Cart Service

July 30, 2020 - Ravilochan Balla

Service Overview

Cart Service is a microservice written in NodeJS using ExpressJS , MongoDB as Database , mongoose ORM & using React in the Frontend. This service has a separate Database which stores the user info like _id i.e, unique identification of a user and Idea info like idea_id, idea_headline, idea_description, price, etc.

Cart Service enables Users to add Ideas and store them in cart , to Checkout before buying Ideas. This is positioned as a phase before paying/buying for an Idea. A User can add an available Idea to the cart and proceed to Checkout.

This service starts when a user clicks on the button (**Add to the Cart**) in/from another service and this service backend is made a **PUT** request from the frontend. Every Request is explained in detail in the sections below.

Users can look at all the ideas they added to the cart by going into "<domain.com>/cart" URL . where the frontend makes a **GET** request to get all the details and ideas of that particular user added to his cart. This data is coming from the backend of this service , which is stored in the MongoDB Database. Here a unique id (_id) of the user is sent as the parameter in the URL of the request made to the backend of the service. This is how a particular user is identified and his particular ideas are fetched and shown in the view / frontend.

Users can also remove an idea from the cart by clicking on the button (**Remove from Cart**), This button is there in the frontend of the cart service "<domain.com>/cart". For removing an idea from the cart the frontend makes a **PUT** request to the backend of the service. Here an unique id (_id) is along with the request in the body of the request to identify the particular idea in the database and of that particular user and delete it.

Users can clear there all the ideas present in there cart by clicking on the button (**Clear Cart**). This button in the front end makes a request to the backend API endpoint. This is a **GET** request with parameters _id , Which is an unique ID sent along as a parameter in the URL of the request.

This unique _id helps to find the particular user and his cart and all the ideas in his cart are cleared.

Backend :

Installation:

Clone this repository into your local

```
git clone https://github.com/Ravilochan/cart-service.git
```

Go to that directory

```
cd <directory-name>
```

Install all Dependencies for Node to run . You need to have Node , npm already installed in your computer to run this command

```
npm install
```

Run Service:

To Run this service your system or Local should have NodeJS Installed. This starts the service on http://localhost:7000 on your computer

```
npm start
```

or you can Start this service by command :

```
node api/index
```

Database Models:

Idea Model Schema:

_id: { type: ObjectId, required: true },

idea_owner: String,

idea_owner_name: String,

idea_genre: String,

idea_headline: String,

idea_description: String,

price: Number,

User Model Schema :

```
_id: {
   type: ObjectId,
   required: true,
 },
 cart: [ { type: ObjectId , ref: "Idea" } ]
```

Backend API Endpoints :

This Service has API endpoints at

```
/api/cart --> PUT Request

/api/uncart --> PUT Request

/api/cart/:id --> GET Request
/api/clearcart/:id --> GET Request
```

For /api/fav - POST Request

Data sending to Request in body should be like

```
{

"user":

     {

     "_id":"5d6ede6a0ba62570afcedd3b"
     },
"cart":
     {
     "_id":"5d6ede6a0ba62570afcedd32",
     "idea_owner": "String",
     "idea_owner_name": "String2",
     "idea_genre": "String3",
     "idea_headline": "String idea_name",
     "idea_description": "String idea_description",
     "price": 50
     }
}
```

Here the _id in user is the MongoDB UID / _id unique for every user. This should be 12 characters and unique and according to the rules of MongoDB _id. Here the _id in cart is the MongoDB UID / _id unique for every Idea. This should be 12 characters and unique and according to the rules of MongoDB _id. For adding an Idea to the cart - PUT Request: The user needs to send User details in "user" and Idea Details in "cart"

The values of the user and the idea are only stored when a particular user makes a request to add an item to his cart. If a user doesn't click on add to cart / doesn't interact with cart service in any manner until , then there will not be that user info in the Database of Cari Service. In the same way if any user didn't add an idea into cart anytime then that idea info is not stored in the database of cart service.

Response :

```
{
    "cart": [
        "5d6ede6a0ba62570afcedd32"
    ],
    "_id": "5d6ede6a0ba62570afcedd3b",
    "__v": 0
}
```

For /api/uncart - PUT Request

PUT data in Body of Request should be like

```
{
"user":
     {
     "_id":"5d6ede6a0ba62570afcedd3b"
     },
"cart":
     {
     "_id":"5d6ede6a0ba62570afcedd32"
     }
}
```

Here the _id in user is the MongoDB UID / _id unique for every user. This should be 12 characters and unique and according to the rules of MongoDB _id. Here the _id in cart is the MongoDB UID / _id unique for every Idea. This should be 12 characters and unique and according to the rules of MongoDB _id. For making this **PUT** un-cart Request (un-cart an Idea) : The user needs to send User details in "user" and only Idea _id in "**cart**"

Response :

```
{
    "cart": [],
    "_id": "5d6ede6a0ba62570afcedd3b",
    "__v": 0
}
```

To Get all cart ideas of a User :

/api/cart/:id -> GET Request

```
http:localhost:7000/api/cart/5d6ede6a0ba62570afcedd3b
```

here Params :id is the User's UID/_id from User collection. Here the _id in user is the MongoDB UID / _id unique for every user. This should be 12 characters and unique and according to the rules of MongoDB _id.

