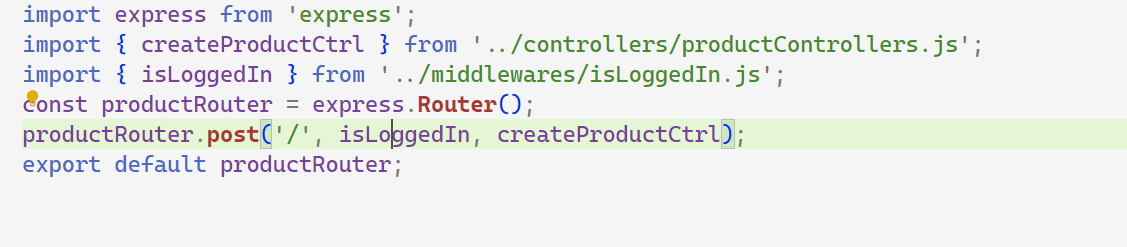
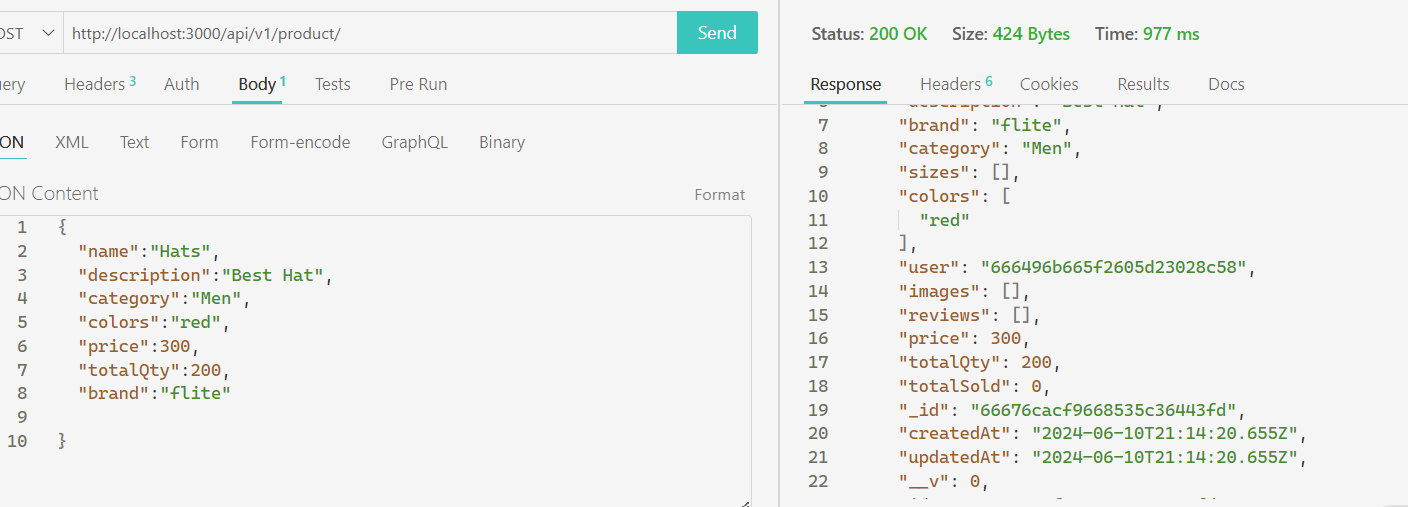
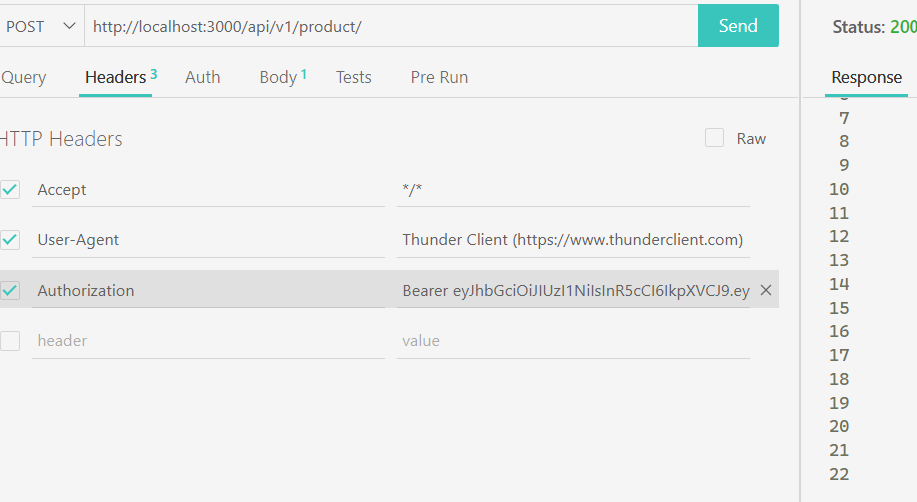
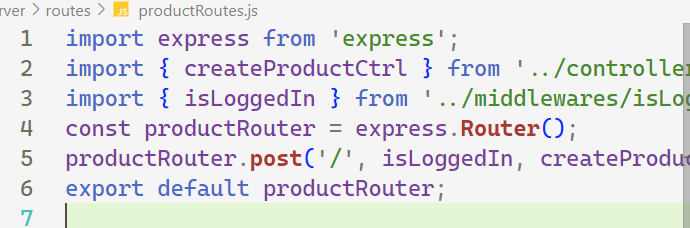
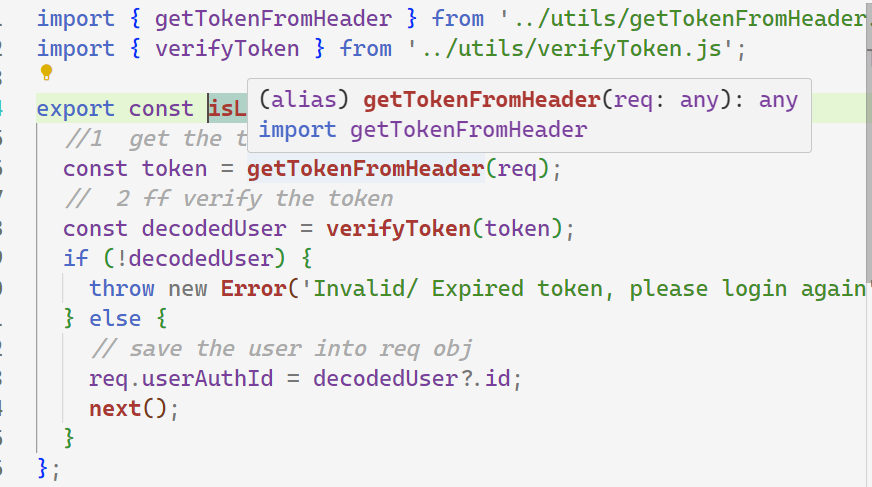
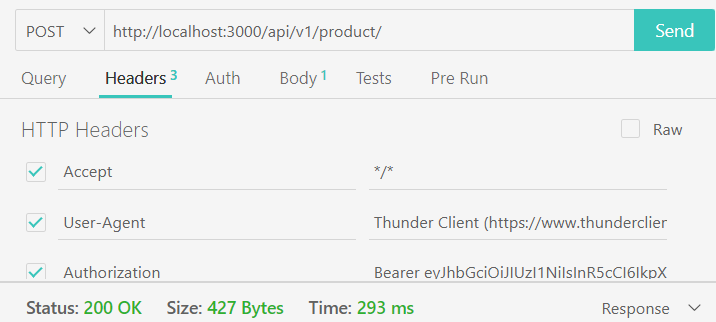
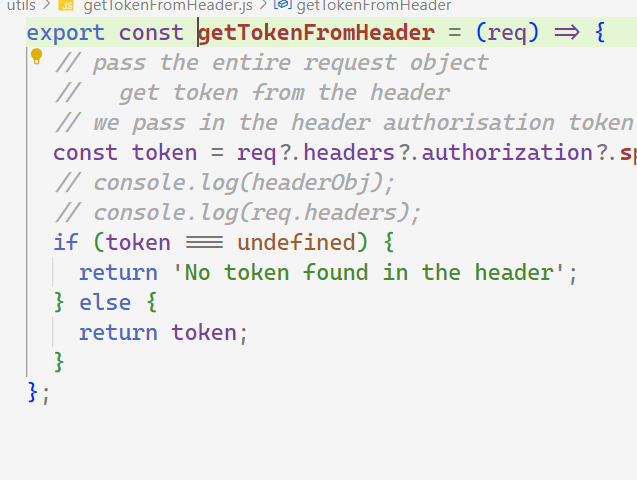
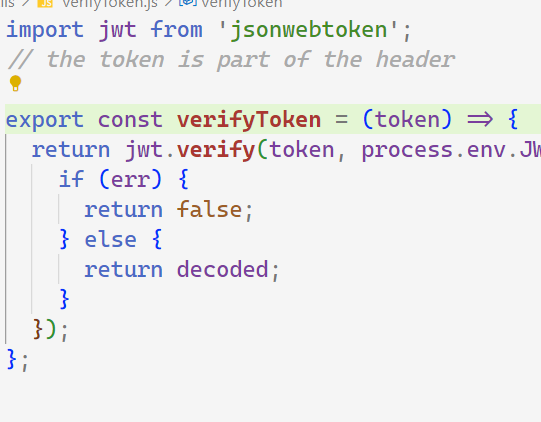
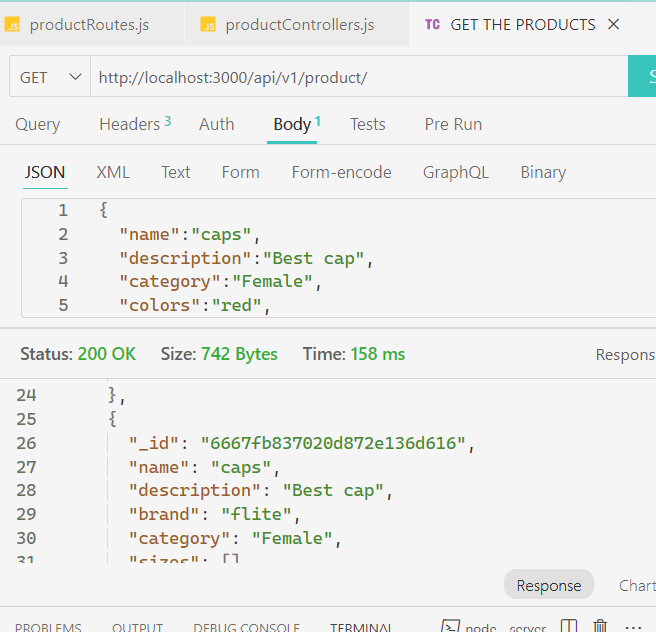
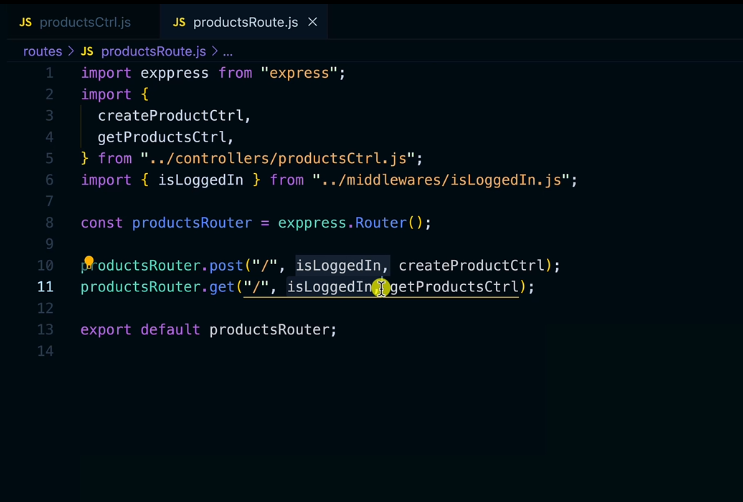
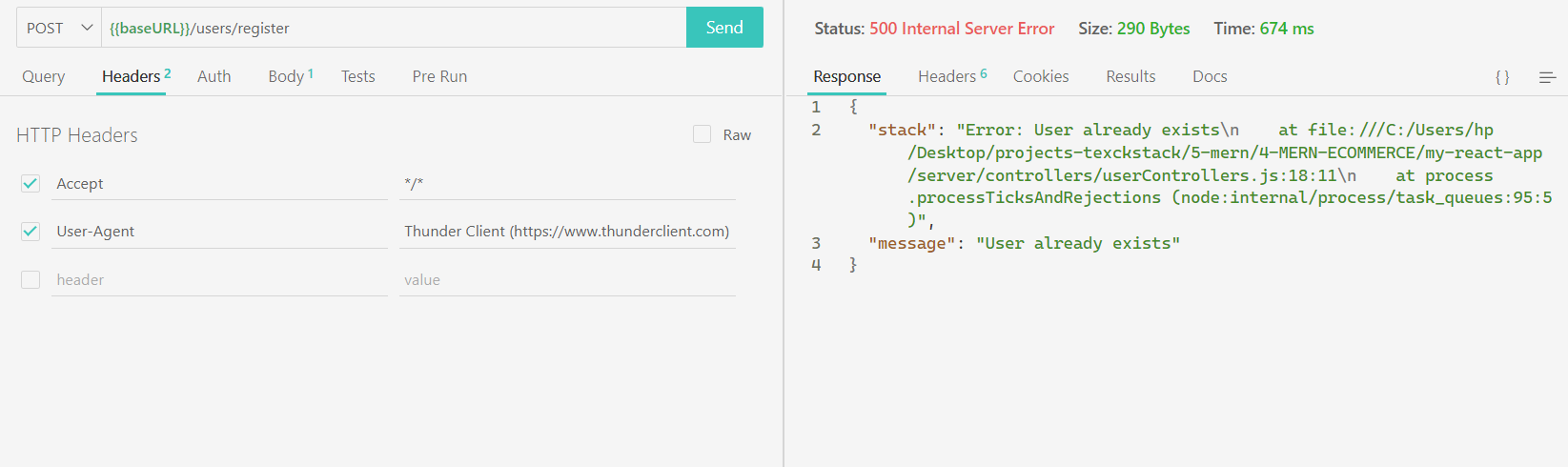
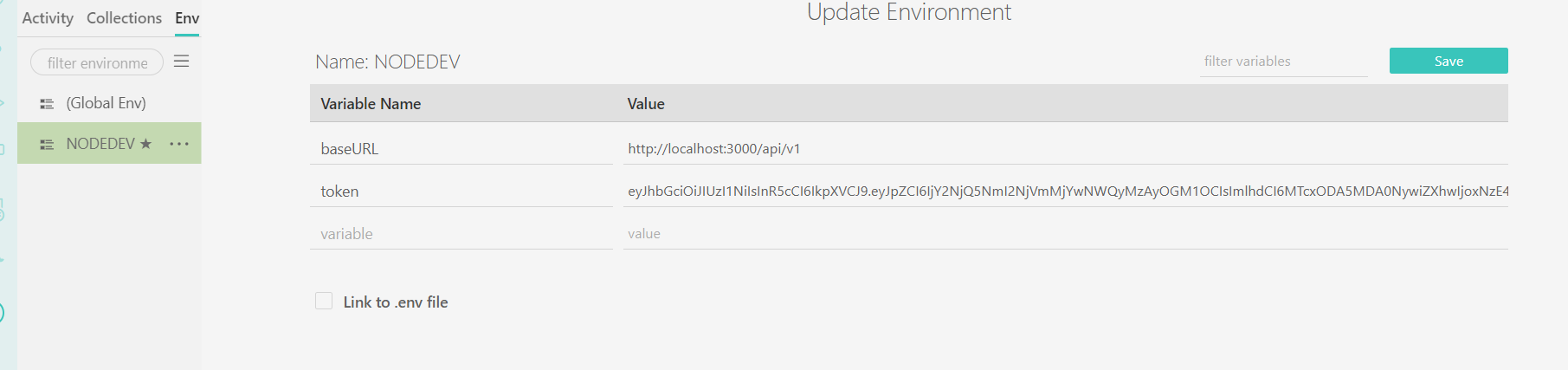
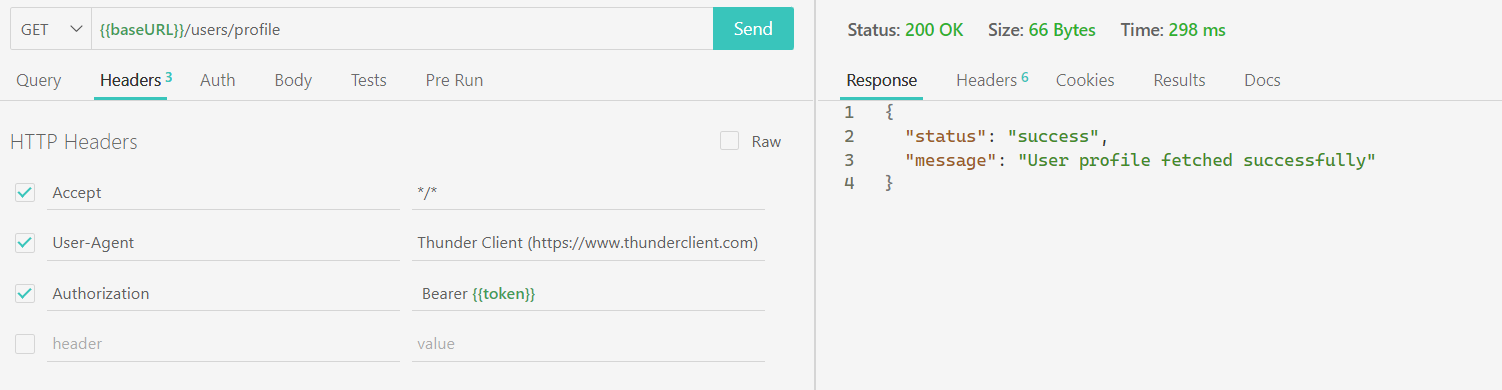
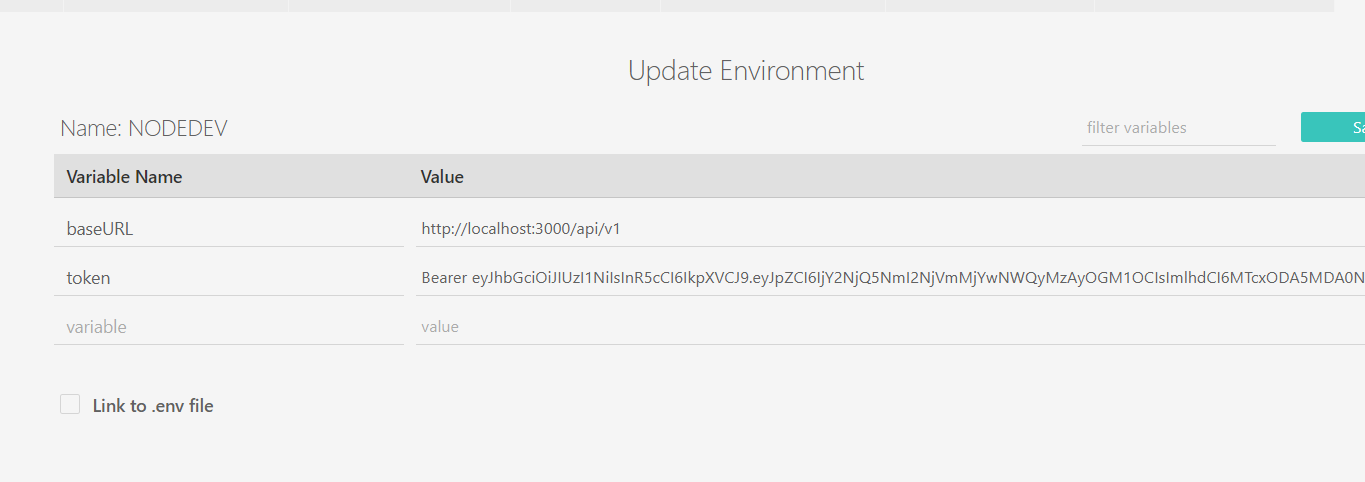
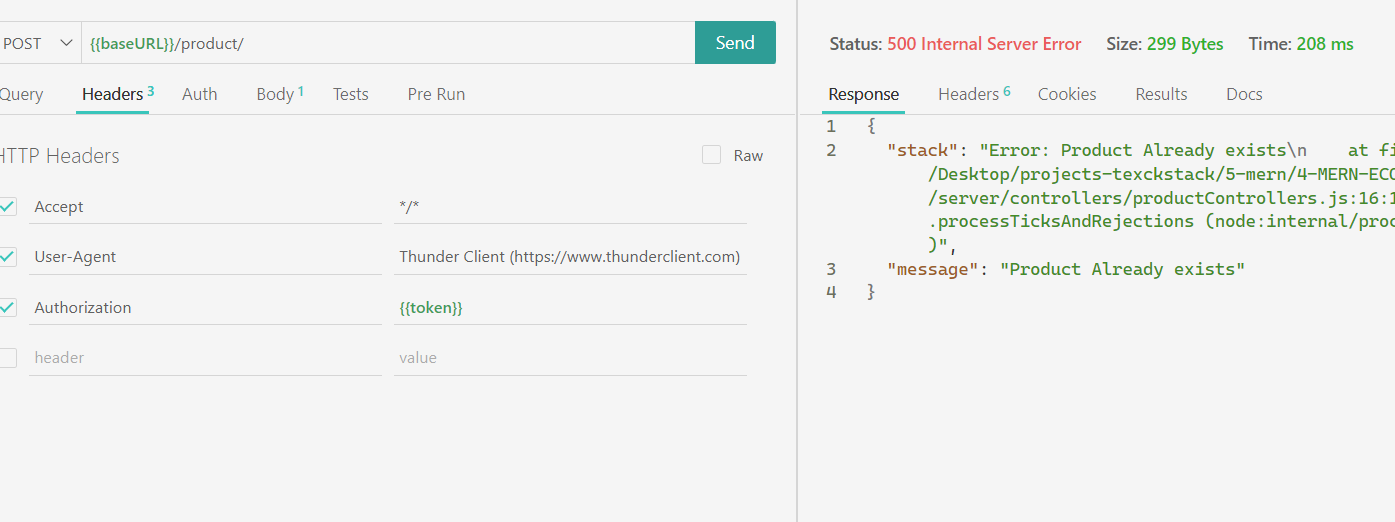
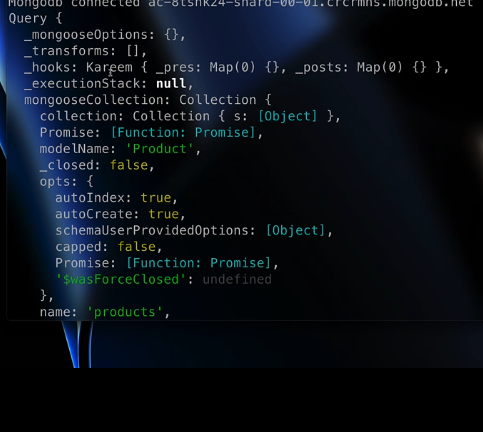
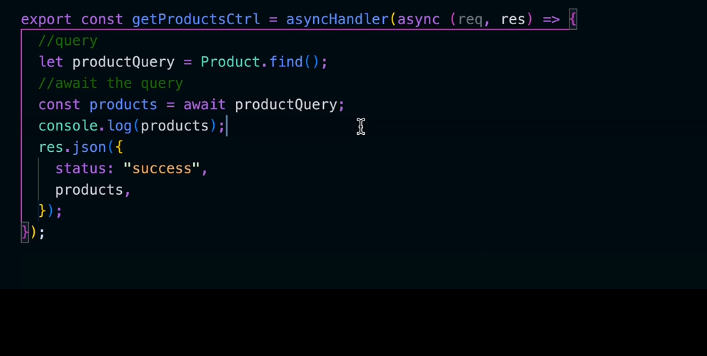
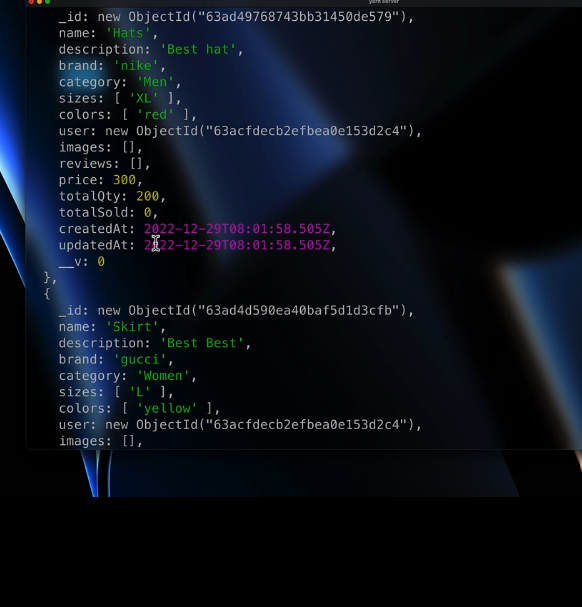
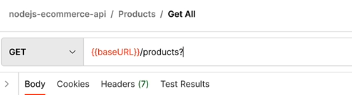
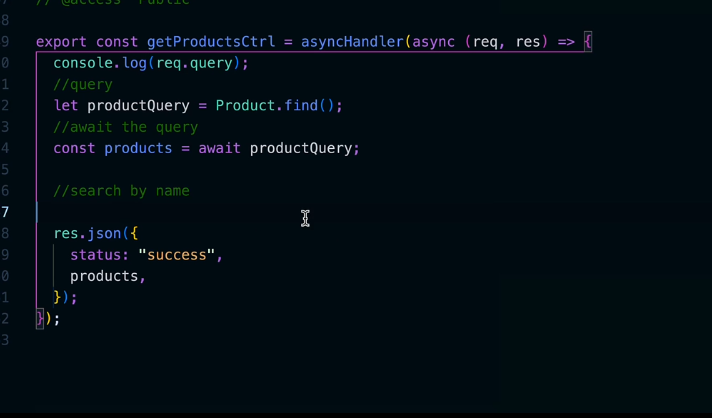
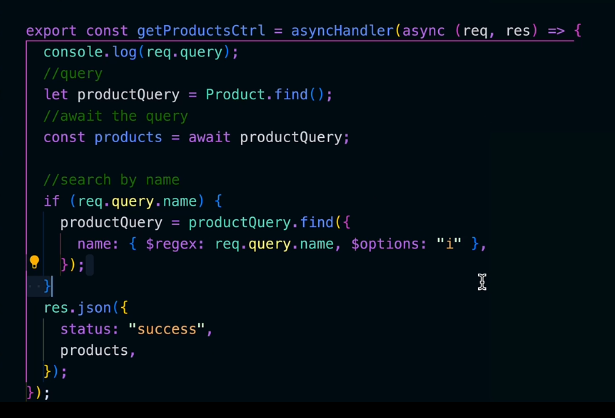
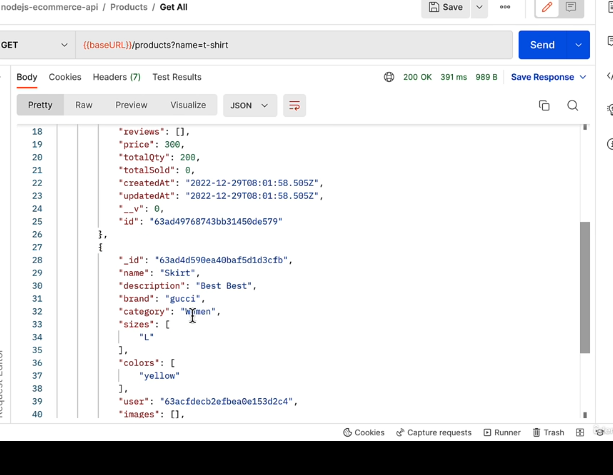
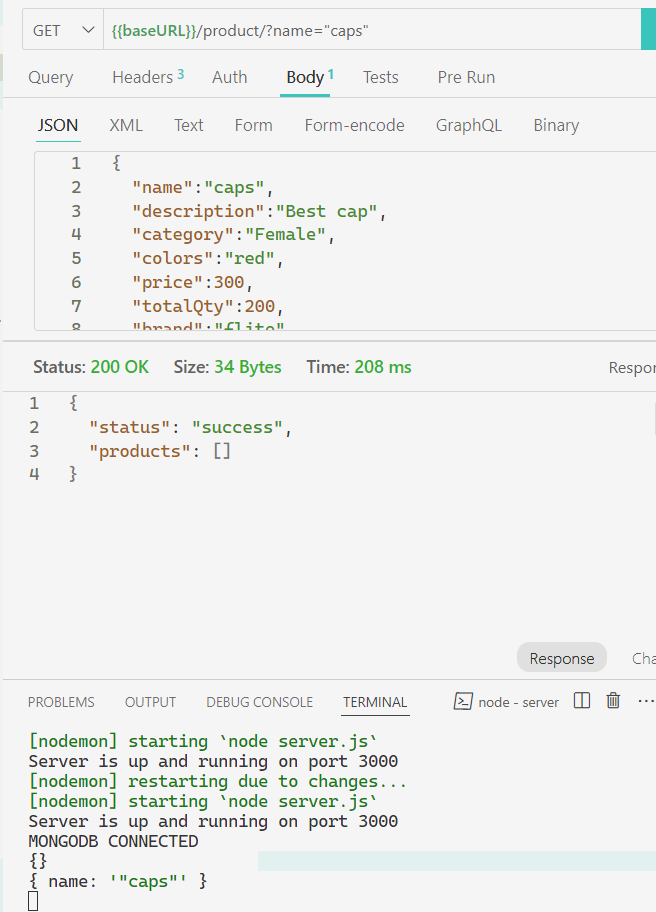
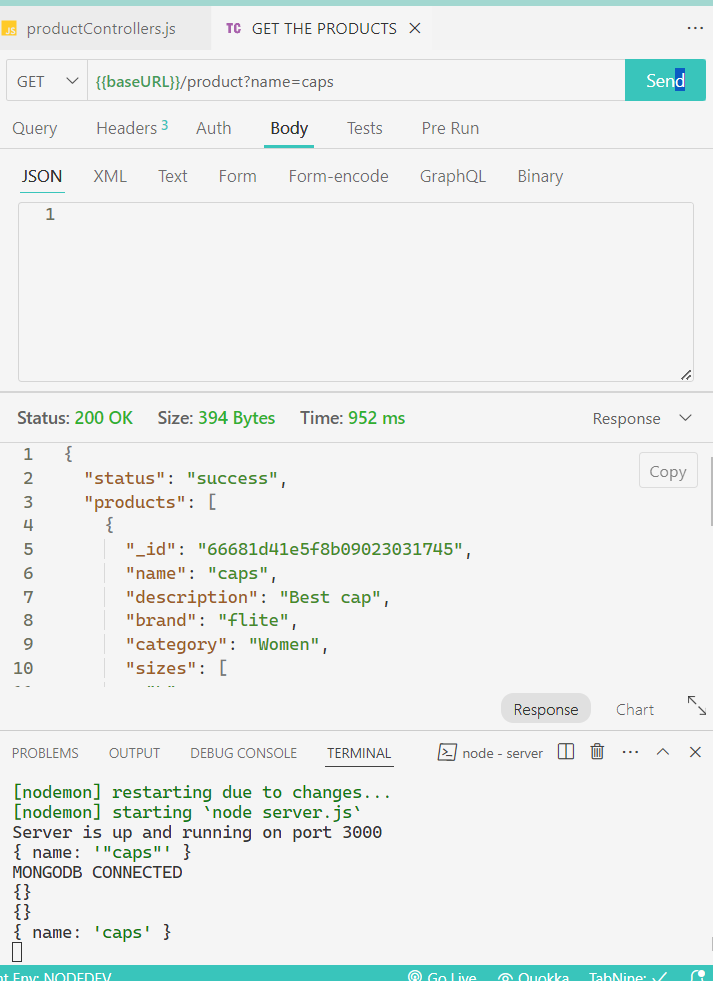
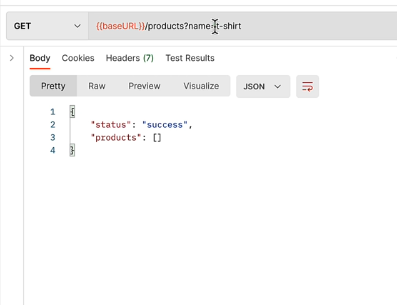
