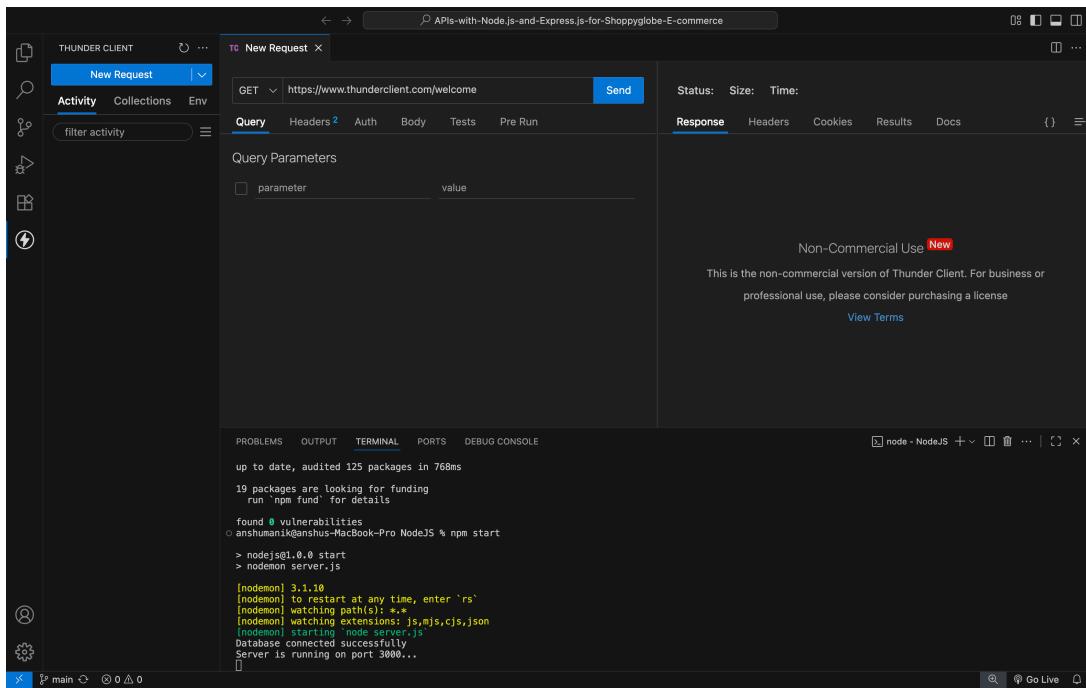


API Testing Document

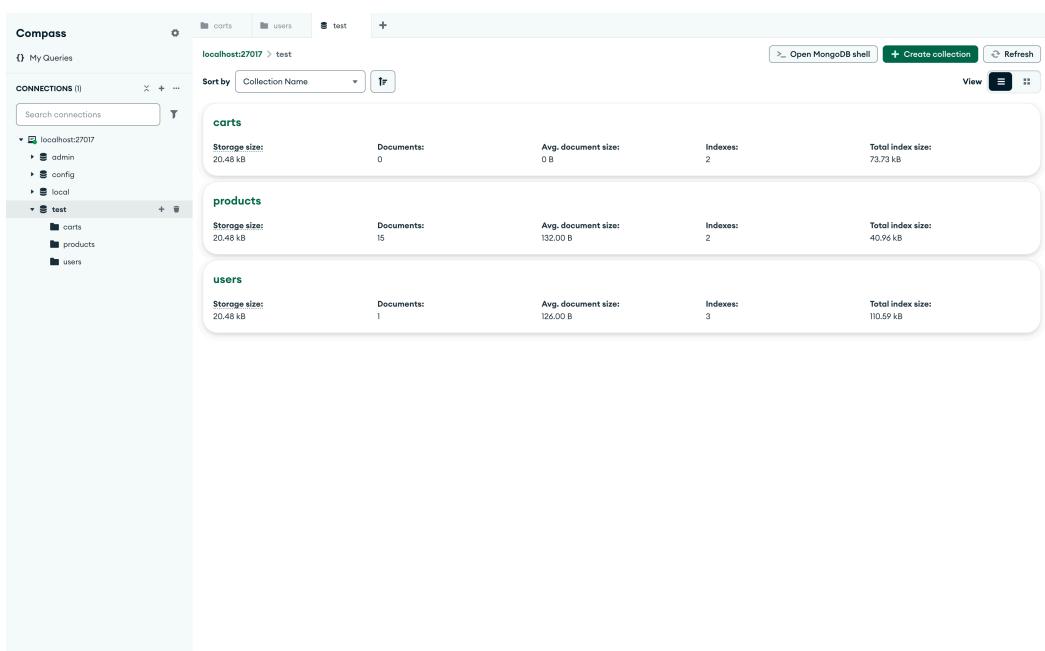
Project: Build APIs with Node.js and Express.js for Shoppyglobe E-commerce

