Machine Test Question for Node.js Backend

Objective: The purpose of this machine test is to evaluate your programming skills in Node.js by creating a set of complex APIs for an e-commerce system.

Task: Develop five APIs that address various functionalities of an e-commerce platform. Ensure that your code is well-organized, follows best practices, and is scalable.

- User Registration API:
  - Allow users to register with the system by providing necessary details.
  - Implement validation for unique email addresses and strong password policies.
  - Generate and return an authentication token upon successful registration.
- Product Listing API:[public]
  - Fetch a paginated list of products available in the e-commerce system.
  - Include options for sorting and filtering based on different product attributes.
  - Implement pagination to handle a large number of products.
- Shopping Cart API:[private]
  - Enable users to add products to their shopping cart and manage the cart contents.
  - Include features like updating quantity, removing items, and clearing the entire cart.
  - Ensure that the shopping cart is associated with the user's authentication token.
- Checkout API:[private]
  - Implement a checkout process to place an order using the items in the user's shopping cart.
  - Calculate the total order amount, including taxes and shipping charges.
  - Deduct the purchased items from the inventory and update order status.
- Order History API:
  - Allow users to view their order history, including details of past purchases.
  - Implement pagination for the order history list.
  - Include relevant information such as order status, date, and total amount.

Notes:

- You are free to use any Node.js version (preferably the latest stable release).
- Leverage Node.js's Eloquent ORM for database interactions.
- Implement middleware for authentication and consider using Node.js Passport for token management.

- Write clean and well-documented code.
- Provide brief instructions on how to test the APIs.

Submission: Please email the screenshots of postman API testing, Database backup as SQL script, Git URL to download the source and any additional instruction/information.

Database

User

- user_id PK
- name
- email
- phone
- password(md5)

Products

- product_id PK
- price
- MRP
- discount_p
- Image
- Brand
- Tag
- category
- stock_bal

Cart_master

- Cart_id PK
- User_id FK->User
- status

Cart_child

- Cart_child_id
- Cart_id FK -> Cart_master
- Product_id -> product FK
- qty

Sales_master

- sales_id
- user_id
- createdAt
- Total_amt
- tax _amt
- delivery_charge

Cart_child

- Cart_child_id
- Cart_id FK -> Cart_master
- Product_id -> product FK
- qty
- rate
- MRP
- Amount

User

- Allow users to register with the system by providing necessary details.
- Implement validation for unique email addresses and strong password policies.
- Generate and return an authentication token upon successful registration.

