TRINITY FASHION

Software Engineering 14:332:452

Group 11 - Final Deliverable - 10/19/2022

Team Members:

Varsini Dhinakaran	vd299@scarletmail.rutgers.edu
Mehraaj Tawa	mt1055@scarletmail.rutgers.edu
Sohum Pohane	sdp195@scarletmail.rutgers.edu
Kavin Sakthivel	ks1605@rutgers.edu
Sahmi Abubakar	saa315@scarletmail.rutgers.edu
Jayanth Sivakumar	js2690@scarletmail.rutgers.edu
Viral Patel	vsp62@scarletmail.rutgers.edu
Aya Hourani	aah173@scarletmail.rutgers.edu
Rohan Patel	rjp289@rutgers.edu
Anurag Vattipalli	vv270@scarletmail.rutgers.edu

GitHub Repository Link:

https://github.com/Mehraaj/SoftwareEngineeringGroup11

Business Terminology Used

- User - any person using the site, either a visitor or a member
- Visitor - a user that does not have a registered account with the website, and has no access to member specific perks
- Member - a user that has a registered account with the website and has member specific perks
- Exclusive - Only available to members of the website
- Visitor Collections - Collections available to both visitors and members of the website (shirts and pants)
- Member Collections - Collections available to only members of the website
    - Every collection is available to a member
- Category - Referring to the type of clothing collection. This includes shirts, pants, shoes etc.
- Sub-Category - Referring to the type of clothing within a collection. Example: For shirts, this includes T-Shirts, Collared Shirts, Sweaters, Hoodies.

Business Rules and Policies

1. There are two types of users for our site, members and visitors. Both users can see the shirts and pants collections on our website. Members will have exclusive perks because they have created an account with our website.
    a. Perks include: Viewing additional product collections such as hats, socks, and shoes. Continuing previous shopping sessions (Items in the cart are restored once logged in again). Viewing past orders and past order details (specific products involved in a particular order. This is shown at the end of the demo but purely the backend, it was not implemented in the front end.)
    b. The reasoning for allowing members to see more products compared to a visitor is to create *brand loyalty*. By creating an account we have a way to contact the customer in the future if need be to market future releases or seasonal sales, enticing them to revisit our site. In addition to this, by showcasing more products/deals to the customer, they can get all of their clothing needs off of our brand alone, creating the brand loyalty we are hoping for. This ideology is found in real companies such as Costco and Ulta Beauty.
2. The company home page will show a few items from the available collections, as well as featured items

    a. For members, the site will also show a few items from member specific collections.
3. Users can select to view specific collections only. The options to view certain collections are found in the header.
4. Shipping is free, which is a benefit to all users of our site.
5. A user can order multiple of the same product if they wish, or different sizes or different colors of the same product. There is no limit on the number of a single product the user can order.
6. Items can be added to the cart after selecting a size and color. All items currently in the cart and their information can be viewed on a cart page, where users can then proceed to checkout.
    a. Upon logging out, visitors lose their cart information. Members cart information is stored and will be retained when logging in again.
7. Orders cannot be modified after they are placed. The idea is that unwanted items would be returned (future set of features, available to members & visitors have a 15 minute period to return items) and additional items will
8.
9.
10. be ordered separately.
11. After placing an order, a receipt is made available to be downloaded by the user for their own records. The specific order information of members is stored in the company's database.
    a. Members can view past order information, including specific items in each order and total price. (Backend only, shown in demo)
12. Sales tax is calculated based on what state the order is placed from.
13. Only valid card payments are accepted, and cannot make a transaction through paypal or other third party billers. Payments are verified by Square API to verify the card information and balance.
14. Members can log in with their registered username and password. Visitors can create an account by providing their email, username, and password. There is email verification to ensure that a valid email is provided.

15. When a member logs into the website, an API Key is generated and assigned to the member. API Keys are strings that have randomly generated letters and numbers. API Keys are valid for 15 minutes, and are passed as a cookie through the HTTP messages.
    a. API Keys are for user authentication. Instead of having the member id as a cookie, the API Key is used so that if our HTTP messages were intercepted, then an outsider would not be able to know which member is logged in, or which member id corresponds to what user.
    b. The API Key is checked to be valid before member specific operations, such as viewing member only collections. The API Key in the cookie is checked across all members to see if it has been assigned to a member and if it is also within 15 minutes since the API Key has been created (The API Key creation date is also stored in the database).
    c. If the API Key is valid, then the member specific operation will continue as expected, otherwise it will not present products to the visitor.
16. Users can search for products by name in the search bar, in case they know exactly what they would like. The search is done across all the products in the database.
17. Filters are only applicable on the products currently listed on the page being viewed. That is, you can filter by ascending or descending price, or by gender, on the current collection tab you are on. For example, if you are currently viewing shirts, you can apply the filters on to all shirts available.
18. An administrator can add or remove products to the database to update collections (not implemented in frontend and not shown in the demo, but the backend code is in the github link)

Brief Description of Project

Trinity Fashion is an online clothing store that customers can either shop on as a visitor or register as a member to get exclusive access to more clothing options and apparel among other perks. Trinity Fashion will sell the following types of clothing: shirts, pants, hats, shoes, and socks, each organized into categories that will make it easier for users to search for specific items or types of clothing. Desired items can be added to a user specific cart and when finished shopping, customers can checkout and purchase the items they have selected by providing credit card information. A downloadable pdf of their receipt will be available if they choose to download it.

Server Side of the Project

The server side of this project consists of a database and backend server. The users will connect to the backend server to see Trinity Fashion's website, cloth offerings, and service. The backend server will have access to the database server, and the amount of information presented to the user will vary depending if the user is a registered user or a visitor. If the user is a visitor, then the backend server will only showcase the Shirts and Pants collection, otherwise all collections will be shown to members. Members are verified through the use of a randomly generated API Key that is made during each shopping session and expires after 15 minutes. The following is a tentative list of all collections that will be stored in the database, followed by who will access said collections:

- Visitors
- Members (members access their own info)
- Orders (members)
- Shirts (visitors & members)
- Pants (visitors & members)
- Shoes (members)
- Hats (members)
- Socks(members)
- Cart (members)
- Product Catalog (Visitors & members)

List of Operations Supported on Collections

Sign In to Website

Members will be able to input their username and password to login to the website to access more products and view past orders. This operation adds the API Key to the members table. Can edit members collection.

Create New Account

Visitors will be able to input their email, username and password to make a new account on the website. Add's to members collection.

Add to Cart

Users will be able to select products available to them to add to their cart for purchase. Adds to cart collection for members in order to retain information in future shopping sessions.

Delete From Cart

Users will be able to remove items from their own cart. Deletes from cart collection for members.