Following are the output screenshots for below test cases:

1. Establish Connection to server :



2. Database at server startup:



3. Product database at server startup:

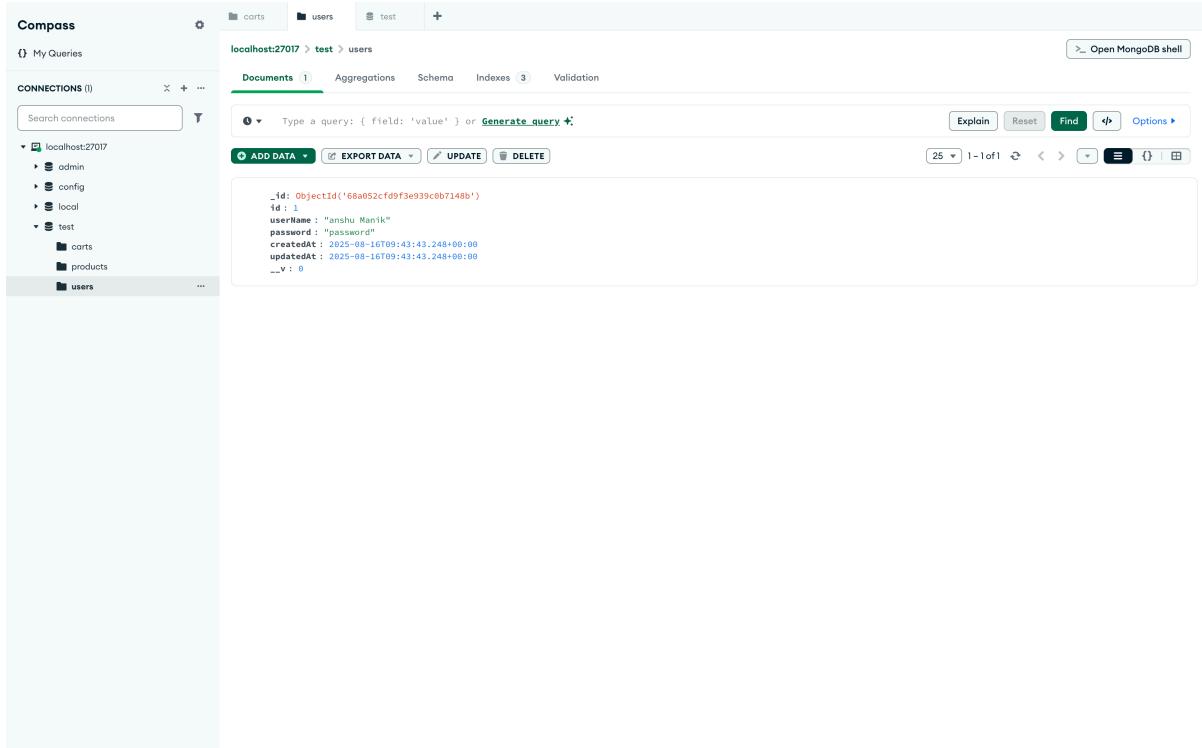
The screenshot shows the Compass MongoDB interface. The left sidebar shows connections to localhost:27017, including admin, config, local, test, carts, products, and users. The main area is titled "products" under "localhost:27017 > test > products". It displays 15 documents. Each document is represented by a card with the following fields:

- Document 1:** _id: ObjectId('68a02d0e4ef388b074a3eabb'), id: 1, name: "Wireless Mouse", price: 25.99, description: "Ergonomic wireless mouse", stockQuantity: 150
- Document 2:** _id: ObjectId('68a02d0e4ef388b074a3eabc'), id: 2, name: "Mechanical Keyboard", price: 79.99, description: "RGB backlit mechanical keyboard", stockQuantity: 85
- Document 3:** _id: ObjectId('68a02d0e4ef388b074a3eabd'), id: 3, name: "HD Monitor", price: 199.99, description: "24-inch Full HD monitor", stockQuantity: 40
- Document 4:** _id: ObjectId('68a02d0e4ef388b074a3eabf'), id: 4, name: "USB-C Hub", price: 49.99, description: "6-in-1 USB-C hub", stockQuantity: 120
- Document 5:** _id: ObjectId('68a02d0e4ef388b074a3eabf'), id: 5, name: "Laptop Stand", price: 29.99, description: "Aluminum laptop stand", stockQuantity: 100
- Document 6:** _id: ObjectId('68a02d0e4ef388b074a3eac0'), id: 6, name: "Webcam 1080p", price: 59.99, description: "HD 1080p webcam with microphone"