Signup API

curl --location 'localhost:5000/public/user/login' \

--header 'Content-Type: application/json' \

--data-raw '{

"name":"akram",

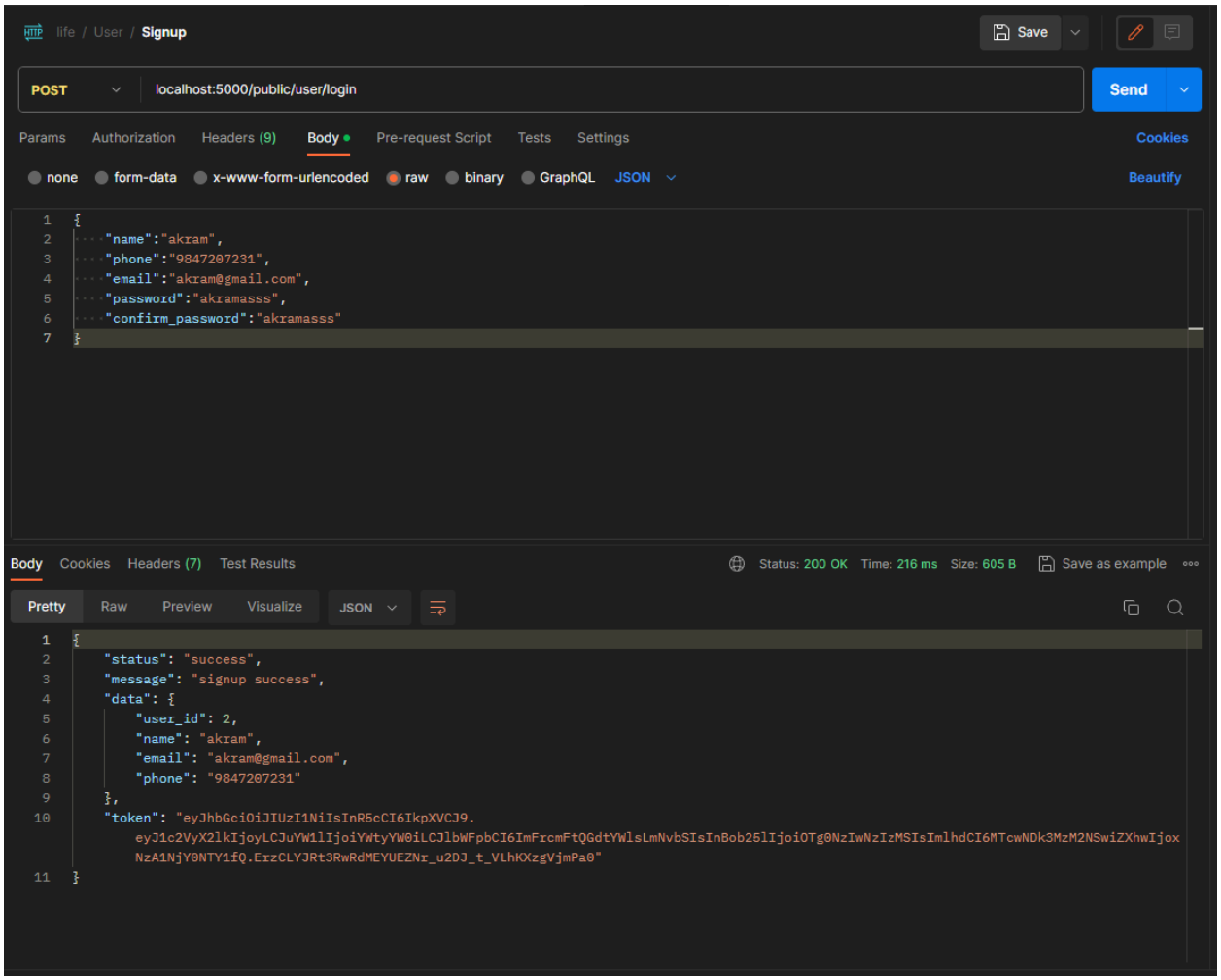
"phone":"9847207231",

"email":"akram@gmail.com",

"password":"akramasss",

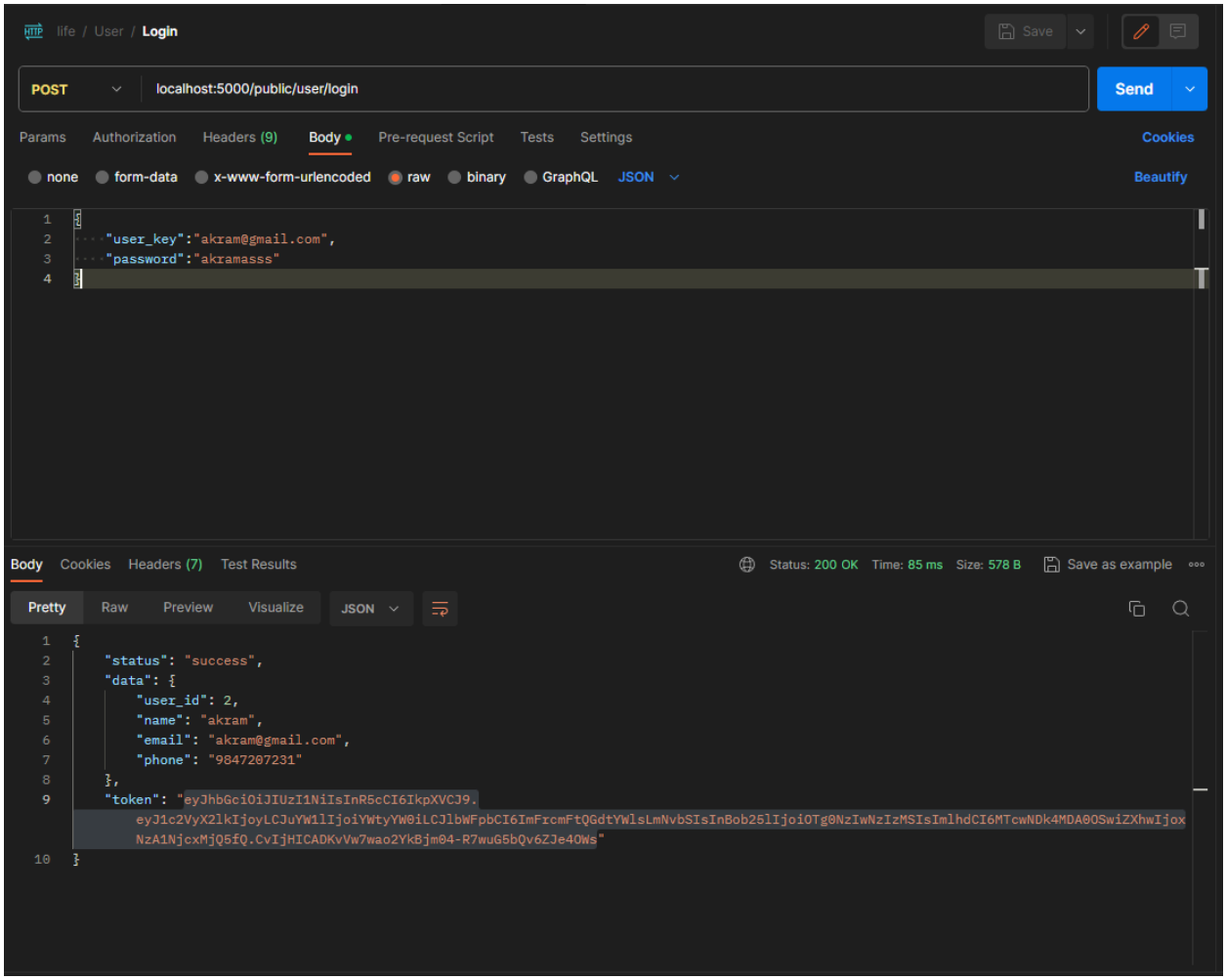
"confirm_password":"akramasss"