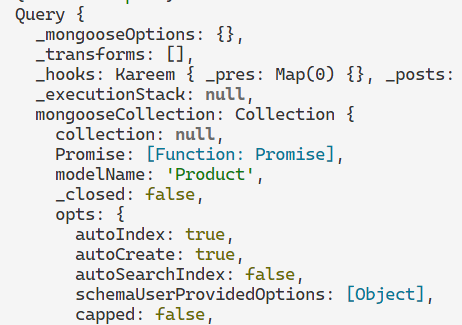
1. Product model.js
   * + 1. Create the product.model.js
2. CREATE PRODUCT CONTROLLER
   * + 1. Create the productController.js and then product.routes and then map into the app.js
       2. FLOW
          1. Make the product controller which include the id of user which created so we make the logic includes in controller nad then while making the routes we include the isLoggedIn.js middle ware
          2. 
          3. 
          4. 
          5. Which takes the token when the user login and send the json
          6. Same procedure
       3. Add the Bearer token and req object has this token same in islOgedin.js
       4. getTOkenfromHeader.js and verified token and then add the req.userAuthId filed for future uses decodedUser?.id
       5. 
       6. 
       7. 
       8. COPY THE TOKEN USER Login
       9. 
       10. 
       11. 
3. FETCHING ALL THE PRODUCTS
   * + 1. 
       2. Make the GETPRODUCTCTRL in the products controller and make the specific route for that function
       3. 
4. ENVIRONMENT VARIABLES
   * + 1. 
       2. 
       3. 
       4. 
       5. 
       6. 
5. FILTER PRODUCTS BY NAME
   * + 1. 
       2. IN MONgoose methods returns the query
       3. 
       4. 
       5. 
       6. All the products fetched
       7. 
       8. Aftert the “? ” is the queryString
       9. Passing the additional payload
       10. Optional
       11. 
       12. 
       13. Ignore the upper and lower casing
       14. 
       15. 
       16. Still got the results even change to t-shirt
       17. Because we first store the product query and if statement not run so it return from the top one productQuery
       18. 
       19. 
       20. 
       21. 
       22. Np product by t-shirt
       23. Flow
       24. So we make productQuery in let keyword for change in future .
       25. 
       26. Which is the mongoose function returns the query in object
       27. So we then check the in the if statement for ?name=caps
       28. If exists then store the productQuery other send the empty array
       29. 