4. Cart database at server startup:

The screenshot shows the Compass MongoDB interface. The left sidebar shows connections to localhost:27017, including admin, config, local, test, carts, products, and users. The main area is titled "carts" under "localhost:27017 > test > carts". It displays 0 documents. A message at the top states "This collection has no data. It only takes a few seconds to import data from a JSON or CSV file." A "Import data" button is visible at the bottom.

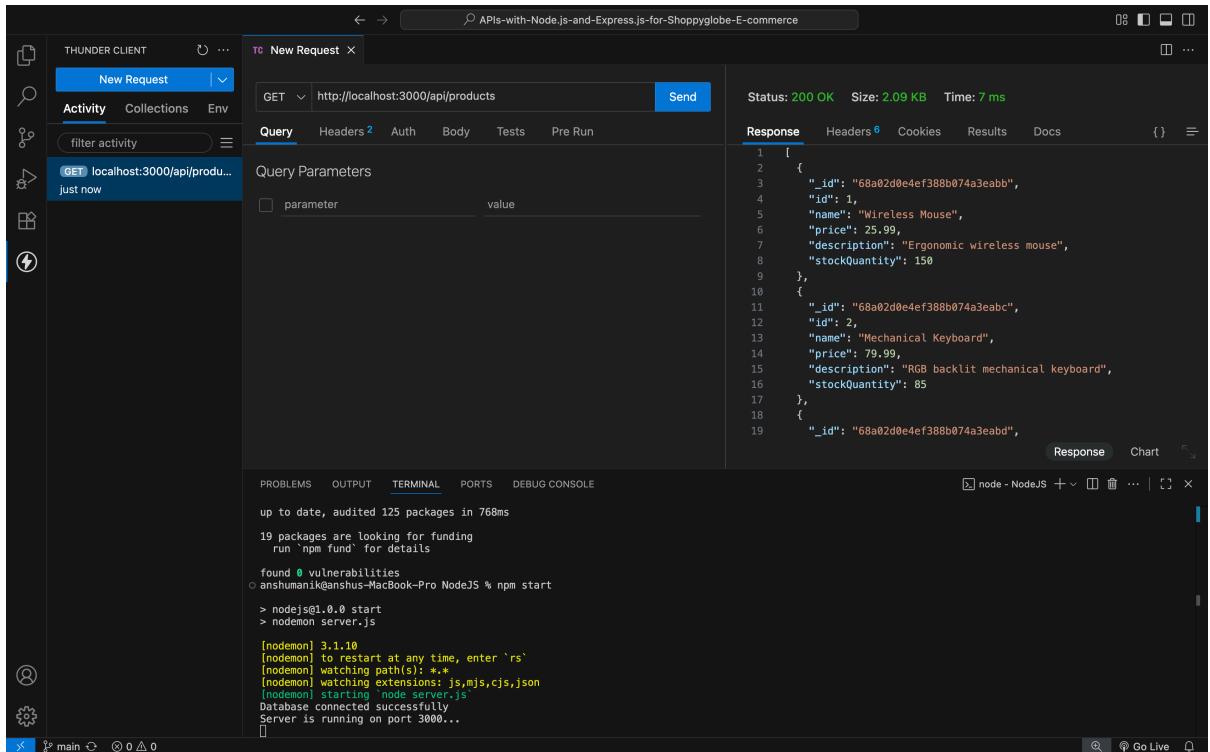
5. User database at server startup:



The screenshot shows the Compass MongoDB interface. On the left, there's a sidebar with 'CONNECTIONS (1)' containing a connection to 'localhost:27017' which includes 'admin', 'config', 'local', 'test', 'carts', 'products', and 'users'. The main area shows the 'test' database with the 'users' collection selected. A single document is listed with the following fields and values:

```
_id: ObjectId('68a052cdf9f3e939c0b7148b')
id: 1
userName: "anshu Manik"
password: "password"
createdAt: 2025-08-16T09:43:43.248+00:00
updatedAt: 2025-08-16T09:43:43.248+00:00
__v: 0
```