}'



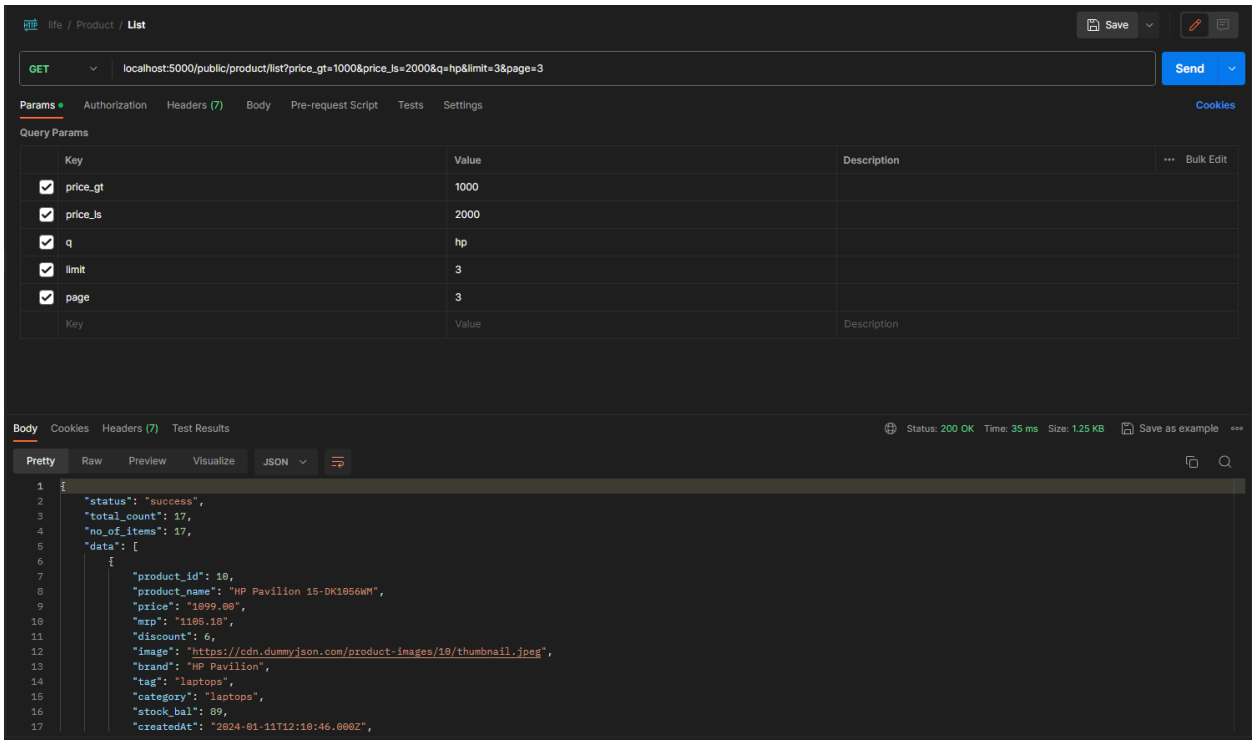
Login

```
curl --location 'localhost:5000/public/user/login' \
--header 'Content-Type: application/json' \
--data-raw '{
    "user_key":"akram@gmail.com",
    "password":"akramasss"
}'
```