Search Product Catalog

Users can view products relating to the page they have selected. This will be to view specific categories, products by gender, by ascending or descending price, or by a text search of the name. Views all product collections (shoes, shirts, pants, etc.). Does not edit/add/delete any collections in any way.

View Product Details

Users can click on individual products to display all attributes and descriptions of the product. Accesses collection of the product (shoes, shirts, pants, etc.). Does not edit/add/delete collections in any way.

Place Order

Users can purchase their cart, confirm the shipping and payment information and place the order. Adds to orders collection for members. Deletes from cart collection. Edits orders collection for members.

User Search Functions

Users will be able to search for their desired product by name of the product through a text search. Or, users can select a collection tab and see all products in that collection, and filter through gender and/or price through the use of a drop down menu. This is to give the user the best searching experience and search efficiently for their product.

Subgroup Partitioning

DATABASE TEAM:

Focused on creating the database and collections to assign attributes as well as creating queries for item selection for display or other functionalities. Will also assist the backend team with implementing the database queries as well as organization of the backend functions. Along with populating the database accordingly, will help in image assignment for each product. As a sub focus, the team will help with front end design for account sign in and new member registration. Handles all functions pertaining to pulling/adding information from the database. Once the database is established, will join the backend team and help in creating remaining API calls.

Members: Mehraaj Tawa, Varsini Dhinakaran, Rohan Patel

BACKEND TEAM:

Focused on creating search (operating on back end and sending to front end), checkout and downloadable receipt functionalities, as well as managing user and payment information. As

a sub focus, will help with designing the frontend to make it easy to work with and connect to the backend.
Members: Sohum Pohane, Aya Hourani, Jayanth Sivakumar

FRONTEND TEAM:

Focused on front end UI and design, creating web pages and formatting all user functionalities, as well as mapping all buttons and inputs for backend teams. Subteam is also responsible for company branding and aesthetic. As a sub focus, will work with the database team to establish what is required for operations.
Members: Viral Patel, Sahmi Abubakar, Anurag Vattipalli, Kavin Sakthivel

Use Cases

Casual Descriptions of Use Cases

Log In/Create Account – Actors: User (Initiating), Authenticator (Database, participating)
Description: When entering the website, on any page, select the login icon on the top right to log in. A member can log into the website if they already have an account by providing their username and password, or new visitors can create a new account with the website by selecting create an account after clicking on the login icon. By providing an email, username and password, visitors will become members and get access to member perks. This use case will lead to either a login error, a create new account page, or the website home page.
Related Requirements: REQ-1, REQ-2

Display Error/Display Homepage/Display Register page – Actor: User (Initiating), Authenticator (Database, participating)
Description: If a member entered a correct username and password,they will be redirected to the homepage of the website. If they entered incorrect information, they will encounter a login error and have to login again. If a user clicks create account, they will be redirected to a register page, and then redirected to the home page.
Related Requirements: REQ-1, REQ-2, REQ-12

View Items - Actor: User (Initiating)
Description: On the home page, based on if the site is in a visitor state or a member state, a few items from each collection will be displayed. After selecting a collection tab, collection specific items will be shown. A member can view all products from all collections, but a visitor can only view all products from the following collections: shirt, pants. This behavior is persistent throughout the site. If a visitor tries to view a member only collection, no items will be shown.
Related Requirements: REQ-3, REQ-8, REQ-11, REQ-13

View Item Description- Actor: User (Initiating)
Description: On any page, a user can view a specific item's description and attributes to see more information about the product by clicking on the product they are interested in. Visitors can only see products they have access to, but after logging in members will see all products and can choose any product to click and view.
Related Requirements: REQ-10

Browse By Filter- Actor: User
Description: A user can sort the list of products by collections such as pants, shorts, etc. This will be presented to them at the top of every page through the website header. A user can also sort the list of products they are currently viewing by price from lowest to highest or highest to lowest, and/or by gender. This is available through a drop down menu. This allows the user to account for their budget while browsing through products.
Related Requirements: REQ-7, REQ-14

Search - Actor: User
Description: A user can search for an item by name using a text field. This allows the user to look for a desired product easier. This can be accessed on any page through the search bar on the website header.
Related Requirements:REQ-9

Add to Cart - Actor: User
Description: Once the user has found a product they may like to purchase, they can select it to view the product details including sizes and colors. Then they can add the item to their cart after selecting a size and color. The user can view their current cart as well by selecting the cart icon on the top right and adjust the quantity of particular items.
Related Requirements: REQ- 5

Delete from Cart - Actor: User (Initiating)
Description: A user can delete an item from the cart if they decide they no longer want a particular item in their cart. This is done on the cart page by either removing the item or by reducing the quantity.
Related Requirements: REQ-15, REQ-5

