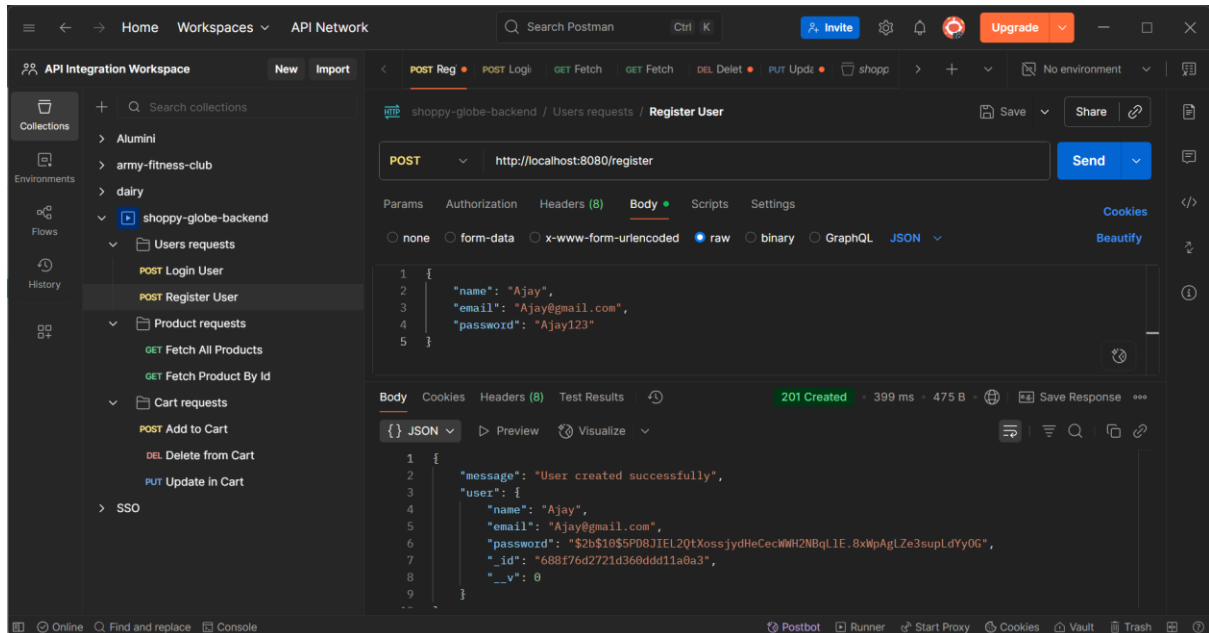


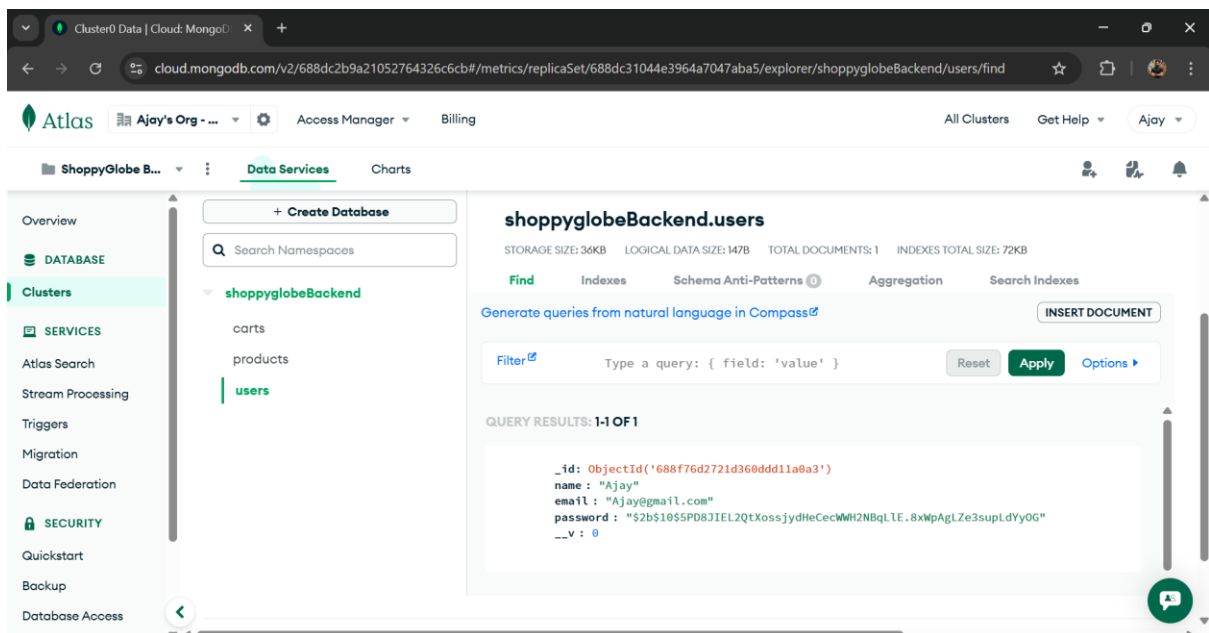
ShoppY Globe Backend - AJAY

Github Repo URL - <https://github.com/Ajay6433/shoppyglobe-backend>

POST_Register_Success_API_Postman



POST_Register_Success_MongoDb_Atlas_Collection



POST_Login_Success_API_JWT_Token

The screenshot shows a Postman interface for an API workspace named 'API Integration Workspace'. The selected request is a POST request to 'http://localhost:8080/login' under the collection 'shoppy-globe-backend / Users requests / Login User'. The request body is raw JSON with the following content:

```
1 {
2   "email": "Ajay@gmail.com",
3   "password": "Ajay123"
4 }
```

The response is a 200 OK status with a response time of 265 ms and a body size of 480 B. The response body is JSON, showing a successful login message and a JWT token:

```
1 {
2   "message": "Login successful",
3   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZiZCI6IjY4OGY3NmQyZDM2MGRKZDExYTBiMyIsIm1ldCI6MTc1NDIzMCJyZWwiZm9uZHMjM2MzIzIj0. f9c1AynNgP8ZpUAKWdiYniU8e1LKLZJXOZnHSuXG0gQ"
4 }
```

GET_Products_API