[

{

\_id: new ObjectId('66681d41e5f8b09023031745'),

name: 'caps',

description: 'Best cap',

brand: 'flite',

category: 'Women',

sizes: [ 'L' ],

colors: [ 'orange' ],

user: new ObjectId('666496b665f2605d23028c58'),

images: [],

reviews: [],

price: 300,

totalQty: 200,

totalSold: 0,

createdAt: 2024-06-11T09:47:45.574Z,

updatedAt: 2024-06-11T09:47:45.574Z,

\_\_v: 0

},

{

\_id: new ObjectId('66681d8fe5f8b09023031748'),

name: 'shirts',

description: 'Best shirt',

brand: 'flite',

category: 'Men',

sizes: [ 'XL' ],

colors: [ 'orange' ],

user: new ObjectId('666496b665f2605d23028c58'),

images: [],

reviews: [],

price: 300,

totalQty: 200,

totalSold: 0,

createdAt: 2024-06-11T09:49:03.793Z,

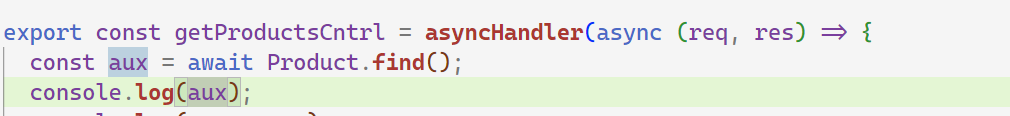
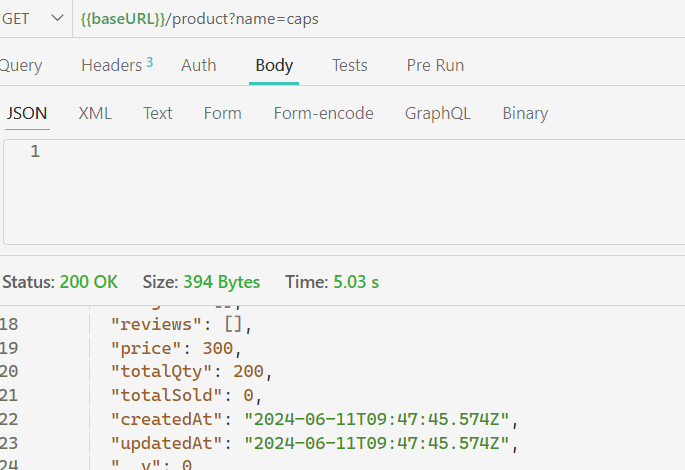
updatedAt: 2024-06-11T09:49:03.793Z,