6. GET products API to get all products (no authentication):



The screenshot shows the Thunder Client interface. On the left, there's a sidebar with 'THUNDER CLIENT' and 'New Request'. The main area shows a 'New Request' dialog for a 'GET' request to 'http://localhost:3000/api/products'. The response status is '200 OK', size is '2.09 KB', and time is '7 ms'. The response body is a JSON array of two products:

```
[{"_id": "68a02d0e4ef388b074a3eabb", "id": 1, "name": "Wireless Mouse", "price": 25.99, "description": "Ergonomic wireless mouse", "stockQuantity": 150}, {"_id": "68a02d0e4ef388b074a3eabc", "id": 2, "name": "Mechanical Keyboard", "price": 79.99, "description": "RGB backlit mechanical keyboard", "stockQuantity": 85}, {"_id": "68a02d0e4ef388b074a3eabd"}]
```

Below the request details, there's a terminal window showing the Node.js environment and a DEBUG CONSOLE window.

7. GET products API to get product by id (no authentication):

The screenshot shows the Thunder Client interface. In the top bar, it says "APIs-with-Node.js-and-Express.js-for-Shoppyglobe-E-commerce". The main area has a "New Request" button and a dropdown menu. Below that, there are tabs for "Activity", "Collections", and "Env". A search bar contains "http://localhost:3000/api/products/1". The "Query" tab is selected. The "Send" button is highlighted in blue. To the right, the response status is "Status: 200 OK", size is "140 Bytes", and time is "11 ms". The "Response" tab is selected, showing the JSON response:

```
1  {
2    "_id": "68a02d0e4ef388b074a2eabb",
3    "id": 1,
4    "name": "Wireless Mouse",
5    "price": 25.99,
6    "description": "Ergonomic wireless mouse",
7    "stockQuantity": 150
8 }
```

Below the response, there's a terminal window showing Node.js startup logs:

```
up to date, audited 125 packages in 768ms
19 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
○ anshumanik@anshus-MacBook-Pro NodeJS % npm start
> nodejs@1.0.0 start
> nodemon server.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Database connected successfully
Server is running on port 3000...
[]
```

8. POST cart API to add item to cart (without user authentication):

The screenshot shows the Thunder Client interface. In the top bar, it says "APIs-with-Node.js-and-Express.js-for-Shoppyglobe-E-commerce". The main area has a "New Request" button and a dropdown menu. Below that, there are tabs for "Activity", "Collections", and "Env". A search bar contains "http://localhost:3000/api/cart". The "Headers" tab is selected. The "Send" button is highlighted in blue. To the right, the response status is "Status: 401 Unauthorized", size is "68 Bytes", and time is "4 ms". The "Response" tab is selected, showing the JSON response:

```
1  {
2    "message": "Unauthorised access. Access token missing or malformed"
3 }
```

Below the response, there's a terminal window showing Node.js startup logs:

```
up to date, audited 125 packages in 768ms
19 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
○ anshumanik@anshus-MacBook-Pro NodeJS % npm start
> nodejs@1.0.0 start
> nodemon server.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Database connected successfully
Server is running on port 3000...
[]
```

9. POST cart API to add item with user authentication and data in request body:

The screenshot shows the Thunder Client interface. A new request is being made to `http://localhost:3000/api/cart`. The Headers tab is selected, showing the following configuration:

- Accept: */*
- User-Agent: Thunder Client (https://www.thunderclient.net)
- Authorization: JWT eyJhbGciOiJIUzI1NiIsInR5cC16lk;

The Response tab shows the following JSON error message:

```
1 {
2   "message": "No data provided"
3 }
```

The Terminal tab at the bottom shows the Node.js server logs:

```
up to date, audited 125 packages in 768ms
19 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities
○ anshumanik@anshus-MacBook-Pro NodeJS % npm start
> nodejs@0.0.0 start
> nodemon server.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Database connected successfully
Server is running on port 3000...
```

The screenshot shows the Thunder Client interface. A new request is being made to `http://localhost:3000/api/cart`. The Body tab is selected, showing the following JSON content:

```
1 {
2   "id": 1,
3   "name": "Apple",
4   "quantity": 100,
5 }
```

The Response tab shows the following JSON response message:

```
1 {
2   "id": 1,
3   "name": "Apple",
4   "quantity": 100,
5   "_id": "68a0721d44a3e11a116e6725",
6   "createdAt": "2025-08-16T11:57:17.857Z",
7   "updatedAt": "2025-08-16T11:57:17.857Z",
8   "__v": 0
9 }
```

The Terminal tab at the bottom shows the Node.js server logs, identical to the previous screenshot.