The screenshot shows a Postman interface for the same API workspace. The selected request is a GET request to 'http://localhost:8080/products' under the collection 'shoppy-globe-backend / Product requests / Fetch All Products'. The request headers are visible, and the response is a 200 OK status with a response time of 111 ms and a body size of 8.13 KB. The response body is JSON, showing a successful message and a list of products:

```
1 {
2   "message": "Products fetched successfully",
3   "products": [
4     {
5       "_id": "688f2d34d32652cf93e1b80b",
6       "title": "Essence Mascara Lash Princess",
7       "price": 9.99,
8       "description": "The Essence Mascara Lash Princess is a popular mascara known for its volumizing and lengthening effects. Achieve dramatic lashes with this long-lasting and"
9     }
10  ]
11 }
```

GET_Product_By_ID_API

The screenshot shows the Postman API client interface. On the left, the 'API Integration Workspace' sidebar lists collections, environments, flows, and history. The 'Fetch Product By Id' endpoint is selected under the 'shoppy-globe-backend' environment. The main panel displays the request details for a GET request to `http://localhost:8080/product/688f2d34d32052cf93e1b80d`. The 'Headers' tab is active, showing 6 hidden headers. The 'Body' tab shows a JSON response with a 200 OK status, 84 ms response time, and 644 B body size. The response body is as follows:

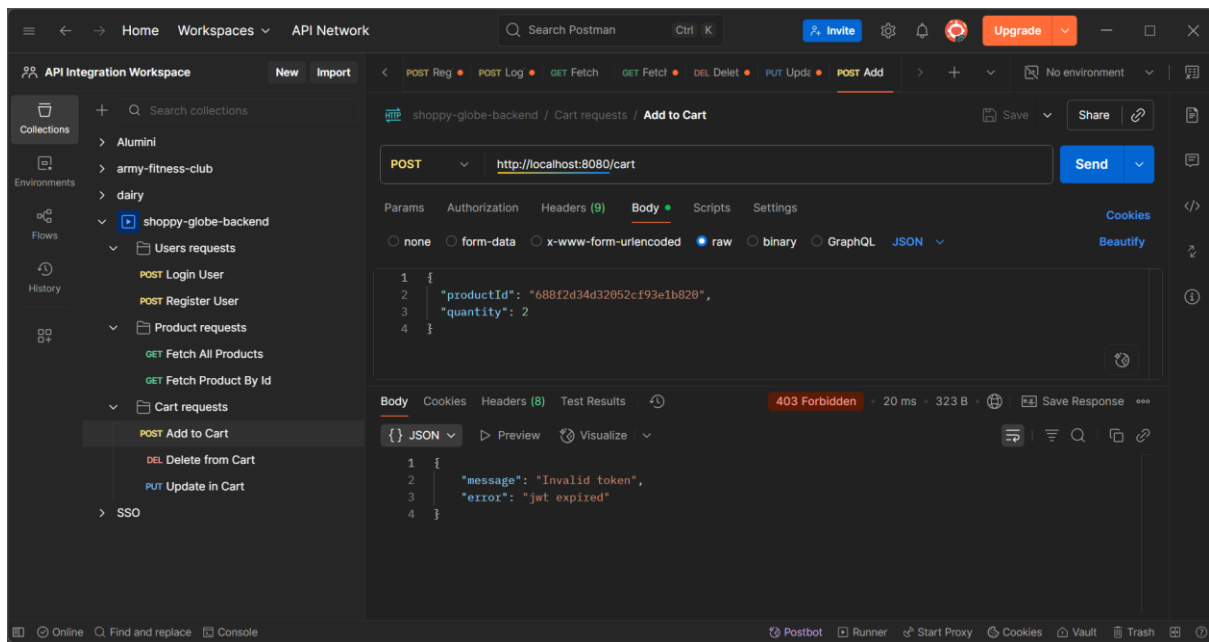
```
1 {
2   "message": "Product fetched successfully",
3   "product": {
4     "_id": "688f2d34d32052cf93e1b80d",
5     "title": "Powder Canister",
6     "price": 14.99,
7     "description": "The Powder Canister is a finely milled setting powder designed to set makeup and control shine. With a lightweight and translucent formula, it provides a smooth and matte finish.",
8   }
9 }
```