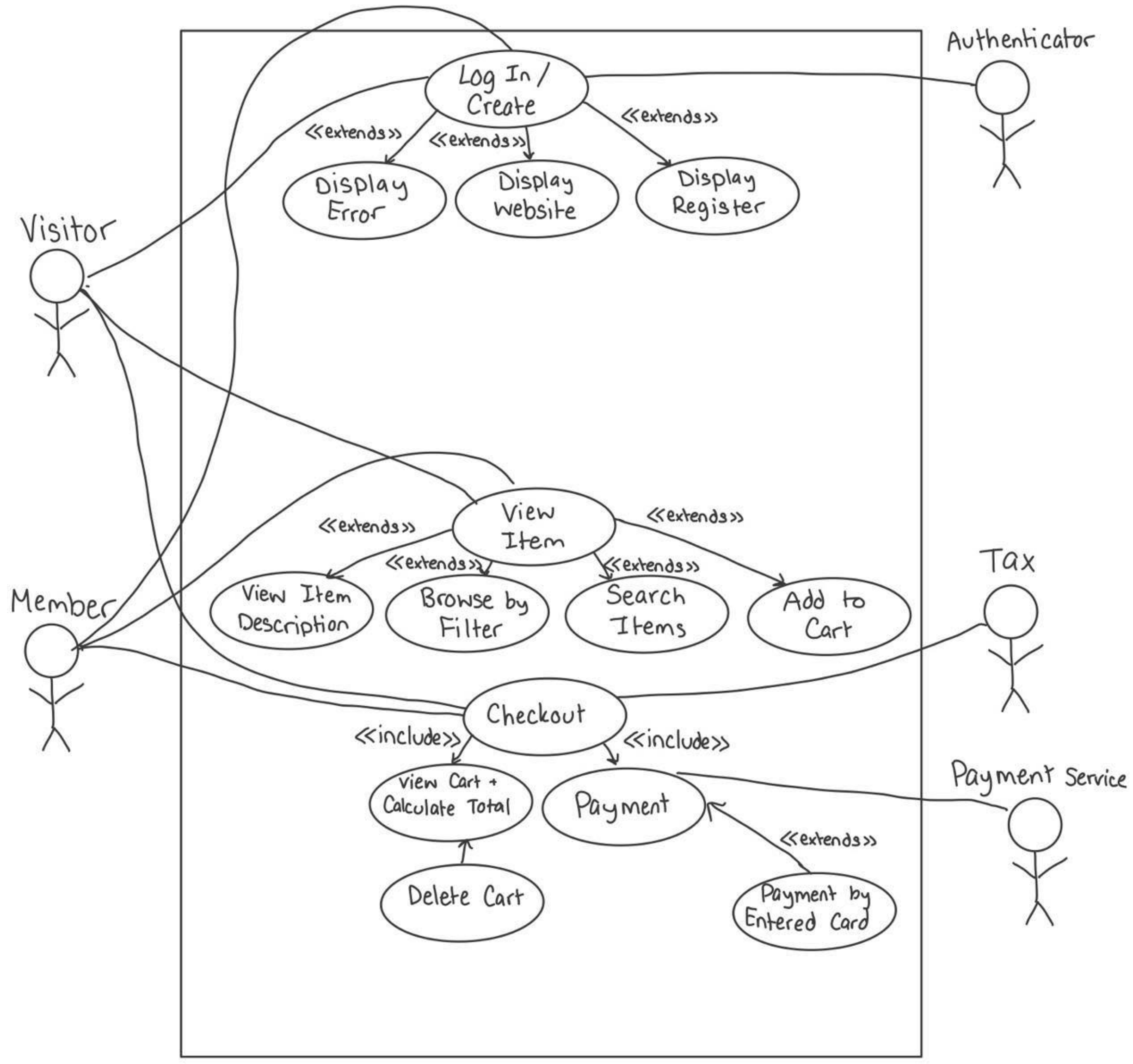
Checkout – Actor: User (Initiating)
Description: Once on the cart page, a user can check out their selected items in their cart by clicking checkout. Then, they will have to enter a shipping address and credit card information to purchase the items. The credit card will be verified through Square API and if successful, they will receive an order number and receipt of the transaction once checkout is complete. Otherwise they will have to enter card information again.

Related Requirements: REQ-18

View Cart Items & Calculate Total – Actors: User (Initiating), Taxation Provider (Participating)
Description: The user will be able to view their cart by selecting the cart page. They can then see the total price of their selected items. If they choose to edit the quantity of each item, then the price will be adjusted as well.
Related Requirements: REQ-16, REQ-17

Payment For Order – Actors: User (Initiating), Payment Service (Participating)
Description: Users will be able to edit the payment information on the checkout page to confirm their purchase. The credit card information will be verified by Square API.
Related Requirements: REQ-6

<u>Use Case Diagram</u>



Detailed Use Case Scenarios

Browse By Filter:

The user will first see the home page with a few products from the shirts and pants collection. If a member is logged in, then a few products from the socks, hats, and shoes collections will also be displayed. At the top of the page there will be options to select a specific collection to view. If a visitor selects a member specific collection, then no products will be displayed. Two filter drop downs will be shown on the collection specific pages, one for price and another for gender. To browse by filter they would click on either one of the two (or both) drop down menus. When a collection is selected, no filters will be applied by default and all the products under the selected collection will be displayed. If a price option (lowest price to highest price or highest price to lowest price), the products will be sorted accordingly. If a gender is

selected, only products for that gender will be displayed. If the user knows the specific name of the item they want, they can use the search bar on the top right of any page to search by name. The corresponding item will show.
Data Collections:

- Shirts collection
- Pants collection
- Shoes collection
- Hats collection
- Socks collection

Operations:

- Search Product Catalog

Business Rules: Visitors can only view certain collections, and this will be checked through the use of an API Key. Users can filter according to gender and price or search by name.

Log In:

When on any page of our website, users can select a login icon found on the top right of the website header. Selecting this button will redirect them to a page where they are presented with a username text box and a password text box. They can enter their username and password into the presented fields and click on a "Sign In" button, or they can select an option to create an account. The username and the password will be sent to the database server and if the information matches with the stored information, then a success signal will be sent back to the web server which will make the user get a message saying they successfully logged in. An API Key will be generated for that member's particular shopping session, which will expire in 15 minutes requiring them to log in again. Having a valid API Key puts the website in a logged in state, where members perks are enabled because before every operation the API Key will be verified allowing for member specific functionalities. If the login information is incorrect the user will have to enter their information again.
Data Collections:

- Member collection

Operations:

- Sign in to website & Generate API Key

Business Rules:

- Members are given exclusive access to collections of clothing, to promote website membership. Members will generate a temporary API Key when logging in.

Delete From Cart:

Once the user has a cart with items, they may decide they don't want to order one of the items and would want to remove it from their cart. Once on the cart screen, they will be able to select a

remove button for each item. Once the button is clicked, the corresponding item will be removed and won't be included when purchasing the order. The total will be adjusted accordingly.
Data Collections:

- Cart collection

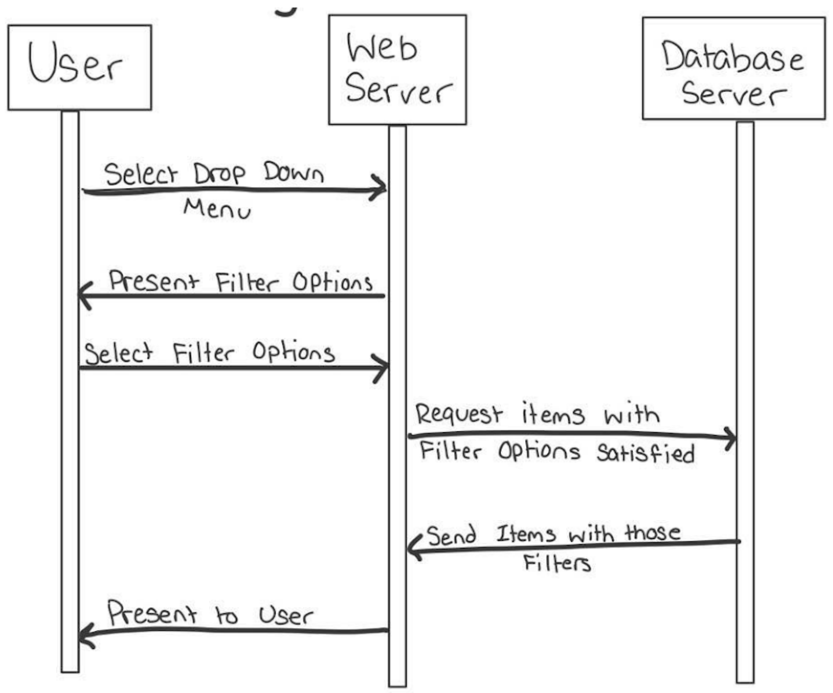
Operations:

- Delete from cart

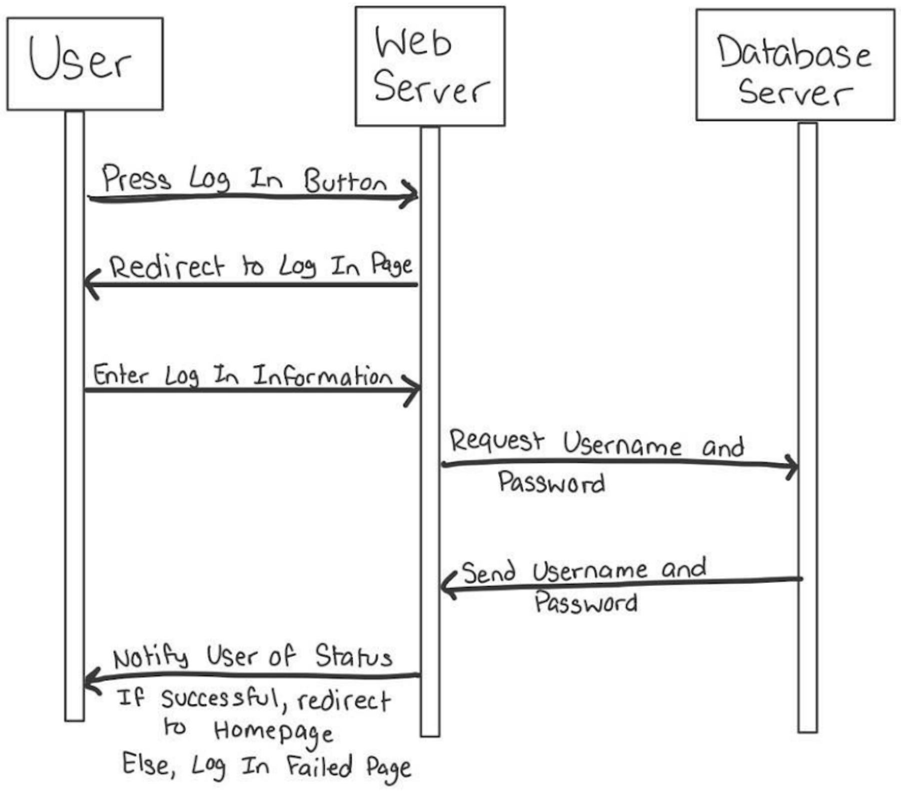
Business Rules: Member cart information will be stored for future shopping sessions.

System Sequence Diagrams:

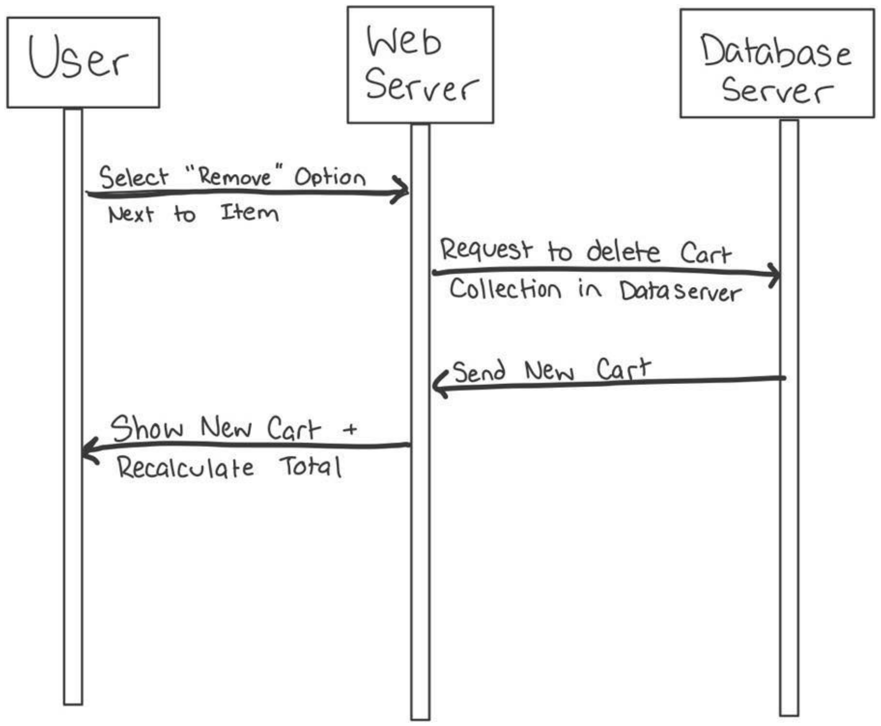
Browse by Filter



Log In



Delete From Cart


User
Web Server
Database Server
Select "Remove" Option Next to Item
Request to delete Cart Collection in Dataserver
Send New Cart
Show New Cart + Recalculate Total

Database Team Reqs

Identifier	User Story	Data Collections + Operations Associated
REQ- 1	As a Member, I will be able to sign in to my account on the website and generate an API Key for my current shopping session	○ Member collection
REQ- 2	As a visitor, I will be able to create an account for the website	○ Member collection ○ Operations Include: Add to members Collection
REQ- 3	As a user, I will be able to view the collections available to me, with the option to select between three filters being collection, gender and price.	○ All Collections ○ Operations: Sign in to website for members ○ Business Policies: Visitors do not have access to shoes, hats and socks collections because they do not have an API Key. Items can be filtered based on gender or price
REQ-4	As a user, I will be able to view (with different images) and select products of various sizes and colors.	○ Product Catalog collection ○ Operations Include: View product catalog. View product details ○ Business Policies: Users can add the same product with various sizes or colors.