The screenshot shows the Compass MongoDB interface. On the left, there's a sidebar titled 'Compass' with sections for 'My Queries', 'CONNECTIONS (1)', and a search bar. The main area shows a database structure with 'localhost:27017' selected. Under 'test', there are collections: 'carts', 'products', and 'users'. The 'carts' collection is expanded, showing one document:

```

_id: ObjectId('68a0721d4a3e11a116e6725')
id: 1
name: "Apple"
quantity: 100
createdAt: 2025-08-16T11:57:17.857+00:00
updatedAt: 2025-08-16T11:57:17.857+00:00
__v: 0

```

10. POST cart API to add item with user authentication and no data in request body:

The screenshot shows the Thunder Client interface. A new request is being made to `http://localhost:3000/api/cart` using a POST method. The 'Body' tab is selected, showing an empty JSON content field. The response status is **400 Bad Request**, with a size of 30 bytes and a time of 6 ms. The response body is:

```

{
  "message": "No data provided"
}

```

Below the request, the terminal shows the Node.js application logs:

```

up to date, audited 125 packages in 768ms
19 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities
anshumanik@anshus-MacBook-Pro NodeJS % npm start
> nodejs@1.0.0 start
> nodemon server.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Database connected successfully
Server is running on port 3000...

```

11. POST cart API to add item with user authentication and incorrect data in request body:

The screenshot shows the Thunder Client interface. A new request is being made to `http://localhost:3000/api/cart` using the `POST` method. The `Body` tab is selected, containing the following JSON content:

```
1 {
2   "id": "hauh",
3   "name": "apple",
4   "quantity": 100
5 }
```

The response status is `400 Bad Request`, with a message: `"message": "Invalid or missing 'id'"`.

The terminal pane shows the Node.js application starting and connecting to the database.

12. PUT cart API to update item with user authentication and quantity data in request body:

The screenshot shows the Thunder Client interface. A PUT request is being made to `http://localhost:3000/api/cart/1`. The `Body` tab is selected, containing the following JSON content:

```
1 {
2   "quantity": 10
3 }
```

The response status is `200 OK`, with a message: `"message": "Quantity updated successfully"`. The response body also includes the updated item details.

The terminal pane shows the Node.js application starting and connecting to the database.

The screenshot shows the Compass MongoDB interface. On the left, there's a sidebar titled 'Compass' with sections for 'My Queries', 'CONNECTIONS (1)', and a tree view of databases ('localhost:27017'), collections ('admin', 'config', 'local', 'test'), and sub-collections ('carts', 'products', 'users'). The main area shows a database structure with 'carts', 'users', and 'test' collections. Under 'test', there are 'carts', 'products', and 'users' collections. The 'carts' collection has one document listed:

```
_id: ObjectId('68a0721d44a3e11a116e6725')
_id: 1
name: "Apple"
quantity: 10
createdAt: 2025-08-16T11:57:17.857+00:00
updatedAt: 2025-08-16T12:01:38.470+00:00
__v: 0
```

13. PUT cart API to update item with user auth and no data in request body:

The screenshot shows the Thunder Client interface. On the left, there's a sidebar with 'New Request' selected, showing activity logs for a 'PUT localhost:3000/api/cart/1' request made just now. The main area shows a 'New Request' dialog for a 'PUT' method to 'http://localhost:3000/api/cart/1'. The 'Body' tab is selected, showing a JSON Content section with a single digit '1'. The response panel shows a status of '500 Internal Server Error', size of '86 Bytes', and time of '8 ms'. The response body is:

```
1 {
2   "success": false,
3   "message": "Cannot read properties of undefined (reading 'quantity'
4 )"
```

At the bottom, the terminal shows the Node.js application output:

```
up to date, audited 125 packages in 768ms
19 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities
anshumanik@anshus-MacBook-Pro NodeJS % npm start
> nodejs@1.0.0 start
> nodemon server.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting 'node server.js'
Database connected successfully
Server is running on port 3000...
```

14. PUT cart API with no user authentication:

The screenshot shows the Thunder Client interface. A failed PUT request is being made to `http://localhost:3000/api/cart/1`. The response status is **401 Unauthorized**, with a size of **68 Bytes** and a time of **3 ms**. The response body is a JSON object with a single key `message` containing the value `"Unauthorised access. Access token missing or malformed"`.