Response :

```
[
    {
        "_id": "5d6ede6a0ba62570afcedd38",
        "idea_owner": "String",
        "idea_owner_name": "String2",
        "idea_genre": "String3",
        "idea_headline": "String idea_name",
        "idea_description": "String idea_description",
        "price": 50
    },
    {
        "_id": "5d6ede6a0ba62570afcedd39",
        "idea_owner": "String",
        "idea_owner_name": "String2",
        "idea_genre": "String3",
        "idea_headline": "String idea_name",

        "idea_description": "String idea_description",

        "price": 50

    }
]
```

To Clear Cart of a User:

/api/clearcart/:id -> GET Request

```
http:localhost:7000/api/clearcart/5d6ede6a0ba62570afcedd3b
```

here Params :id is the User's UID/_id from User collection. Here the _id in user is the MongoDB UID / _id unique for every user. This should be 12 characters and unique and according to the rules of MongoDB _id.

This end point is used to clear all the available ideas present in the cart of a particular user. Here the :id which is _id of the user is used to identify the user and clear all his ideas in his cart.

Response :

```
{

    "cart": [],

    "_id": "5d6ede6a0ba62570afcedd3b",

    "__v": 0
}
```

Frontend:

This is Service which enables Users to add Ideas and store them in a cart. To Checkout ideas a phase before paying/buying for an Idea. A User can add an available Idea to the cart and proceed to Checkout.

This Service Frontend is bootstrapped with Create React App.

Installation

Clone this repository into your local

```
git clone https://github.com/Ravilochan/cart-service.git
```

Go to that directory

```
cd <directory-name>
```

Install all Dependencies for Node to run . You need to have Node , npm already installed in your computer to run this command

```
npm install
```

Running in development

In the project directory, To start the frontend

`yarn start` or `npm start`

Runs the app in the development mode.

Open http://localhost:3000 to view it in the browser

Build for Production

`yarn build` or `npm build`

Builds the app for production to the `build` folder.

It correctly bundles React in production mode and optimizes the build for the best performance.

The build is minified and the filenames include the hashes.

Your app is ready to be deployed!

Usage:

This service can list all the ideas stored in the cart of a user , remove an idea from the cart of a user, clear the cart and confirm ideas in the cart and checkout.

Listing all the ideas

`/cart` --> List of all the ideas in the cart of a user

Checkout Page

`/checkout` --> CheckOut page confirming your ideas before going into the payment service

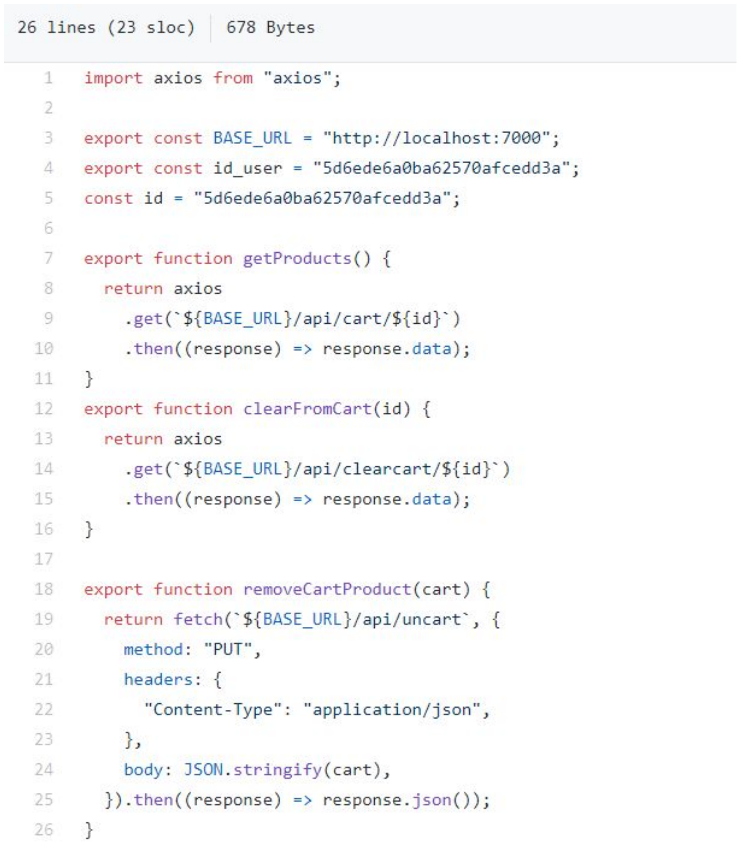
Root Page

`/` --> Home/ Root Page of the whole application.

Steps for Integration with other Services

1) Setup Redux in the cart service to access the info like authenticated user's unique id (_id) from the Redux store of the whole application.

   Here in the image is screenshot from file ***cart-service/frontend/src/api.js***



```
26 lines (23 sloc) | 678 Bytes
```

```javascript
import axios from "axios";

export const BASE_URL = "http://localhost:7000";
export const id_user = "5d6ede6a0ba62570afcedd3a";
const id = "5d6ede6a0ba62570afcedd3a";

export function getProducts() {
  return axios
    .get(`${BASE_URL}/api/cart/${id}`)
    .then((response) => response.data);
}
export function clearFromCart(id) {
  return axios
    .get(`${BASE_URL}/api/clearcart/${id}`)
    .then((response) => response.data);
}

export function removeCartProduct(cart) {
  return fetch(`${BASE_URL}/api/uncart`, {
    method: "PUT",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify(cart),
  }).then((response) => response.json());
}
```

Here we can see that "**id**" is initialized as a constant value , as during the development of this service the user unique id (_id) is unknown. This value is obtained from the USer Service while authenticating and authenticated users' unique id (_id) is stored in the redux store of the React+Redux Frontend application. Here in this cart service the i**d** should be obtained from redux store

2) Step and configure the BASE URL in the proxy of the react frontend of the cart service to appropriate URL . This is shown in the above image .

   File location : ***cart-service/frontend/src/api.js***

These are the steps to be altered while integrating this service with the rest of the application.