Backend Team Reqs

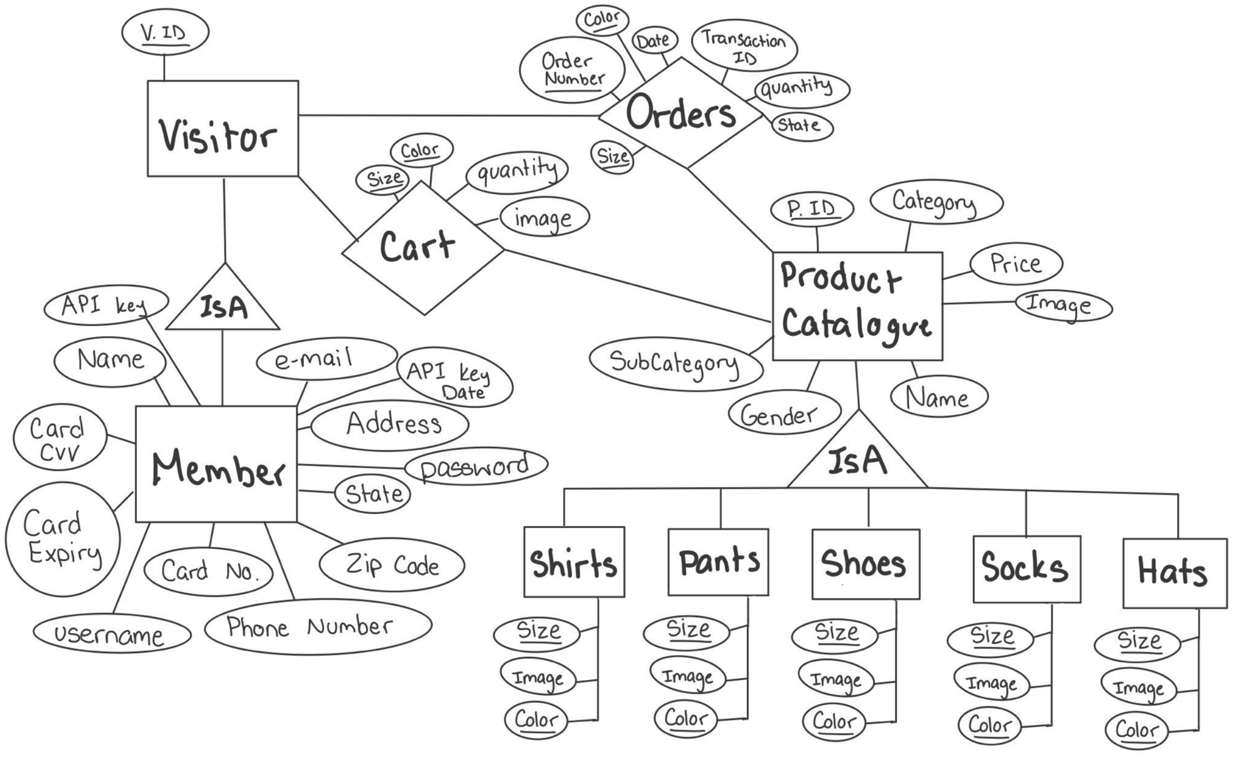
Identifier	User Story	Data Collections + Operations Associated
REQ- 5	As a user, I will be able to add and delete items to my cart. I will also be able to change the quantity of each item I would like.	○ Cart collection (if member) ○ Operations include: Add to cart. Delete from cart.
REQ- 6	As a user, I will be able to purchase the items in the cart and receive a downloadable pdf of my receipt.	○ Cart collection (if member) ○ Orders Collection (if member) ○ Operations include: Place order.

REQ- 7	As a user, I will be able to view items lowest to highest or vice versa (in terms of monetary value). Or, I will be able to view items based on their intended gender.	○ Shirts collection ○ Pants collection ○ Shoes collection (member) ○ Hats collection (member) ○ Socks collection (member) ○ Operations include: Filter product catalog
REQ- 8	As a visitor, I will have limited access to the products they can search for.	○ Shirts collection ○ Pants collection ○ Operations include: Search product catalog, view product details ○ Business policies: Registered members have exclusive access to collections of clothing.
REQ- 9	As a user, I can search for the name of a product, granted the name/collection is available to them (member vs visitor collections).	○ Shirts collection ○ Pants collection ○ Shoes collection (member) ○ Hats collection (member) ○ Socks collection (member) ○ Operations include: Search product catalog, sign into website ○ Business policies: Registered members have exclusive access to collections of clothing.
REQ- 10	As a user, I can click on a product and access its descriptions and attributes.	○ Shirts collection ○ Pants collection ○ Shoes collection (member) ○ Hats collection (member) ○ Socks collection (member) ○ Operations include: View product details ○ Business policies: Users can select color and size. Users can select an item to view details and add to cart

Frontend Team Reqs

Identifier	User Story	Data Collections + Operations Associated
Req- 11	As a user, accurate data/images should be displayed neatly.	● Taking tabular data as an input from the database and converting it as an output in HTML, JS, and CSS.
Req- 12	As a user, I have access to a main menu at the top of the screen that allows for ease of navigation.	● Operations include: view items by specific categories ● Gender of clothing ● Type of clothing ● Collection status ● Search bar ● View Cart
Req- 13	As a user, I observe consistent themes throughout all webpages.	● Similar CSS pages should be used across the website
Req- 14	As a user, I can view collections sorted by gender and clothing type in an organized and easily navigable manner	● All clothing articles ● Operations: Each gender has categories of clothing articles the user can select and search through.
Req- 15	As a user, I am able to remove/change the quantity of items in my cart.	● All collections, Cart collection (member) ● Operations: remove items from cart, change quantity of item in cart
Req- 16	As a user, I am able to view a checkout summary that lists the number of items in cart, tax/delivery fees, and total, allowing me to easily view costs of my purchase.	● Cart Collection (member) ● Operations: display costs
Req- 17	As a user, I am able to view my cart in an organized webpage	● Cart Collection (member) ● Operations: view cart
Req- 18	As a user, I am able to receive an order number and receipt once I have made a purchase. This benefits me by allowing me to keep track of my purchased order.	● Cart Collection ● Operations: user receives an email with confirmation and order number/tracking numbers

ER Diagram



Schema

```
drop database if exists TrinityFashion;
create database TrinityFashion;
use TrinityFashion;

create table Visitor(
VID varchar(50),
primary key(VID)
);

create table Member(
VID varchar(50),
email varchar(100),
username varchar (50),
password varchar(50),
Name varchar(50),
Address varchar(50),
State varchar(50),
ZIP varchar(50),
Phone varchar(50),
CreditCardNo varchar(50),
CreditCardCVV varchar(50),
CreditCardExpiry varchar(50),
APIKey varchar(50),
APIKeyDate varchar(100),
primary key(VID),
foreign key (VID) references Visitor(VID)
);

create table ProductCatalog(
PID int,
Category enum('shirts', 'pants', 'shoes', 'hats', 'socks'),
Name varchar(50),
Price decimal(9,2),
SubCategory varchar(50),
gender varchar(50),
image varchar(150),
primary key(PID)
);

create table Shirts(
PID int,
Size varchar(50),
color varchar(50),
image varchar(150),
primary key(PID, Size, color),
foreign key(PID) references ProductCatalog(PID)
);

create table Pants(
PID int,
Size varchar(50),
color varchar(50),
image varchar(150),
primary key(PID, Size, color),
foreign key(PID) references ProductCatalog(PID)
);

create table Shoes(
PID int,
Size varchar(50),
color varchar(50),
image varchar(150),
primary key(PID, Size, color),
foreign key(PID) references ProductCatalog(PID)
);
```

```
create table Socks(
PID int,
Size varchar(50),
color varchar(50),
image varchar(150),
primary key(PID, Size, color),
foreign key(PID) references ProductCatalog(PID)
);

create table Hats(
PID int,
Size varchar(50),
color varchar(50),
image varchar(150),
primary key(PID, Size, color),
foreign key(PID) references ProductCatalog(PID)
);


create table Cart(
VID varchar(50),
PID int,
quantity int,
Size varchar(50),
color varchar(50),
image varchar(150),
primary key (VID, PID, Size, color),
foreign key(VID) references Visitor(VID),
foreign key(PID) references ProductCatalog(PID)
);

create table Orders(
VID varchar(50),
PID int,
quantity int,
color varchar(50),
Size varchar(50),
orderNumber varchar(100),
state varchar(50),
date varchar(100),
transactionId varchar(100),
primary key(VID, PID, color, Size, orderNumber),
foreign key(VID) references Visitor(VID),
foreign key(PID) references productCatalog(PID)
) ;
```