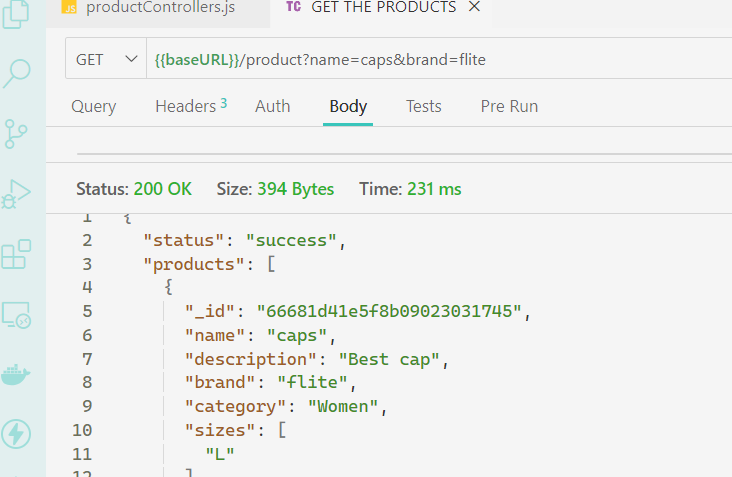
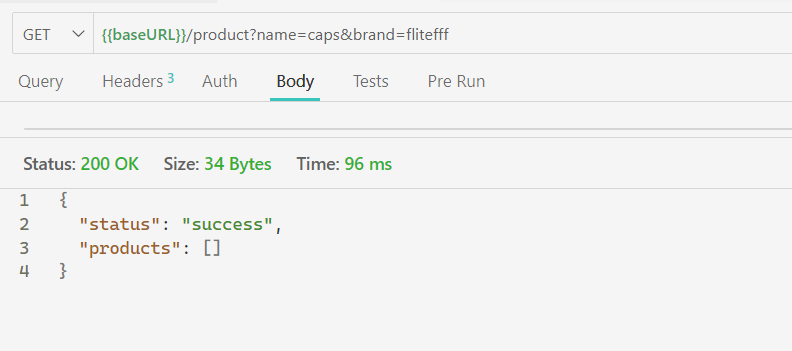
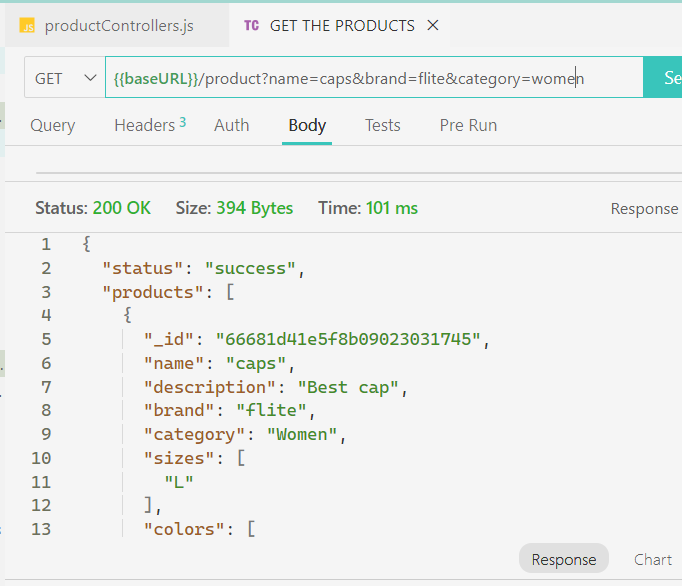
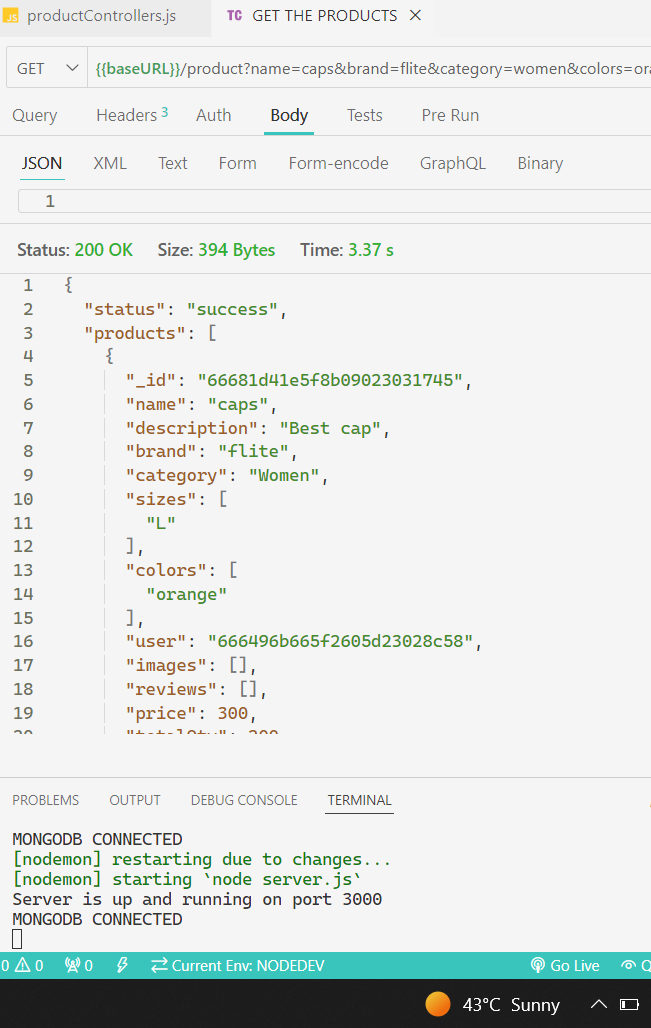
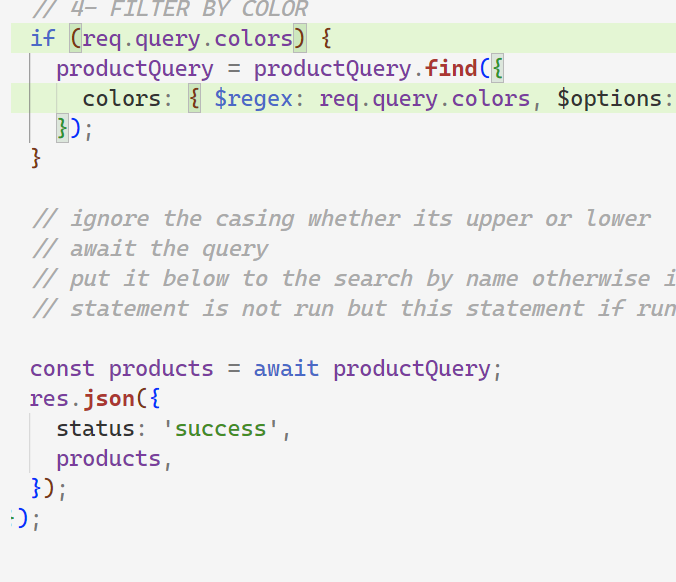
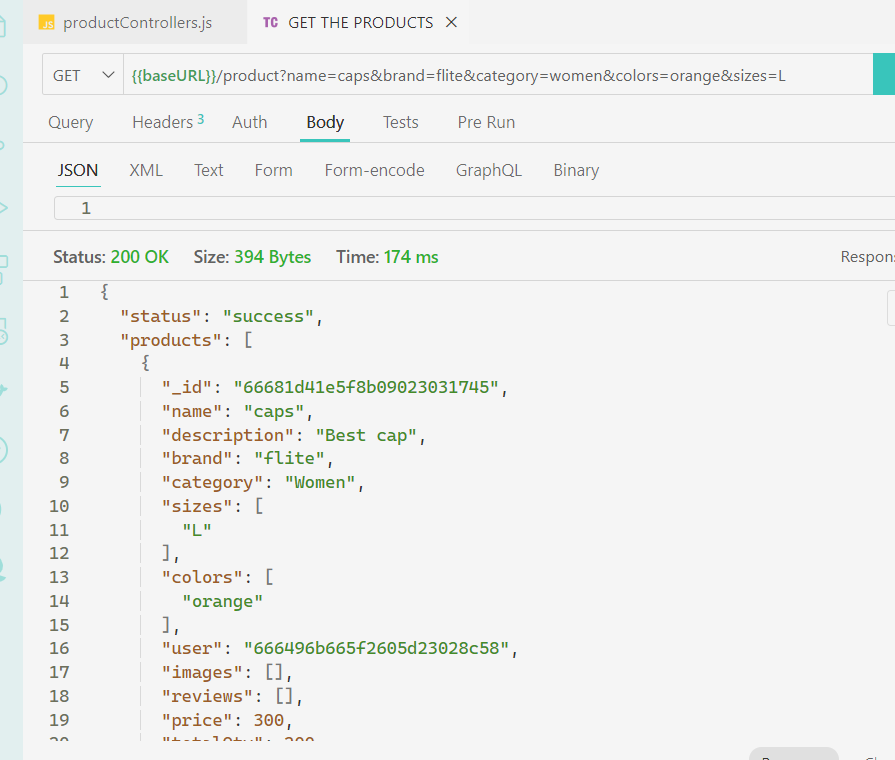
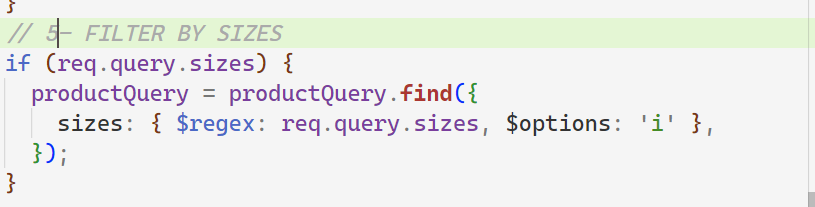
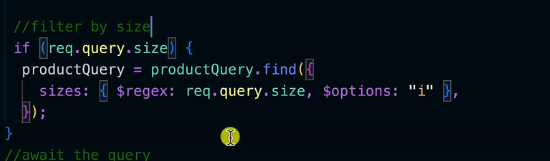
