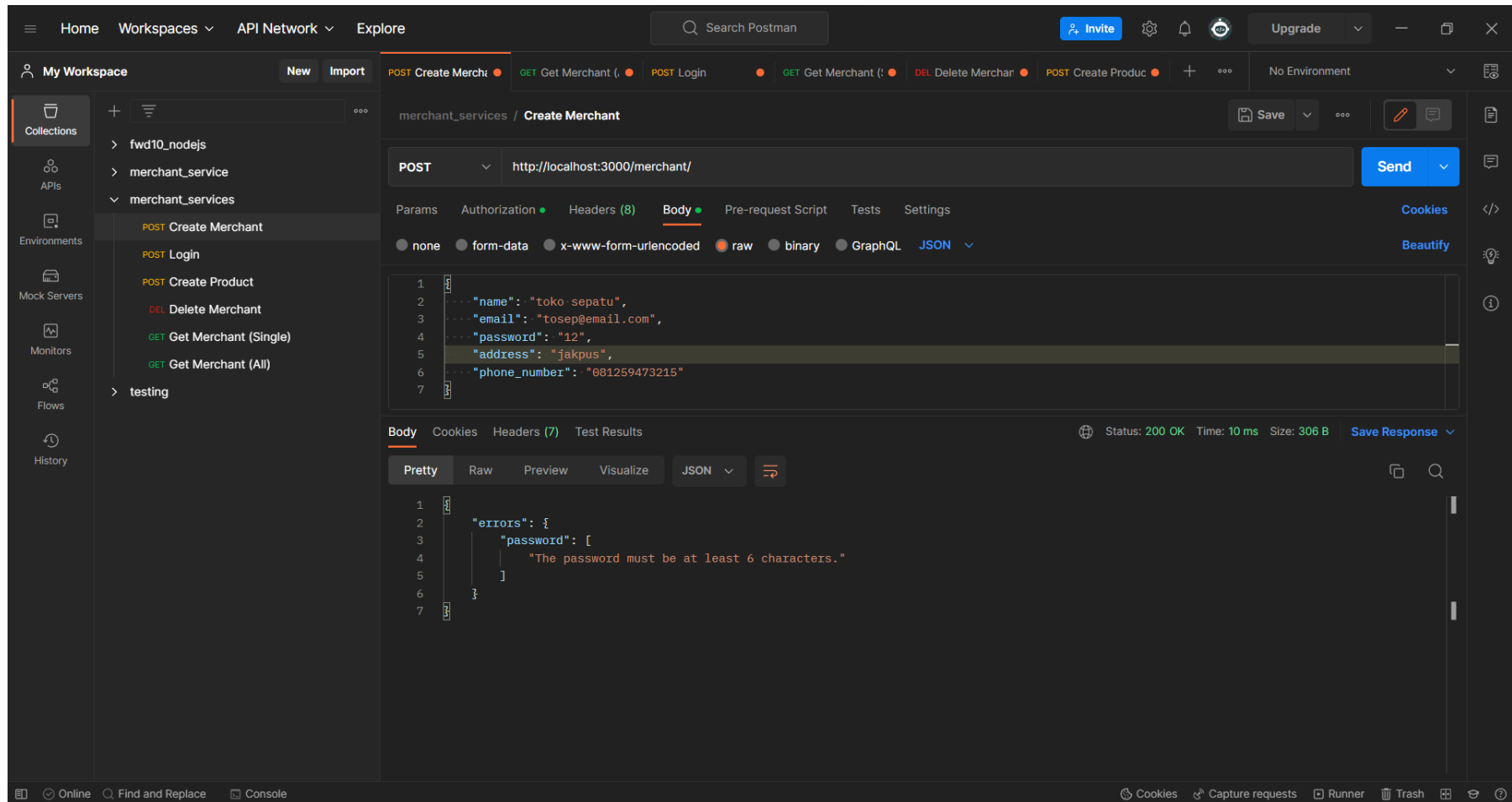
# 1) Screenshot: Implementation of A merchant could register itself/create an account in the merchant service