MongoDB_Atlas_Products_Collection

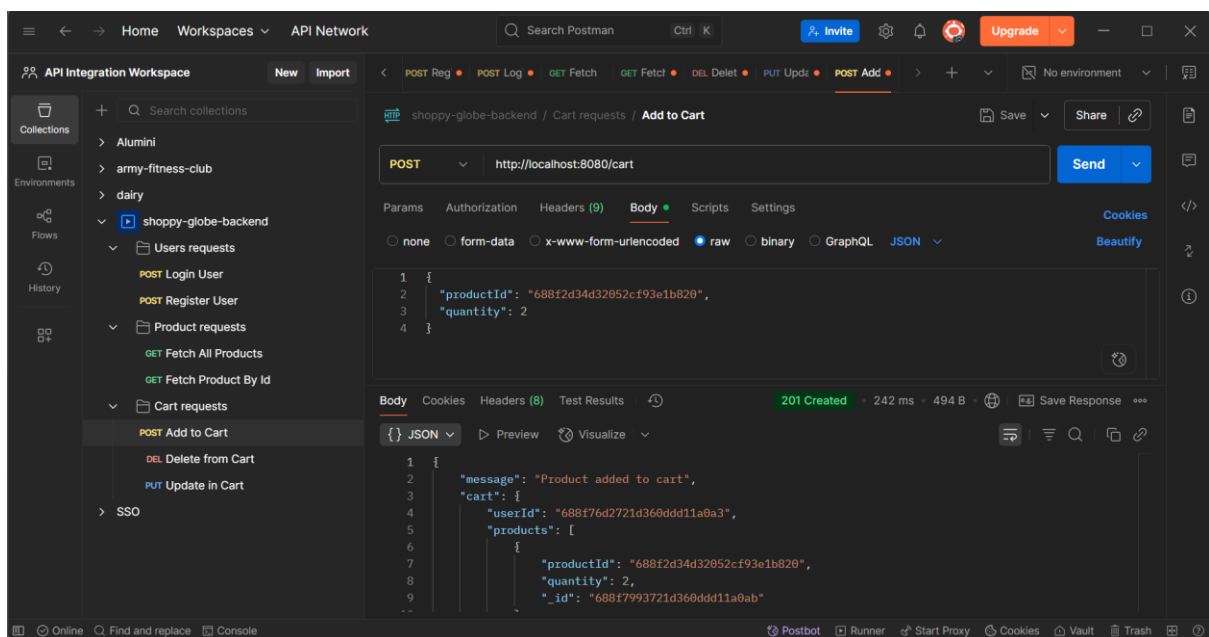
The screenshot shows the MongoDB Atlas web interface. The 'Data Services' tab is active, displaying the 'shoppyglobeBackend.products' collection. The collection statistics show a storage size of 44KB, logical data size of 765KB, 30 total documents, and 20KB total index size. The 'Find' tab is selected, showing a query filter of `{ field: 'value' }`. The query results are displayed in a JSON format, showing 1-20 of many results. The first result is as follows:

```
{
  "_id": ObjectId('688f2d34d32052cf93e1b80b'),
  "title": "Essence Mascara Lash Princess",
  "price": 9.99,
  "description": "The Essence Mascara Lash Princess is a popular mascara known for its volume and length.",
  "stock": 99,
  "category": "beauty",
  "rating": 2.56,
  "__v": 0
}
```

POST_JWT_Expired_Token



POST_Add_To_Cart_API_Success



PUT_Update_Item_Success

This screenshot shows a successful PUT request in Postman. The request is sent to `http://localhost:8080/cart` with a JSON body containing product and quantity information. The response is a 200 OK status with a detailed JSON payload.

Request:

```
PUT http://localhost:8080/cart
{
  "productId": "688f2d34d32052cf93e1b820",
  "quantity": 5
}
```

Response:

```
200 OK
{
  "message": "Cart updated successfully",
  "cart": {
    "_id": "688f7993721d360ddd11a0aa",
    "userId": "688f76d2721d360ddd11a0a3",
    "products": [
      {
        "productId": "688f2d34d32052cf93e1b820",
        "quantity": 5,
        "_id": "688f7993721d360ddd11a0ab"
      }
    ]
  }
}
```

PUT_Update_Item_In_Cart_With_Wrong_Product_ID

This screenshot shows a failed PUT request in Postman. The request is sent to `http://localhost:8080/cart` with a JSON body containing a product ID that does not exist in the cart. The response is a 404 Not Found status with a message indicating the product was not found.

Request:

```
PUT http://localhost:8080/cart
{
  "productId": "688f2d34d32252cf93e1b820",
  "quantity": 5
}
```

Response:

```
404 Not Found
{
  "message": "Product not found in cart"
}
```

DELETE_Delete_Item_In_Cart_Success

The screenshot displays the Postman API client interface. The left sidebar shows the 'API Integration Workspace' with a collection tree. The main panel shows a DELETE request to 'http://localhost:8080/cart/688f2d34d32052cf93e1b820' with an 'Authorization' header. The response is a 200 OK status with a JSON body indicating successful deletion.

Request Details:

- Method: DELETE
- URL: `http://localhost:8080/cart/688f2d34d32052cf93e1b820`
- Headers (7):

Key	Value	Description
Authorization	JWT eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXV...	

Response Details:

- Status: 200 OK
- Time: 186 ms
- Size: 407 B
- Body (JSON):

```
1 {
2   "message": "Product removed from cart",
3   "cart": {
4     "_id": "688f7993721d366ddd11a9aa",
5     "userId": "688f76d2721d366ddd11a9a3",
6     "products": [],
7     "__v": 1
8   }
9 }
```