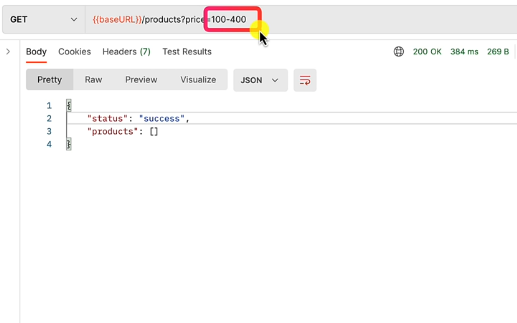
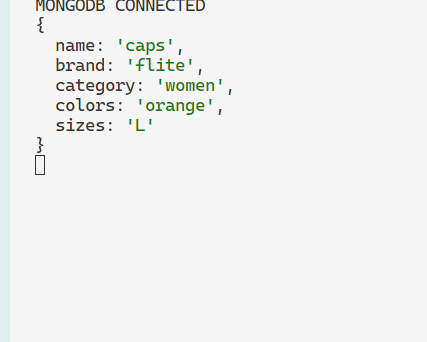
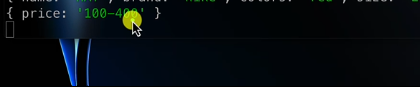
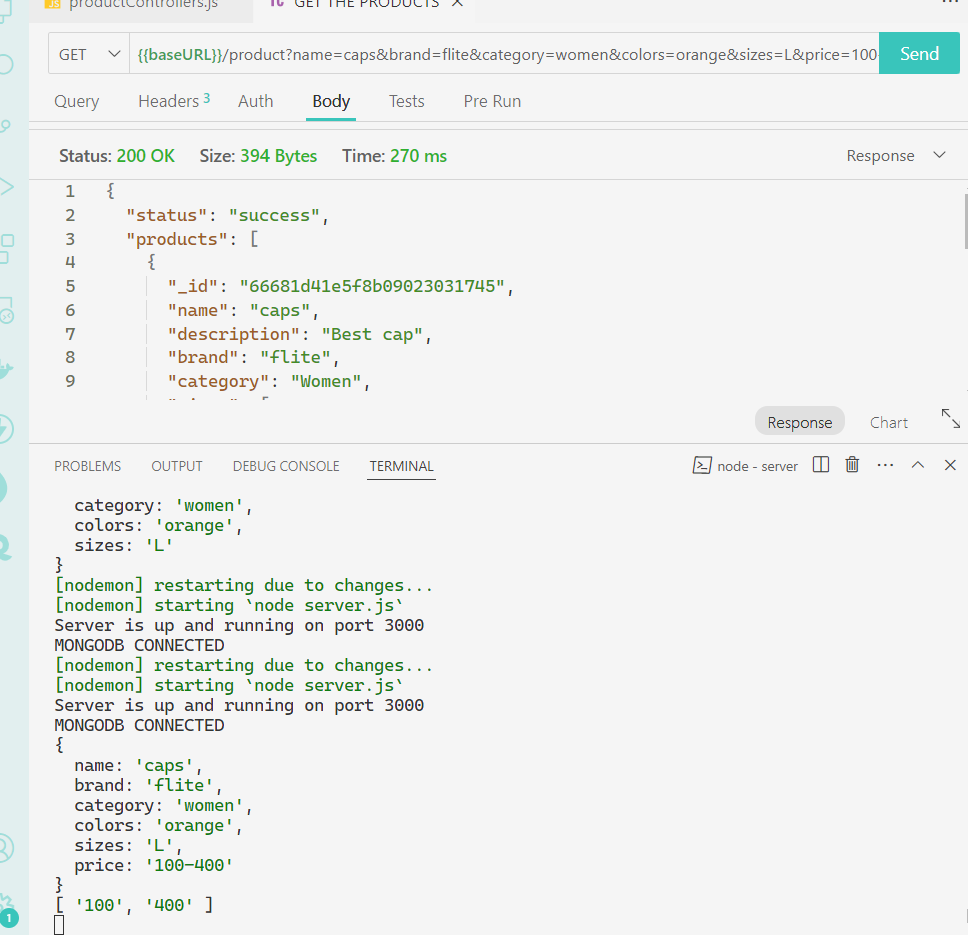
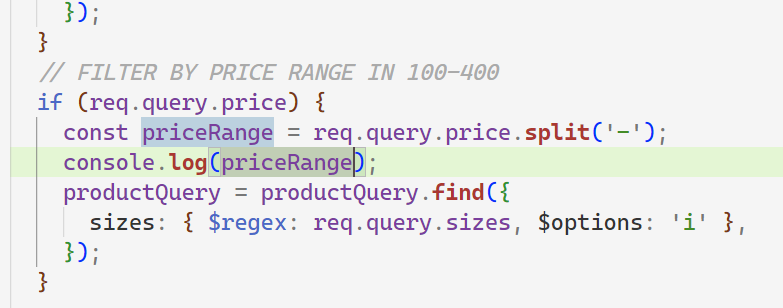
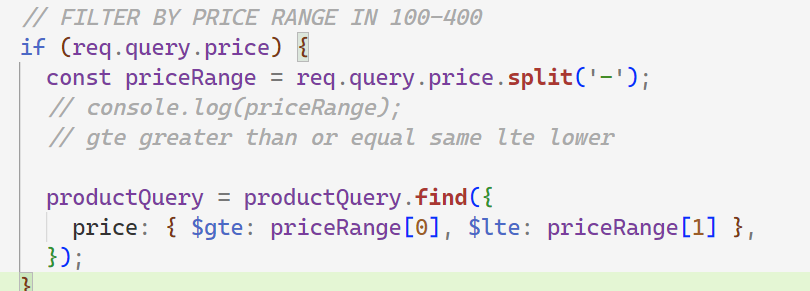
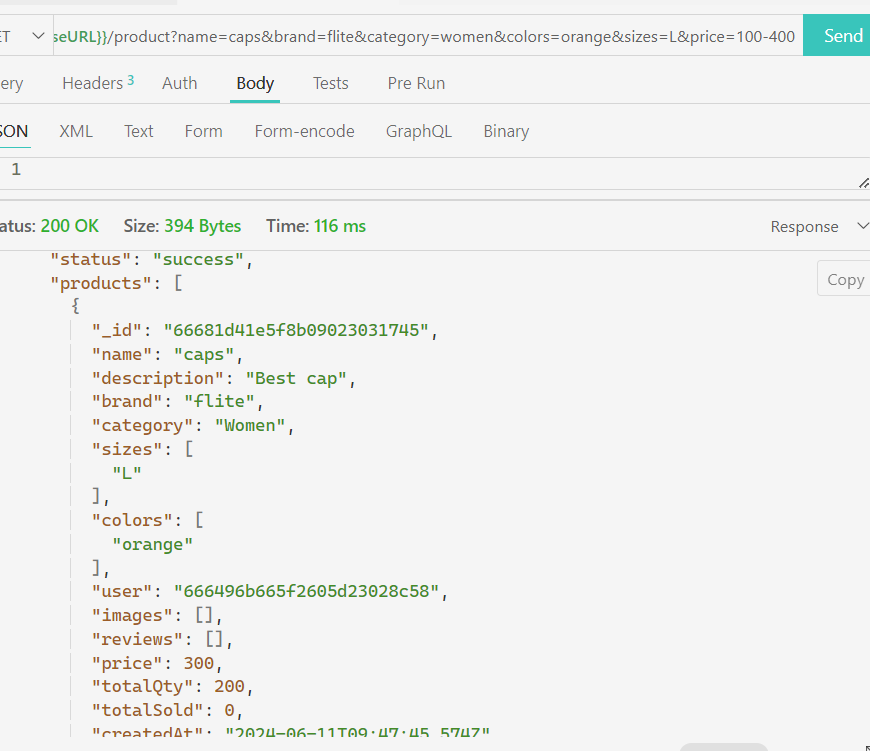
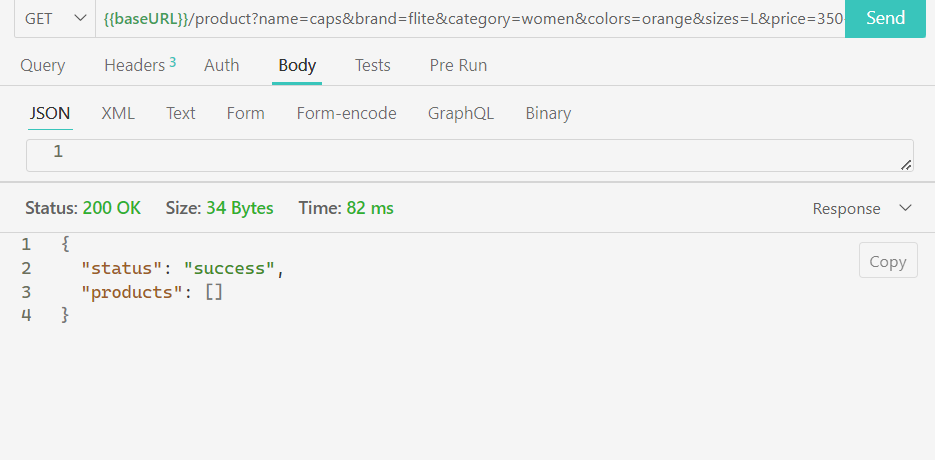
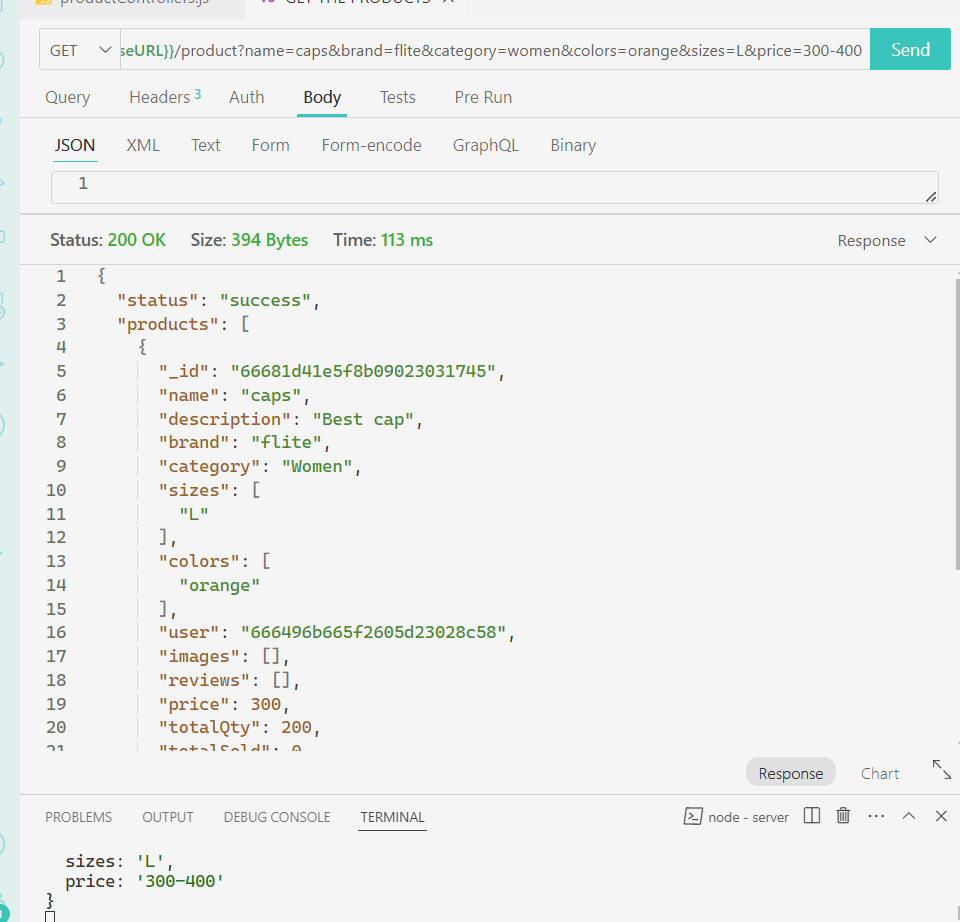