Successfully to create or register merchant account



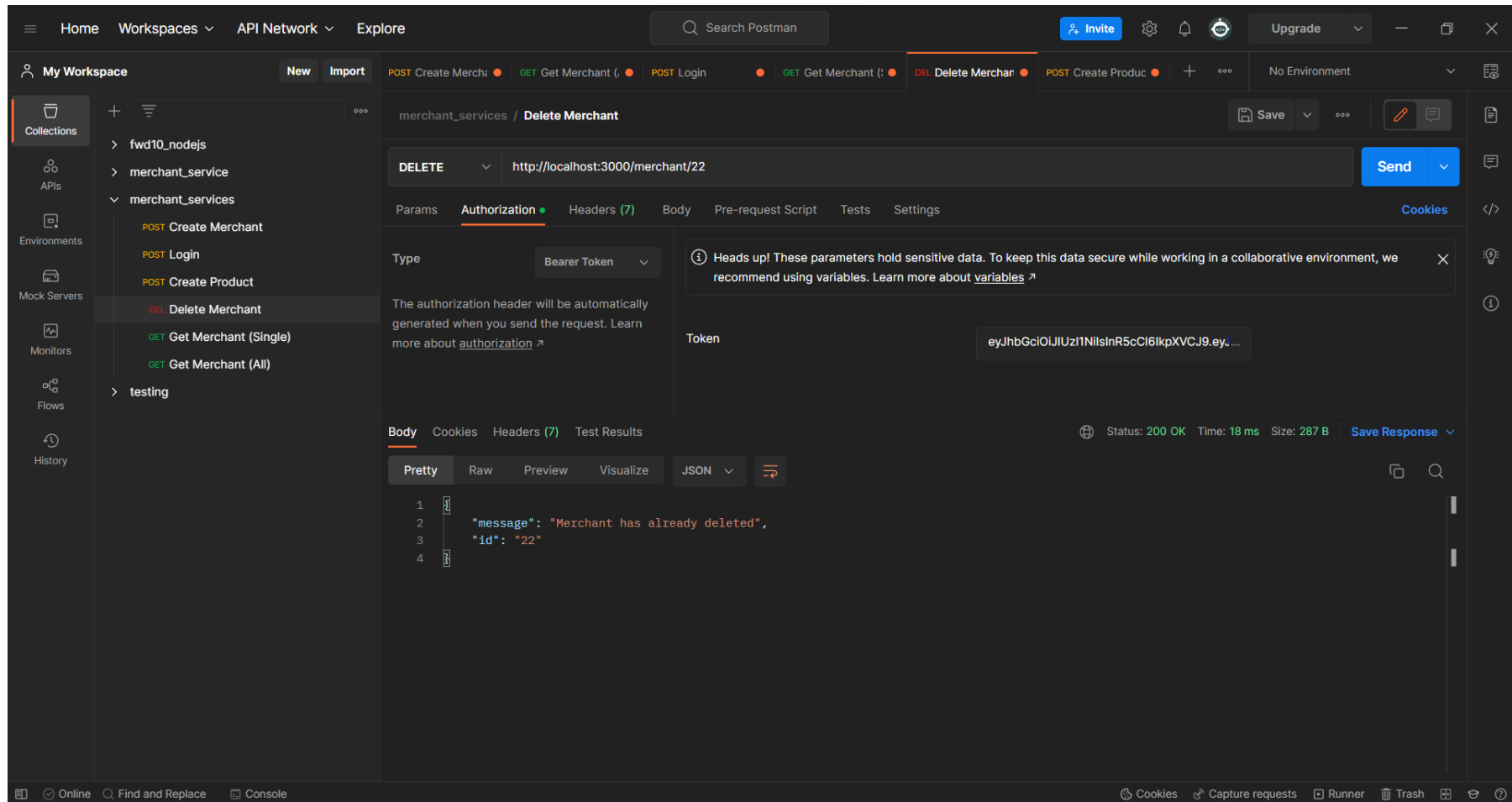
## 2) Screenshot: Implementation of A merchant could register itself/create an account in the merchant service and merchant validation

It's failed to register or create account because password character is less than 6 characters