The terminal output shows the Node.js application starting and connecting to a database.

```
up to date, audited 125 packages in 768ms
19 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
o anshumanik@anshus-MacBook-Pro NodeJS % npm start
> nodejs@1.0.0 start
> nodemon server.js

[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting 'node server.js'
Database connected successfully
Server is running on port 3000...
```

15. DELETE cart API to delete cart item using id without user auth:

The screenshot shows the Thunder Client interface. A failed DELETE request is being made to `http://localhost:3000/api/cart/1`. The response status is **401 Unauthorized**, with a size of **68 Bytes** and a time of **5 ms**. The response body is a JSON object with a single key `message` containing the value `"Unauthorised access. Access token missing or malformed"`.

The terminal output shows the Node.js application starting and connecting to a database.

```
up to date, audited 125 packages in 768ms
19 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
o anshumanik@anshus-MacBook-Pro NodeJS % npm start
> nodejs@1.0.0 start
> nodemon server.js

[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting 'node server.js'
Database connected successfully
Server is running on port 3000...
```

16. DELETE cart API to delete cart item using id with user auth:

The screenshot shows the Thunder Client interface. In the top navigation bar, it says "THUNDER CLIENT" and "New Request". Below that, there's a "DELETE" button and the URL "http://localhost:3000/api/cart/1". The "Headers" tab is selected, showing the following headers:

Header	Value
Accept	/*
User-Agent	Thunder Client (https://www.thunderclient.com)
Authorization	JWT eyJhbGciOiJIUzI1NiIsInR5cCI6Ik
header	value

The "Response" tab shows the JSON response:

```
1  {
2      "message": "Product deleted successfully",
3      "deletedItem": {
4          "_id": "68a0721d44a3e11a116e6725",
5          "id": 1,
6          "name": "Apple",
7          "quantity": 10,
8          "createdAt": "2025-08-16T11:57:17.857Z",
9          "updatedAt": "2025-08-16T12:01:38.470Z",
10         "__v": 0
11     }
12 }
```

Below the main interface, there's a terminal window showing the command line output of the application:

```
up to date, audited 125 packages in 768ms
19 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities
anshumanik@anshus-MacBook-Pro NodeJS % npm start
> nodejs@1.0.0 start
> nodemon server.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Database connected successfully
Server is running on port 3000...
```

The screenshot shows the Compass interface. On the left, there's a sidebar with "Compass" and "My Queries". Below that is a "CONNECTIONS" section with a dropdown menu. The main area shows a database structure with "localhost:27017" and "test" selected. Under "test", there are three collections: "carts", "users", and "test". The "carts" collection is currently selected. At the top of the "carts" page, there are tabs for "Documents", "Aggregations", "Schema", "Indexes", and "Validation". Below the tabs, there's a search bar and a "Generate query" button. At the bottom of the page, there are buttons for "ADD DATA", "EXPORT DATA", "UPDATE", and "DELETE". A message states "This collection has no data".

17. DELETE cart API to delete cart item which does not exist in db with user auth:

The screenshot shows the Thunder Client interface. In the top navigation bar, it says "APIs-with-Node.js-and-Express.js-for-Shoppyglobe-E-commerce". A "New Request" button is highlighted. Below it, a "DELETE" method is selected, and the URL is set to "http://localhost:3000/api/cart/1". The "Headers" tab is active, showing the following headers:

Header	Value
Accept	/*
User-Agent	Thunder Client (https://www.thunderclient.net)
Authorization	JWT eyJhbGciOiJIUzI1NiIsInR5cCI6Ik
header	value

The "Response" tab shows the following JSON response:

```
1 {
2   "message": "Product with given ID does not exist!!!"
3 }
```

Below the request area, there is a "TERMINAL" tab displaying the Node.js server logs:

```
up to date, audited 125 packages in 768ms
19 packages are looking for funding
  run 'npm fund' for details
  found 0 vulnerabilities
  anshumanik@anshus-MacBook-Pro NodeJS % npm start
> nodejs@1.0.0 start
> nodemon server.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Database connected successfully
Server is running on port 3000...
```

The screenshot shows the MongoDB Compass interface. On the left, the "Connections" sidebar lists databases: "localhost:27017" (selected), "admin", "config", "local", and "test". Under "test", there are collections: "carts" (selected), "products", and "users". The main panel shows the "carts" collection in the "test" database. The "Documents" tab is active, showing a search bar and a "Generate query" button. Below the search bar are buttons for "ADD DATA", "EXPORT DATA", "UPDATE", and "DELETE". To the right are "Explain", "Reset", "Find", "Options", and pagination controls (25, 0 - 0 of 0). A message at the bottom states "This collection has no data".