Rest APIs

Operation	Resource URI	Method	Inputs	Returns	Response Code
Create New Account (Member)	/users/member	Post	Email, username, password	none	Success - 200, 201, 202 Failure - 400, 403, 409
Create Visitor	/users/visitor	Post	Email, username, password	none	Success - 200, 201, 202 Failure - 400, 403, 409
Sign in	/users	Get	Username, password	API Key	Success- 200 Failure - 406, 408, 400
Update Cart	/orders/cart	Post	Cart items and item details, API Key	none	Success - 200, 201, 202 Failure - 400, 401, 403, 409
Get Cart	/orders/cart	Get	API Key	Cart items and item details	Success - 200, 201, 202 Failure - 400, 401, 403, 409
Search Product Catalog	/products/ productcatalog?{filters}	Get	API Key, gender, price, or name filter (all optional inputs)	Returns list of products that match search	Success - 200, 202 Failure - 400, 501
View Product Details	/products/ productcatalog/:id	Get	Id of product	Returns details of product	Success - 200, 202 Failure- 404, 501, 502
Place Order	/orders/	Post	Cart items and item	none	Success - 200, 201, 202

			details		Failure - 400, 403, 409, 502
Fetch Orders	/orders/	Get	API Key	All orders made by member	Success - 200, 201, 202 Failure - 404, 502
Fetch Order details	/orders/details/:id	Get	API Key	All items in a specific order	Success - 200, 201, 202 Failure - 404, 502
Create Payment	/orders/payment/:order Number	Post	Order Number	None	Success - 200, 201, 202 Failure - 404, 502
Create Receipt	/orders/receipts/:order Number	Post	Order Number	Receipt	Success - 200, 201, 202 Failure - 404, 502
State Tax	/orders/tax/:state	get	state	Tax rate	Success - 200, 201, 202 Failure - 404, 502

General response types:
2xx - successful type
3xx- redirection
4xx - client error
5xx- server error
Specific error codes can be found at:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#client_error_responses

Deliverable 3

How To Run Server Side Code:

Note: At this point in time, we are currently running off of the expressRestructured Branch.

After downloading our project folder, to run the server-side code in NodeJS, type the following command when in our project folder:

Npm i

This is to receive all of the necessary libraries. You only need to do this the first time to download the necessary libraries.

You will also need to create a .env file that contains the host, username and password to the database server that we are using.

Then, run the command

npm run start

This will shortcut into a .json file named package.json that will look for the term start and run a different command, that being the following:

nodemon backend/src/index.js

Note that index.js is our starting point for the backend server that contains the Express routers and middlewares that we have developed for the project. The env file is a file describing machine specific parameters and the package.json and package-lock.json files are the two that describe the routing to the scripts and dependencies of the project described above.

<u>How to Run Front End Code</u>

In a separate terminal, please run the following command. Make sure to use a different port than the server:

python -m http.server [port]

Then in a browser, type localhost:[port]/[path]

Note that the homepage path is homepage.html

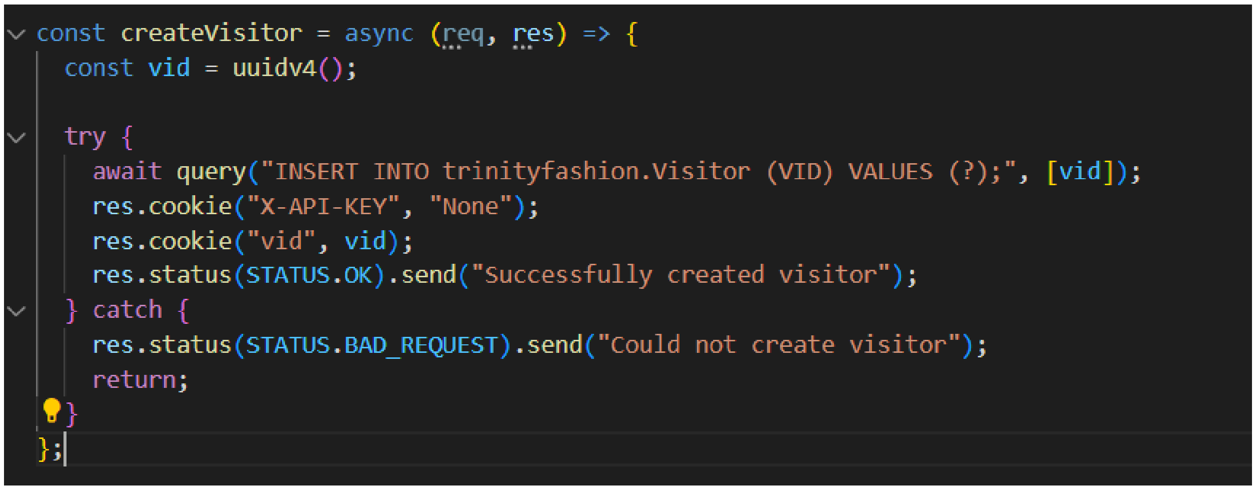
<u>API Calls</u>

Create Visitor Function: (Create new item in collection)

Server Endpoint:

POST http://localhost:8000/users/visitor

Function Definition: (X-API-KEY is for security reasons)



```
const createVisitor = async (req, res) => {
  const vid = uuidv4();

  try {
    await query("INSERT INTO trinityfashion.Visitor (VID) VALUES (?);", [vid]);
    res.cookie("X-API-KEY", "None");
    res.cookie("vid", vid);
    res.status(STATUS.OK).send("Successfully created visitor");
  } catch {
    res.status(STATUS.BAD_REQUEST).send("Could not create visitor");
    return;
  }
};
```

Request

```
POST /users/visitor HTTP/1.1
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: 6747923e-e857-46e1-9749-ece77d572e38
Host: localhost:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Length: 0
```