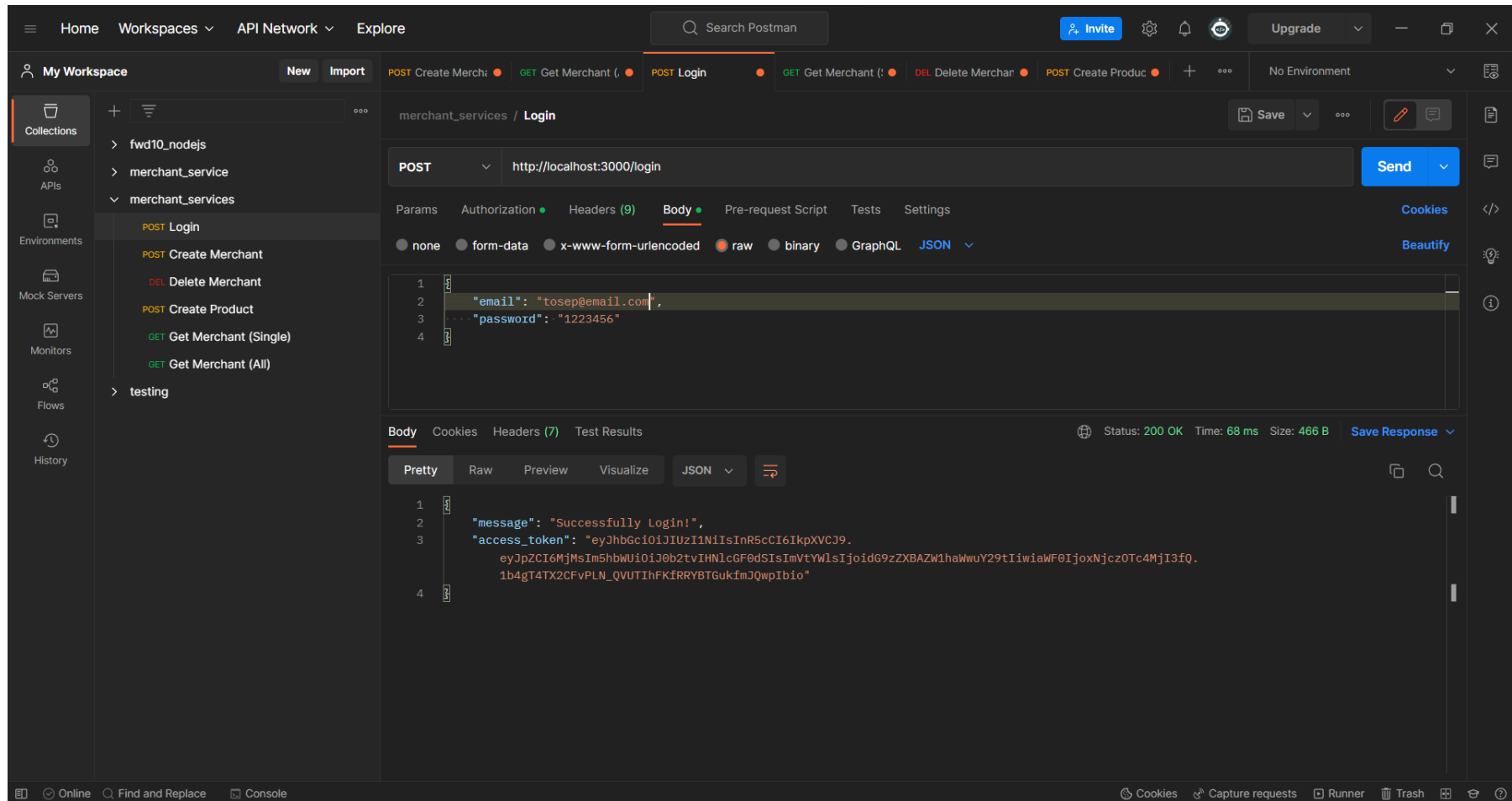
### 3) Screenshot: Implementation of *A merchant could remove its data/delete its account in the merchant service*