18. POST user API to register new user (no auth required) with no input data in body:

The screenshot shows the Thunder Client interface. In the top navigation bar, it says "New Request" and "POST http://localhost:3000/api/user/register". The "Headers" tab is selected, showing the following configuration:

Header	Value
Accept	/*
User-Agent	Thunder Client (https://www.thunderclient.net)
Authorization	JWT eyJhbGciOiJIUzI1NlslslnR5cCl6lkp
header	value

The "Response" tab shows the following JSON response:

```
1 {  
2   "message": "user details not entered"  
3 }
```

Below the request and response panes, there is a terminal window titled "node - NodeJS" showing the server's logs:

```
up to date, audited 125 packages in 768ms  
19 packages are looking for funding  
  run 'npm fund' for details  
  
found 0 vulnerabilities  
anshumanik@anshus-MacBook-Pro NodeJS % npm start  
> nodejs@1.0.0 start  
> nodemon server.js  
  
[nodemon] 3.1.10  
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,mjs,cjs,json  
[nodemon] starting 'node server.js'  
Database connected successfully  
Server is running on port 3000...
```

19. POST user API to register new user (no auth required to access api) with input data in body and userid already present in db:

The screenshot shows the Thunder Client interface. A POST request is being made to `http://localhost:3000/api/user/register`. The request body contains the following JSON:

```
1 {
2   "id": 1,
3   "userName": "anshuManik",
4   "password": "password"
5 }
```

The response status is **409 Conflict**, size is **48 Bytes**, and time is **4 ms**. The response body is:

```
1 {
2   "message": "User with given ID already exists."
3 }
```

The terminal below shows the Node.js application output:

```
up to date, audited 125 packages in 768ms
19 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
c anshumanik@anshus-MacBook-Pro:~/NodeJS % npm start
> nodejs@1.0.0 start
> nodemon server.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Database connected successfully
Server is running on port 3000...
```

The screenshot shows the Compass MongoDB interface. A connection named `localhost:27017` is selected, and the `test` database is chosen. In the `users` collection, a single document is displayed:

```
_id: ObjectId("68a052cfdf3e939c0b7148b")
id: 1
userName: "anshu Manik"
password: "password"
createdAt: 2025-08-16T09:43:43.248+00:00
updatedAt: 2025-08-16T09:43:43.248+00:00
__v: 0
```

20. POST user API to register new user (no auth required to access api) with input data in body and userid not present in db:

The screenshot shows the Thunder Client interface. In the top navigation bar, it says "APIs-with-Node.js-and-Express.js-for-Shoppyglobe-E-commerce". A "New Request" dialog is open, showing a POST request to "http://localhost:3000/api/user/register". The "Body" tab is selected, displaying the JSON input:

```
1 {
2   "id": 2,
3   "userName": "internshala",
4   "password": "password"
5 }
```

The response tab shows the result of the POST request:

```
1 {
2   "message": "user created successfully.",
3   "user": {
4     "id": 2,
5     "userName": "internshala",
6     "password": "password",
7     "_id": "68a076cc44a3e11a116e6730",
8     "createdAt": "2025-08-16T12:17:16.556Z",
9     "updatedAt": "2025-08-16T12:17:16.556Z",
10    "__v": 0
11  }
12 }
```

Below the main interface, there's a terminal window showing Node.js startup logs:

```
up to date, audited 125 packages in 768ms
19 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities
anshumanik@anshus-MacBook-Pro ~ % npm start
> nodejs@1.0.0 start
> nodemon server.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Database connected successfully
Server is running on port 3000...
```

The screenshot shows the Compass MongoDB interface. On the left, the connection tree shows "localhost:27017" with "admin", "config", "local", and "test" databases expanded, and "users" selected under "test". The main area displays the "users" collection with the following documents:

```
_id: ObjectId("68a052cfdf3e939c0b7148b")
id: 1
userName: "anshu Manik"
password: "password"
createdAt: 2025-08-16T09:43:43.248+00:00
updatedAt: 2025-08-16T09:43:43.248+00:00
__v: 0
```

```
_id: ObjectId("68a076cc44a3e11a116e6730")
id: 2
userName: "internshala"
password: "password"
createdAt: 2025-08-16T12:17:16.556+00:00
updatedAt: 2025-08-16T12:17:16.556+00:00
__v: 0
```