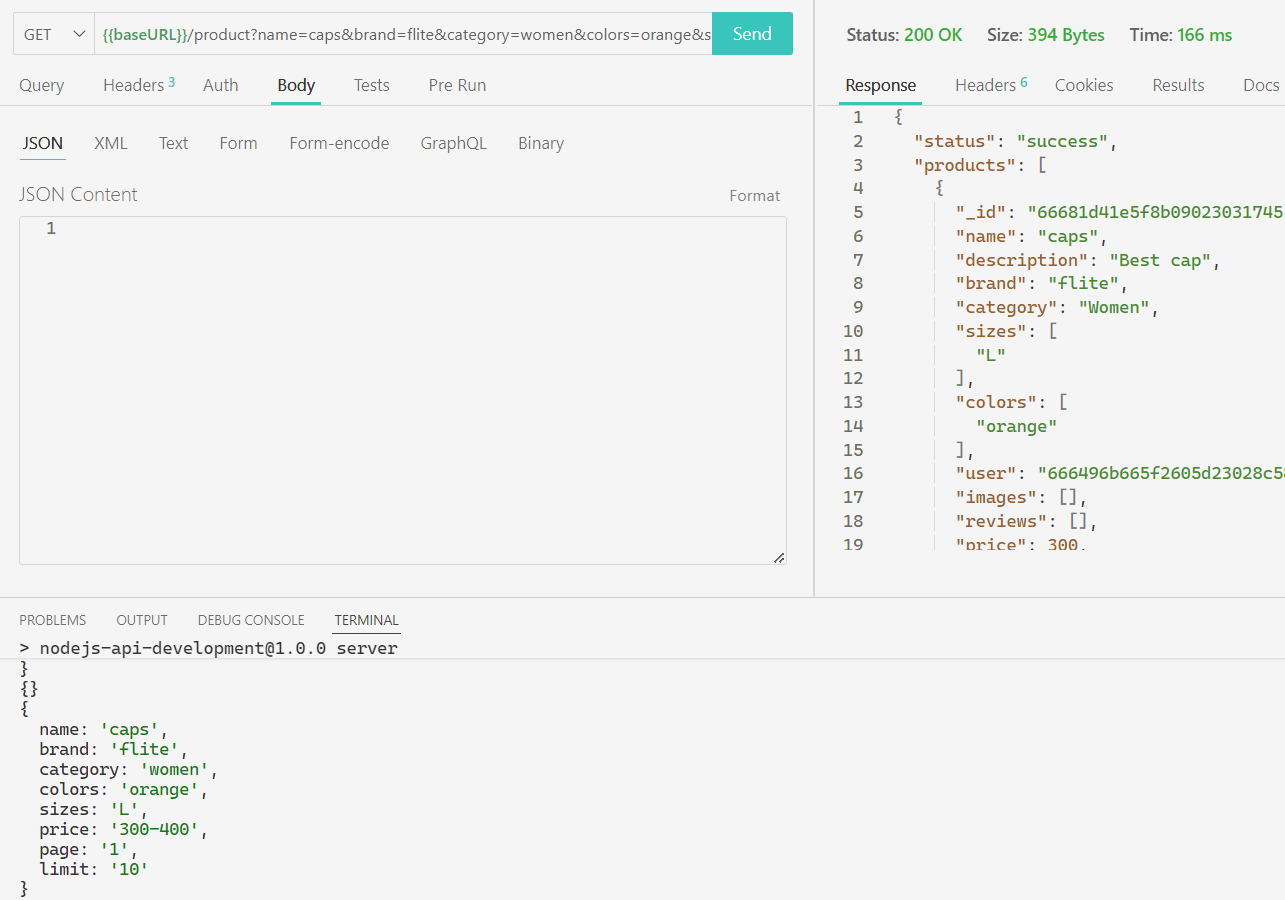
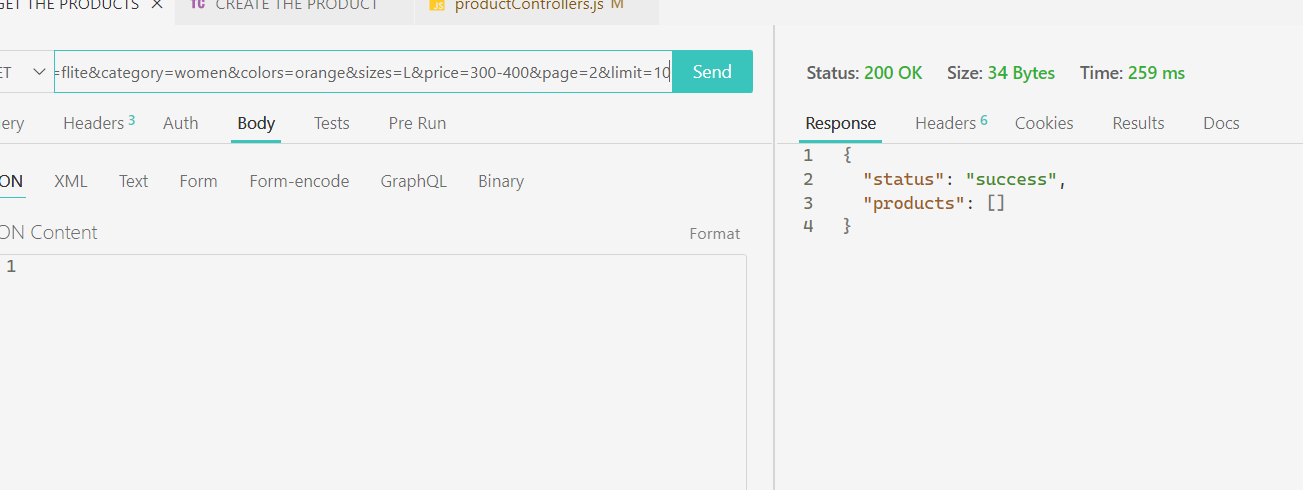
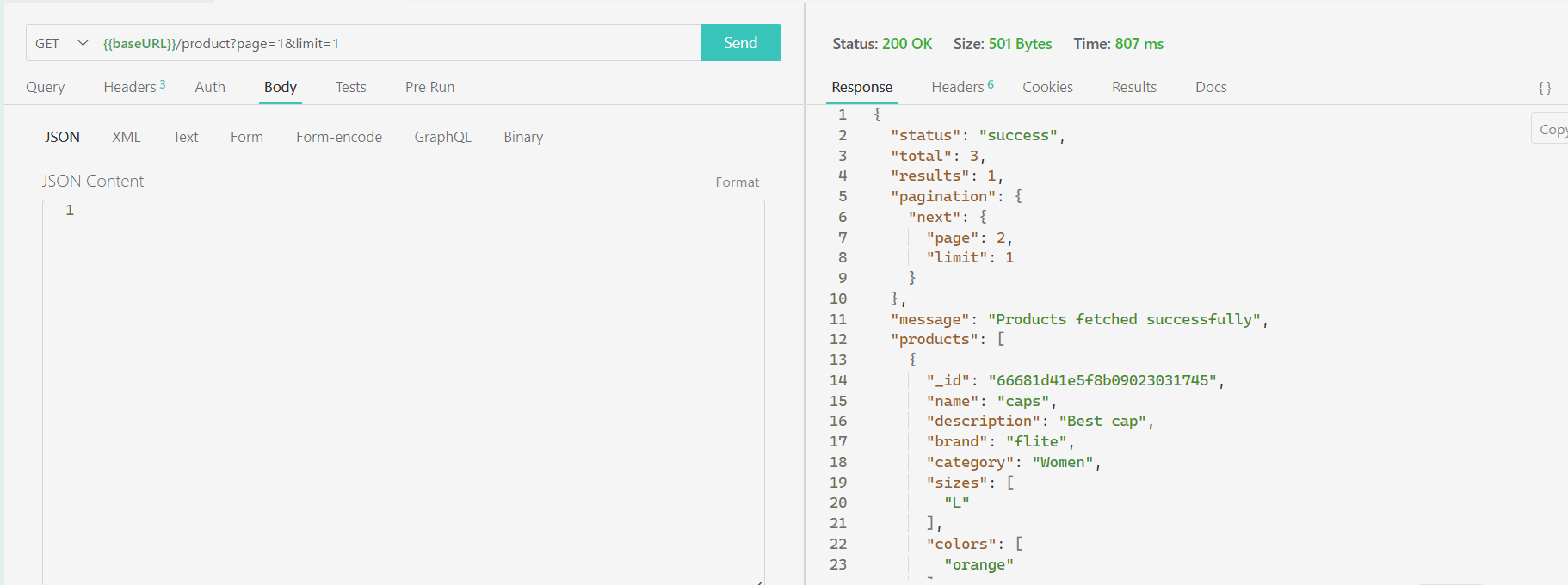
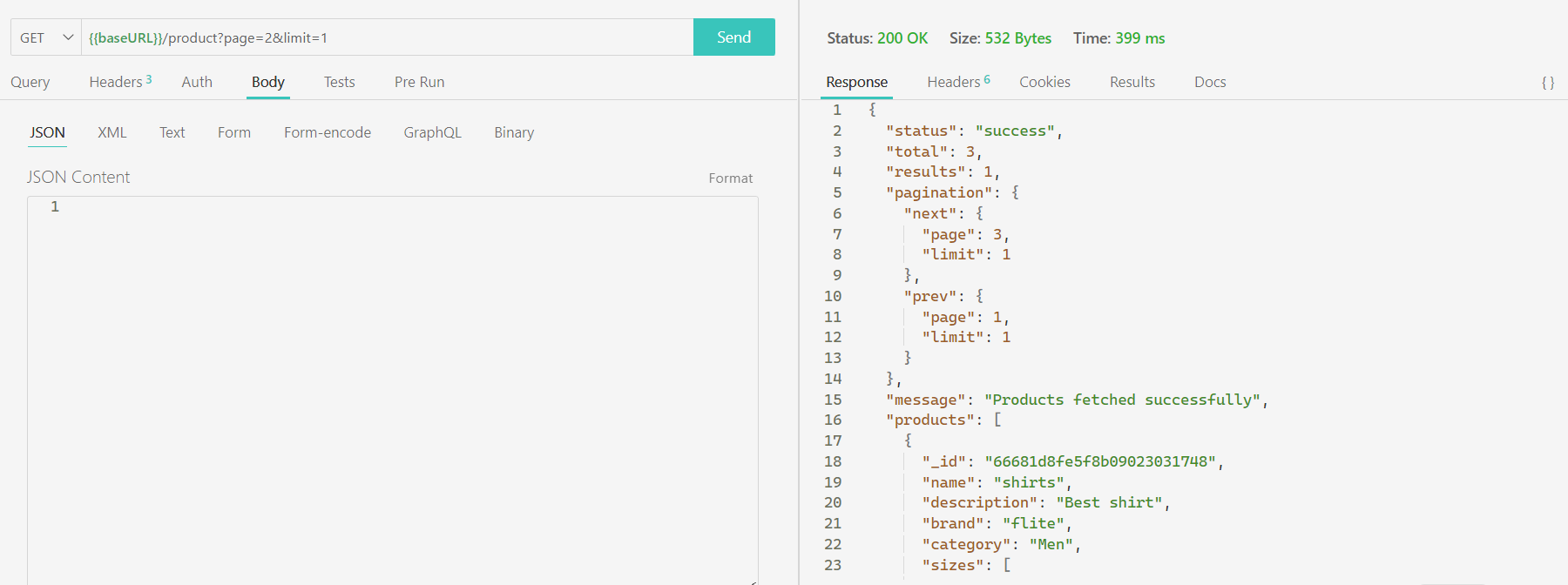
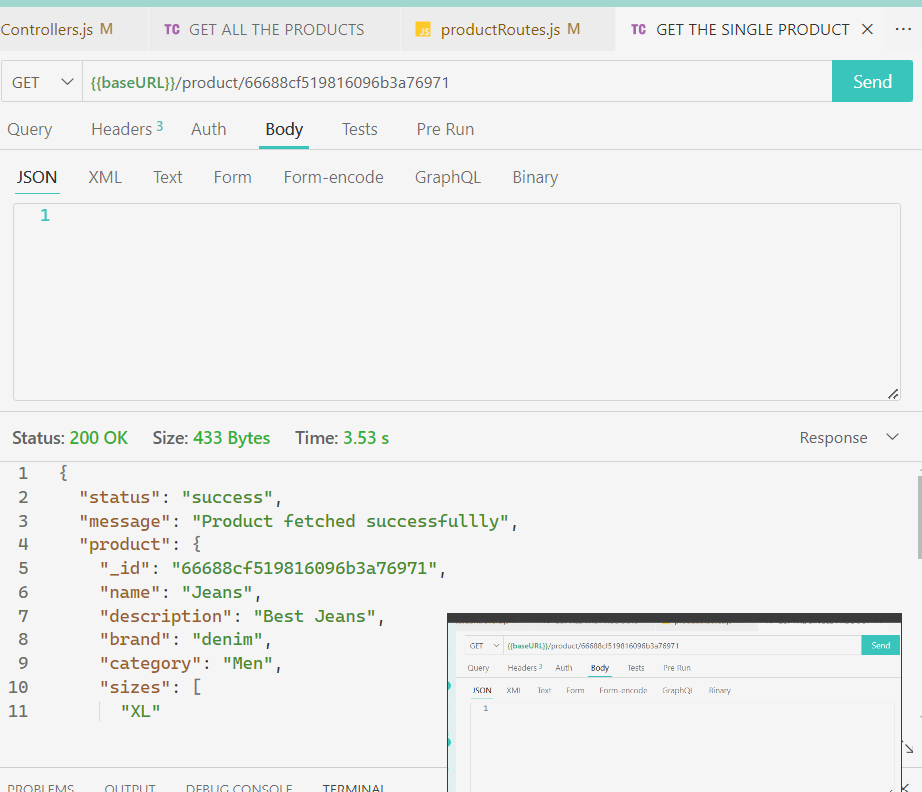
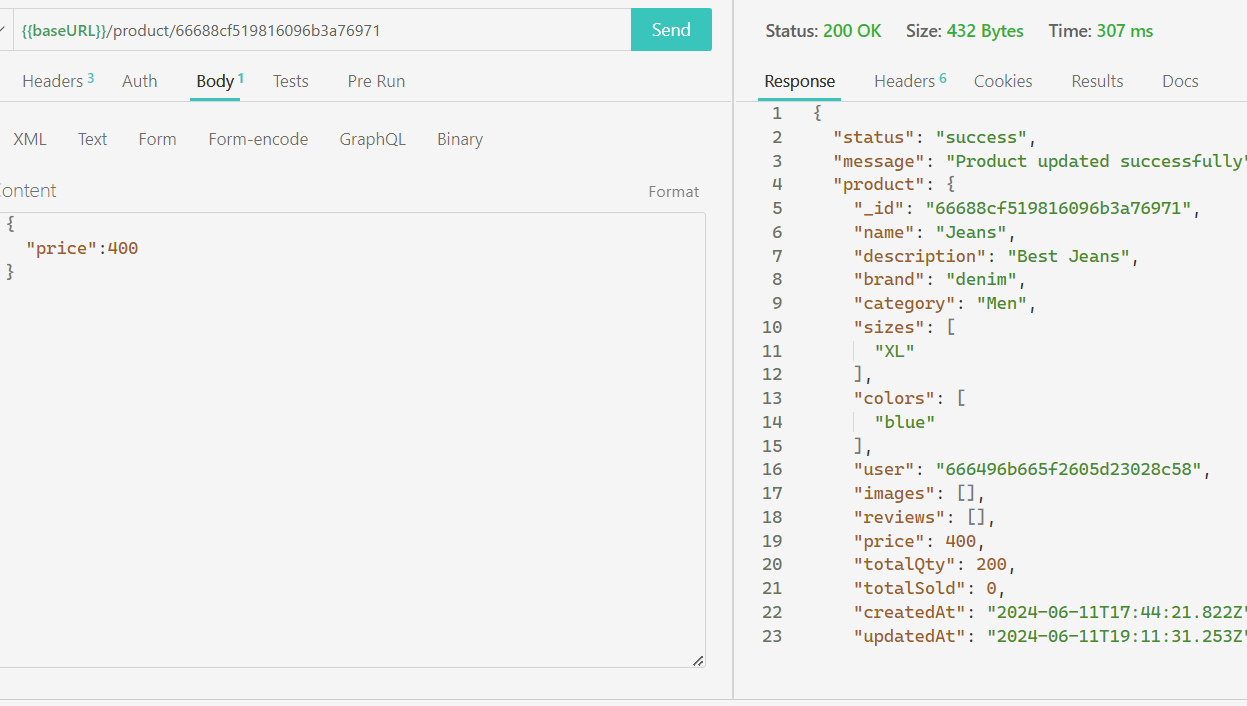
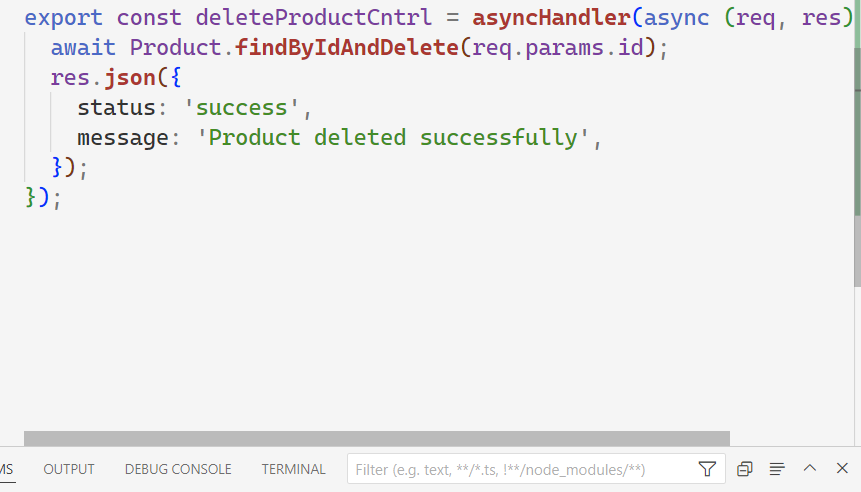
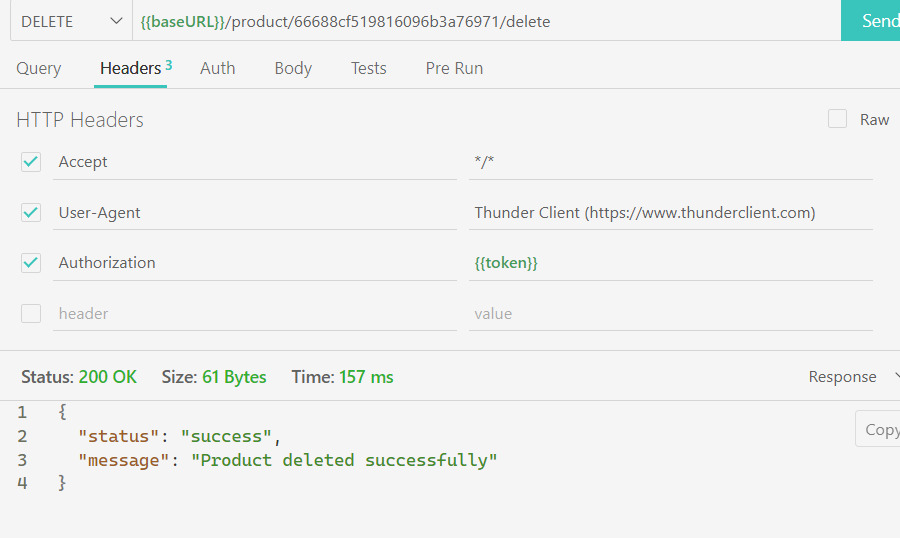
\_\_v: 0