Successfully to delete merchant account



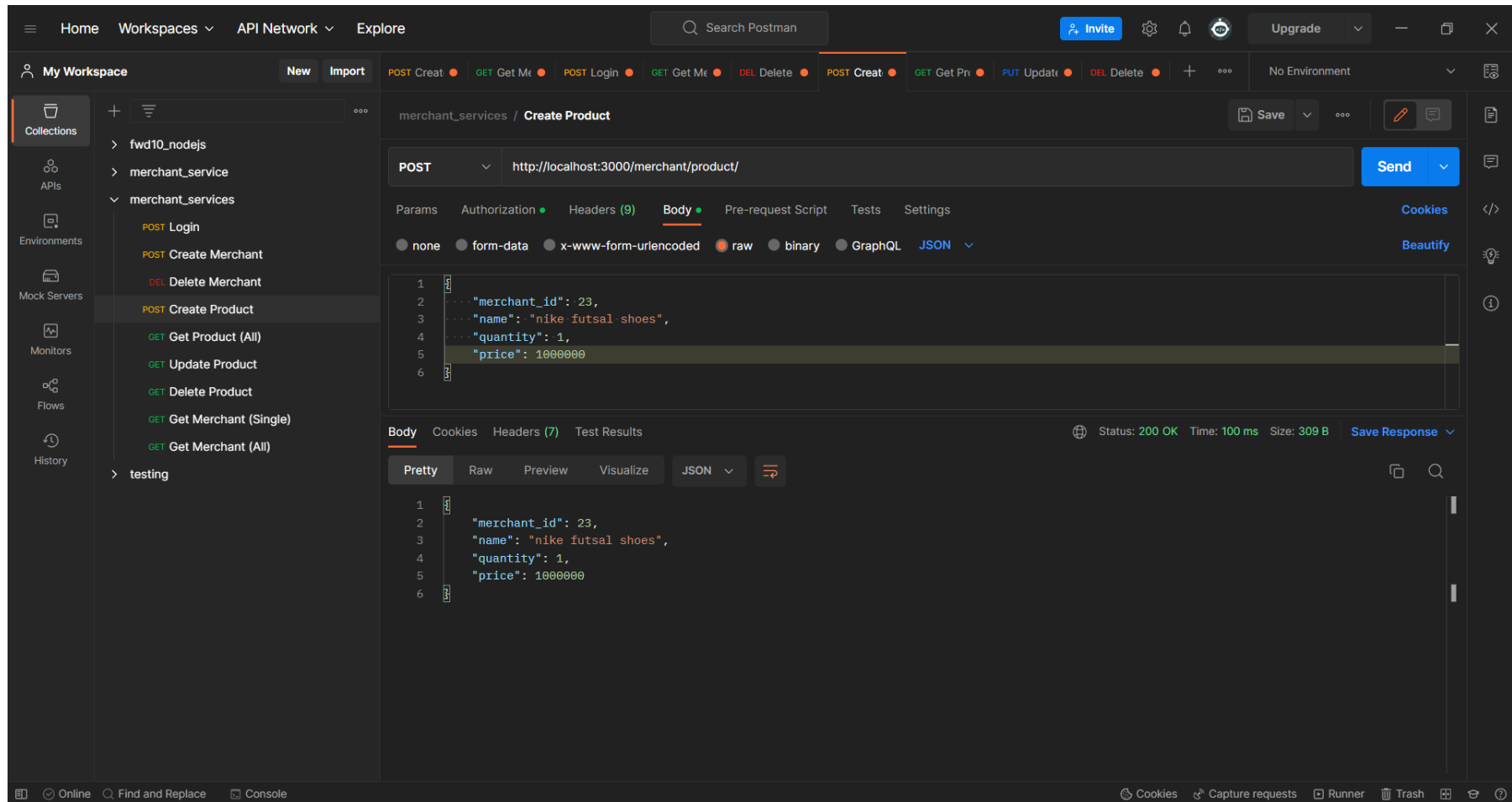
**4) Screenshot:** Implementation of *Add a /login api to authenticate user, Authentication using Basic Auth (email and password), Add JWT authorization as access\_token & JWT is passed in Auth header as Bearer token*

Successfully login to authenticate user with its email and password



## 5) Screenshot: Implementation of *A merchant could add products in the merchant service*

Successfully to create or add product in merchant account



## 6) Screenshot: Implementation of *A merchant could add products in the merchant service and product validation*

It's failed to create or add product in merchant account because the price data is less than 1000

The screenshot displays the Postman interface with a workspace named 'My Workspace'. The left sidebar shows a collection of API endpoints under 'merchant\_services', including 'POST Create Product', which is currently selected. The main panel shows the details of this POST request to 'http://localhost:3000/merchant/product/'. The request body is a JSON object: 

```
{  "merchant_id": 23,  "name": "nike futsal shoes",  "quantity": 1,  "price": 1}
```