Response

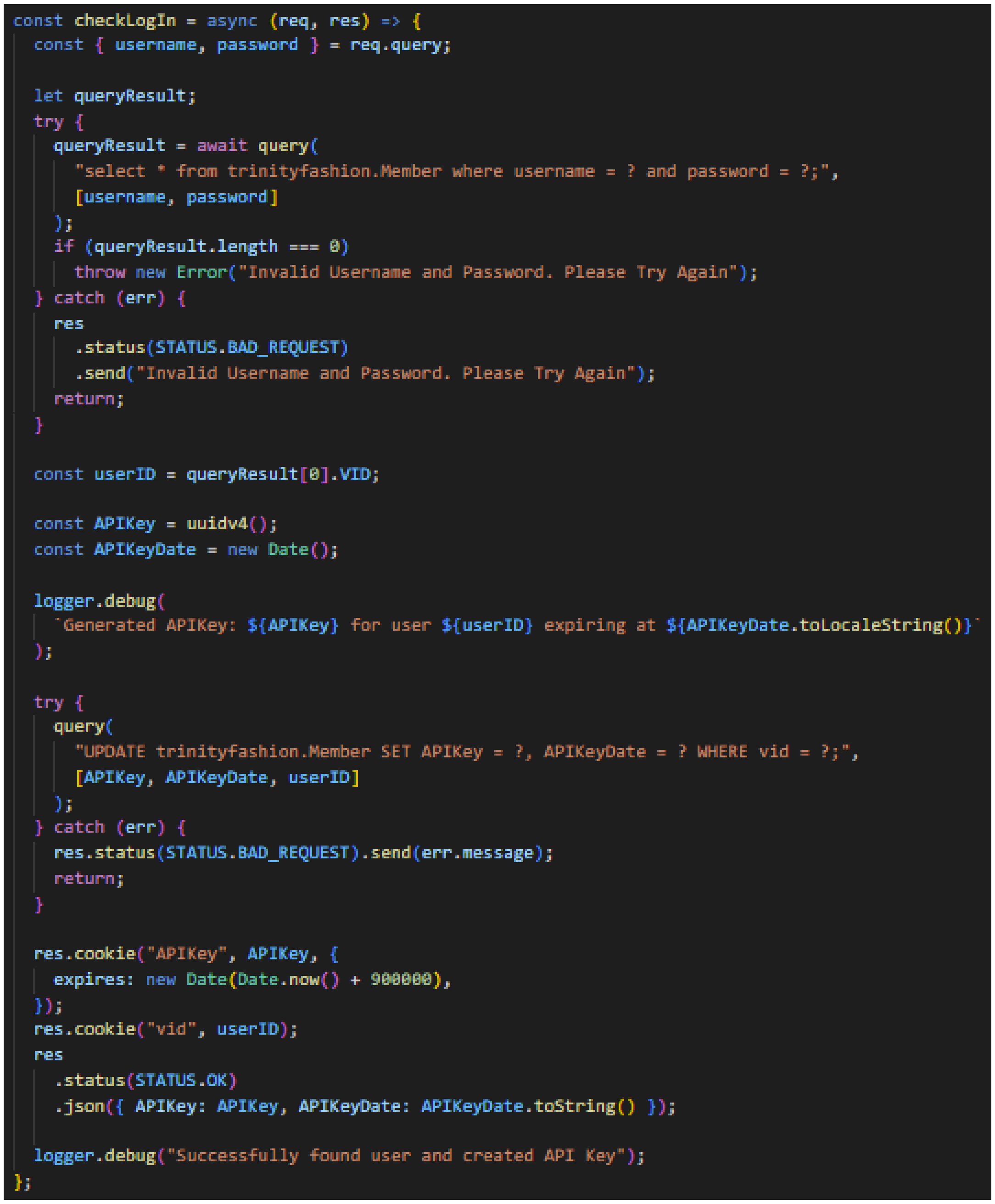
```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
Set-Cookie: APIKey=None; Path=/
Set-Cookie: vid=74a1b46c-3d75-4055-828e-ffdaac4c6d3c; Path=/
Content-Type: text/html; charset=utf-8
Content-Length: 28
ETag: W/"1c-8vRp4w/QQC5T88ns0DFpV7fRIXM"
Date: Wed, 07 Dec 2022 23:38:14 GMT
Connection: keep-alive
Keep-Alive: timeout=5

Successfully created visitor
```

Check Log In: (Query and display the content of a collection)

Server Endpoint:

GET http://localhost:8000/users

Function Definition:



```
const checkLogIn = async (req, res) => {
  const { username, password } = req.query;

  let queryResult;
  try {
    queryResult = await query(
      "select * from trinityfashion.Member where username = ? and password = ?;",
      [username, password]
    );
    if (queryResult.length === 0)
      throw new Error("Invalid Username and Password. Please Try Again");
  } catch (err) {
    res
      .status(STATUS.BAD_REQUEST)
      .send("Invalid Username and Password. Please Try Again");
    return;
  }

  const userID = queryResult[0].VID;

  const APIKey = uuidv4();
  const APIKeyDate = new Date();

  logger.debug(
    `Generated APIKey: ${APIKey} for user ${userID} expiring at ${APIKeyDate.toLocaleString()}`
  );

  try {
    query(
      "UPDATE trinityfashion.Member SET APIKey = ?, APIKeyDate = ? WHERE vid = ?;",
      [APIKey, APIKeyDate, userID]
    );
  } catch (err) {
    res.status(STATUS.BAD_REQUEST).send(err.message);
    return;
  }

  res.cookie("APIKey", APIKey, {
    expires: new Date(Date.now() + 900000),
  });
  res.cookie("vid", userID);
  res
    .status(STATUS.OK)
    .json({ APIKey: APIKey, APIKeyDate: APIKeyDate.toString() });

  logger.debug("Successfully found user and created API Key");
};
```