21. POST user API to login user (no auth required to access login api) with input data in body:

The screenshot shows the Thunder Client interface. In the top navigation bar, it says "APIs-with-Node.js-and-Express.js-for-Shoppyglobe-E-commerce". A "New Request" button is highlighted. Below it, a "POST" request is being made to "http://localhost:3000/api/user/login". The "Body" tab is selected, showing a JSON payload:

```
1 {
2   "username": "internshala",
3   "password": "password"
4 }
```

The response status is "200 OK", size is "218 Bytes", and time is "4 ms". The response body contains a JSON object with "message" and "accessToken" fields. The "accessToken" value is a long string of characters.

At the bottom, there's a terminal window showing Node.js startup logs:

```
up to date, audited 125 packages in 768ms
19 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities
anshumanik@anshus-MacBook-Pro NodeJS % npm start
> nodejs@1.0.0 start
> nodemon server.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting 'node server.js'
Database connected successfully
Server is running on port 3000...
```

22. POST user API to login user (no auth required to access login api) with input data in body but user data not in db:

The screenshot shows the Thunder Client interface. A new request is being made to `http://localhost:3000/api/user/login`. The request method is POST, and the body contains the following JSON:

```
1 {
2   "userName": "trainings",
3   "password": "password"
4 }
```

The response status is 401 Unauthorized, with a message: "Invalid user Name or User not registered".

Below the request, the terminal output shows the Node.js application starting and running on port 3000.

```
up to date, audited 125 packages in 768ms
19 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities
○ anshumanik@anshus-MacBook-Pro NodeJS % npm start
> nodejs@1.0.0 start
> nodemon server.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Database connected successfully
Server is running on port 3000...
```

The screenshot shows the Compass MongoDB interface. It displays two documents in the `users` collection of the `test` database.

Document 1:

```
_id: ObjectId('68a052cf9f3e939c0b7148b')
id: 1
username: "anshu Manik"
password: "password"
createdAt: 2025-08-16T09:43:43.248+00:00
updatedAt: 2025-08-16T09:43:43.248+00:00
__v: 0
```

Document 2:

```
_id: ObjectId('68a076cc44a3ella116e6730')
id: 2
username: "internshala"
password: "password"
createdAt: 2025-08-16T12:17:16.556+00:00
updatedAt: 2025-08-16T12:17:16.556+00:00
__v: 0
```

23. POST user API to login user (no auth required to access login api) with input data in body but user password not in provided:

The screenshot shows the Thunder Client interface. A POST request is being made to `http://localhost:3000/api/user/login`. The request body contains the following JSON:

```
1 {
2   "userName" : "internshala"
3 }
```

The response status is **400 Bad Request**, size **34 Bytes**, and time **3 ms**. The response body is:

```
1 {
2   "message": "password not entered"
3 }
```

The terminal pane shows the Node.js application output:

```
up to date, audited 125 packages in 768ms
19 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
o anshumanik@anshus-MacBook-Pro NodeJS % npm start
> nodejs@1.0.0 start
> nodemon server.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting 'node server.js'
Database connected successfully
Server is running on port 3000...
```

24. POST user API to login user (no auth required to access login api) with input data in body but incorrect user password:

The screenshot shows the Thunder Client interface. A POST request is being made to `http://localhost:3000/api/user/login`. The request body contains the following JSON:

```
1 {
2   "userName" : "internshala",
3   "password": "password1"
4 }
```

The response status is **401 Unauthorized**, size **53 Bytes**, and time **4 ms**. The response body is:

```
1 {
2   "message": "Invalid password for the given username"
3 }
```

The terminal pane shows the Node.js application output:

```
up to date, audited 125 packages in 768ms
19 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
o anshumanik@anshus-MacBook-Pro NodeJS % npm start
> nodejs@1.0.0 start
> nodemon server.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting 'node server.js'
Database connected successfully
Server is running on port 3000...
```

Test result summary:

- 1. Protected cart routes, so only logged-in users can access them.**
- 2. Implemented JWT-based authentication.**
- 3. Implemented error handling for all API routes.**
- 4. Validated input data (e.g., check if user ID exists before registering new user).**
- 5. Implemented CRUD operations on MongoDB collections for products and cart items.**
- 6. API runs without errors.**