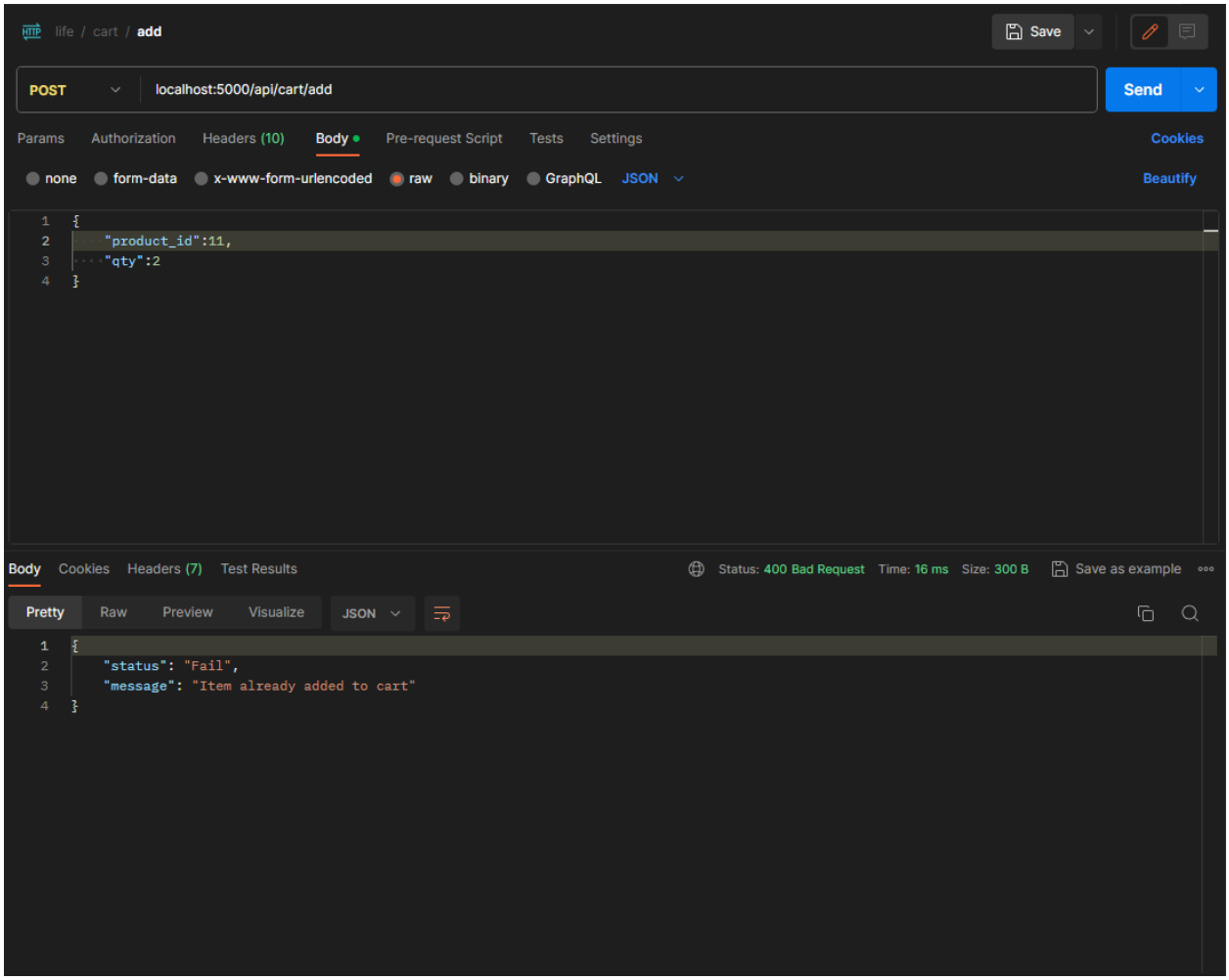
Product list

curl --location
'localhost:5000/public/product/list?price_gt=1000&price_ls=2000&q=hp&limit=3&page=3'



Add to cart

curl --location 'localhost:5000/api/cart/add' \
--header 'Authorization:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoyLCJuYW1lIjoiYWtyYW0iLCJlbWFpbCI6ImFrcmFtQGdtYWlsLmNvbSIsInBob25lIjoiOTg0NzIwNzIzMSIsImlhdCI6MTcwNDk4MDA0OSwiZXhwIjoxNzA1NjcxMjQ5fQ.CvIjHICADKvVw7wao2YkBjm04-R7wuG5bQv6ZJe4OWs' \
--header 'Content-Type: application/json' \
--data '{
"product_id":11,
"qty":2
}'


life / cart / add
Save
POST
localhost:5000/api/cart/add
Send
Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"product_id":11,
"qty":2
}
Body
Cookies
Headers (7)
Test Results
Status: 400 Bad Request
Time: 16 ms
Size: 300 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"status": "Fail",
"message": "Item already added to cart"
}