}

]

{ name: 'caps' }

* + - 1. 
      2. Fetch all the data like previously
      3. 

1. FILTER PRODUCT BY COLOR
   * + 1. 
       2. 
       3. Non-exist brand
       4. Filter by brand
       5. 
       6. & means both caps and band is flite
       7. Filter by category
       8. 
       9. 
       10. Filter by colors
       11. 
       12. 
       13. Filter by size
       14. 
       15. 
       16. 
       17. 
       18. Should be same
       19.  I take as “sizes” same in database and he size in query
       20. And 
       21. Same as in database field name
2. FILTER PRODUCT BY PRICE RANGE
   * + 1. 
       2. MONGODB will do it in range
       3. 
       4. Req.query
       5. 
       6. Split by “-” and take first one and second one
       7. 
       8. 
       9. 
       10. 
       11. All the products lies between 100-400 greater or equat to 100 and lower and equal to 400
       12. A product got one price and range got all the products in ranges means return all the products range [100-400]
       13. 
       14. 
3. PRODUCT PAGINATION
   * + 1. 
       2. Mae the pagination {{baseURL}}/product?name=caps&brand=flite&category=women&colors=orange&sizes=L&price=300-400&page=1&limit=10
       3. &page=1&limit=10
       4. 
       5. 
4. PAGINATION RESULT
   * + 1. 
       2. {{baseURL}}/product?name=caps&brand=flite&category=women&colors=orange&sizes=L&price=300-400&page=1&limit=10
       3. NOW I HAVE 3 PRODUCT IF I SET LIMIT TO 1 THEN EACH PAGE GET 1 PRODUCT LET SEE
       4. 
       5. 
       6. 
5. GET THE SINGLE PRODUCT
   * + 1. MAKE THE function in the product controller
       2. 
       3. 
6. UPDATE THE PRODUCTS
   * + 1. 
       2. Update the product by the id
       3. 
       4. 
       5. 
7. DELETE THE PRODUCT
   * + 1. 
       2. 
       3. 
       4. 