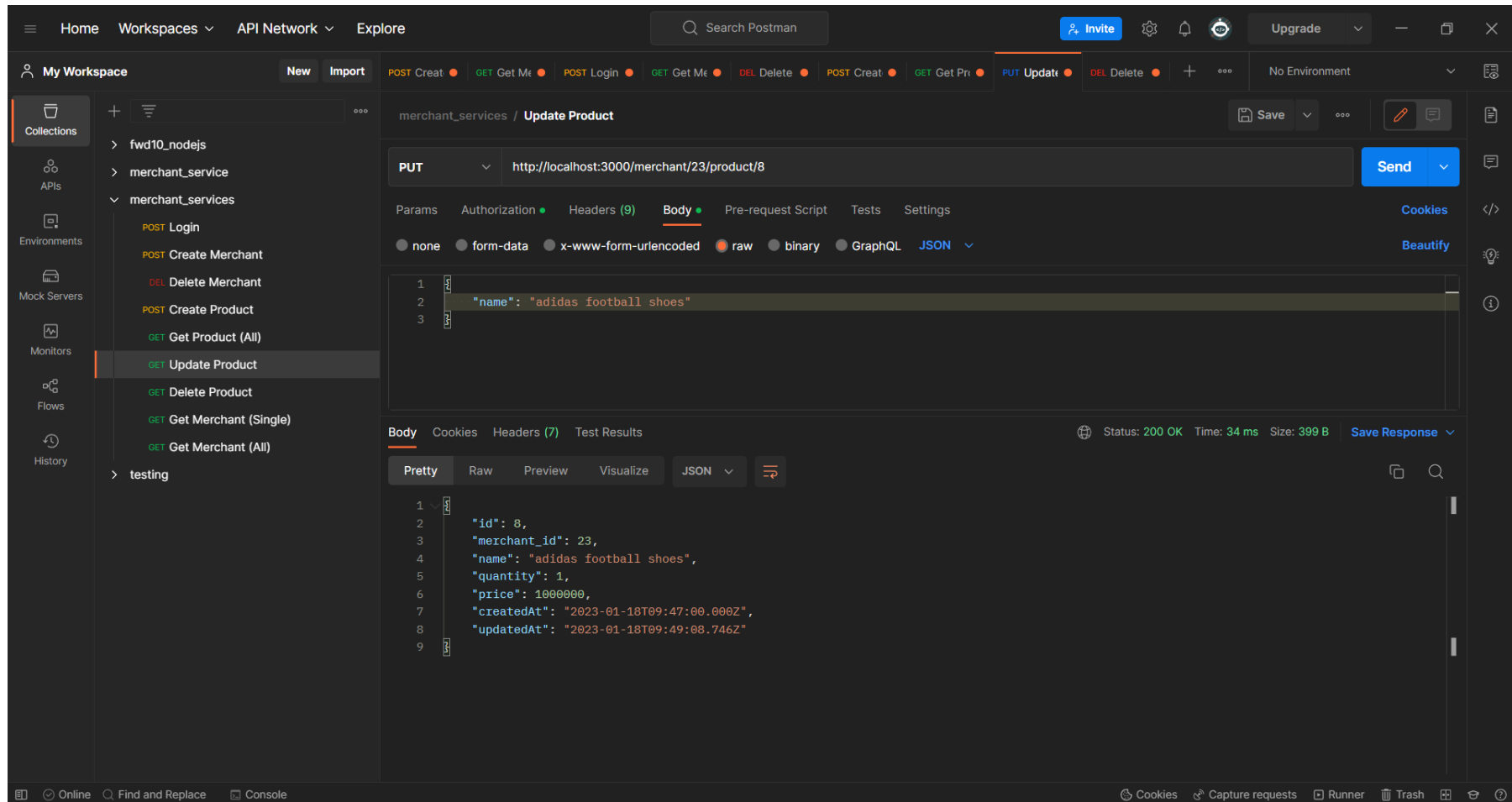
. The response status is '200 OK' with a time of '28 ms' and size of '292 B'. The response body, shown in 'Pretty' format, contains an error message: 

```
{  "errors": {    "price": [      "The price must be at least 1000."    ]  }}
```

. The bottom status bar includes options like 'Online', 'Find and Replace', 'Console', 'Cookies', 'Capture requests', 'Runner', 'Trash', and a help icon.

## 7) Screenshot: Implementation of *A merchant could update a product in the merchant service*

Successfully to update the name of product from “nike futsal shoes” to “adidas football shoes”





## 8) Screenshot: Implementation of A merchant could get the list of its products from the merchant service

Successfully to get all product in merchant account

The screenshot displays the Postman application interface. The left sidebar shows the 'My Workspace' tree with a collection named 'merchant\_services' containing several endpoints. The 'GET Get Product (All)' endpoint is selected. The main panel shows the request details for this endpoint, including the URL 'http://localhost:3000/merchant/23/product/'. The 'Send' button is visible. Below the request details, the 'Body' tab is active, showing the response in 'Pretty' format. The response is a JSON object with the following structure:

```
{
  "id": 8,
  "merchant_id": 23,
  "name": "adidas football shoes",
  "quantity": 1,
  "price": 1000000,
  "createdAt": "2023-01-18T09:47:00.000Z",
  "updatedAt": "2023-01-18T09:49:08.000Z"
}
```

The status bar at the bottom indicates a successful response with status 200 OK, a time of 17 ms, and a size of 401 B. The bottom right corner shows various utility icons like Cookies, Capture requests, Runner, Trash, and Help.

## 9) Screenshot: Implementation of *A merchant could delete a product in the merchant service*

Successfully to remove the specific product in merchant account