Request

```
GET /users?username=username&password=password HTTP/1.1
User-Agent: PostmanRuntime/7.29.2
```

```
Accept: */*
Postman-Token: 3242cc22-a83f-4166-9820-3213278fcfdc
Host: localhost:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

Response

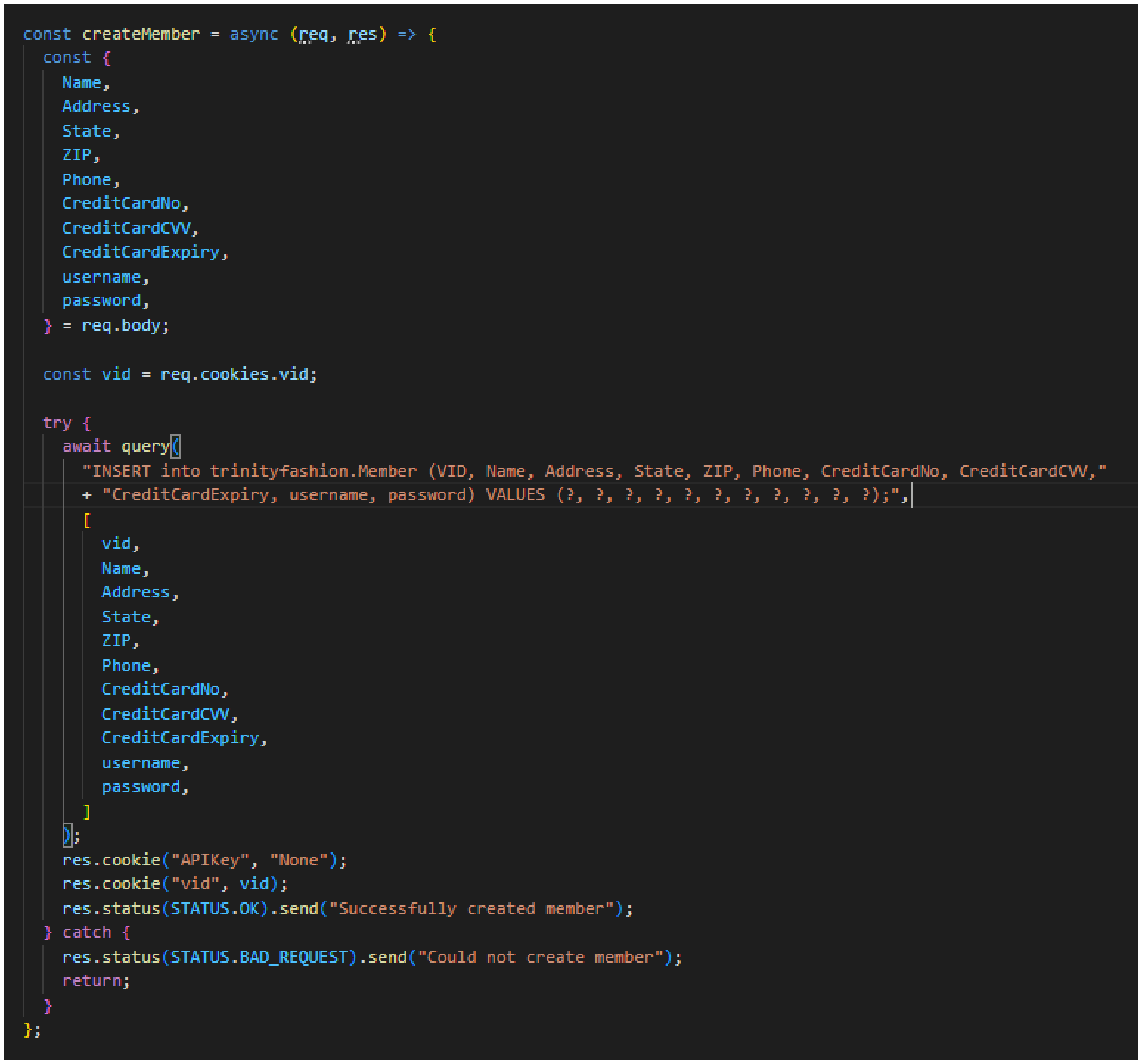
```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
Set-Cookie: APIKey=30b7133f-5d9b-413f-8dcc-a7daae31d1eb; Path=/; Expires=Wed, 07 Dec
2022 23:58:41 GMT
Set-Cookie: vid=001; Path=/
Content-Type: application/json; charset=utf-8
Content-Length: 122
ETag: W/"7a-DZDbZxJVT/h0Rk+0B1x7SZfDDfk"
Date: Wed, 07 Dec 2022 23:43:41 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"APIKey":"30b7133f-5d9b-413f-8dcc-a7daae31d1eb","APIKeyDate":"Wed Dec 07 2022
18:43:40 GMT-0500 (Eastern Standard Time)"}
```

Create Member (Create new item in collection and link an item in one collection to an item in another collection)

Server Endpoint:

POST http://localhost:8000/user/member

Function Definition:



```
const createMember = async (req, res) => {
  const {
    Name,
    Address,
    State,
    ZIP,
    Phone,
    CreditCardNo,
    CreditCardCVV,
    CreditCardExpiry,
    username,
    password,
  } = req.body;

  const vid = req.cookies.vid;

  try {
    await query(
      "INSERT into trinityfashion.Member (VID, Name, Address, State, ZIP, Phone, CreditCardNo, CreditCardCVV,"
      + "CreditCardExpiry, username, password) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);",
      [
        vid,
        Name,
        Address,
        State,
        ZIP,
        Phone,
        CreditCardNo,
        CreditCardCVV,
        CreditCardExpiry,
        username,
        password,
      ]
    );
    res.cookie("APIKey", "None");
    res.cookie("vid", vid);
    res.status(STATUS.OK).send("Successfully created member");
  } catch {
    res.status(STATUS.BAD_REQUEST).send("Could not create member");
    return;
  }
};
```

Request

```
POST /users/member HTTP/1.1
Content-Type: application/json
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: 62dca968-dbf2-4baa-be7a-cd12f2fb95a1
Host: localhost:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Length: 305
Cookie: APIKey=None; vid=df4429bc-79cd-42c6-9f73-665aa7289b92

{
"Name": "Joe Scott",
"Address": "256 Lincoln Ave",
"State": "CA",
"ZIP": "97234",
"Phone": "6850450284",
"CreditCardNo": "7670274078634658",
"CreditCardCVV": "839",
"CreditCardExpiry": "06/23",
"username": "joescott123",
"password": "joeisthebest123"
}
```

Response

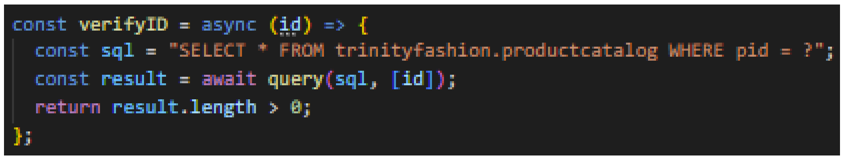
```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
Set-Cookie: APIKey=None; Path=/
Set-Cookie: vid=df4429bc-79cd-42c6-9f73-665aa7289b92; Path=/
Content-Type: text/html; charset=utf-8
Content-Length: 27
ETag: W/"1b-pyMv4m6lanEObF3JKNf1zC5JqKM"
Date: Thu, 08 Dec 2022 00:11:02 GMT
Connection: keep-alive
Keep-Alive: timeout=5

Successfully created member
```

Get All Products: (Query and display the content of a collection)

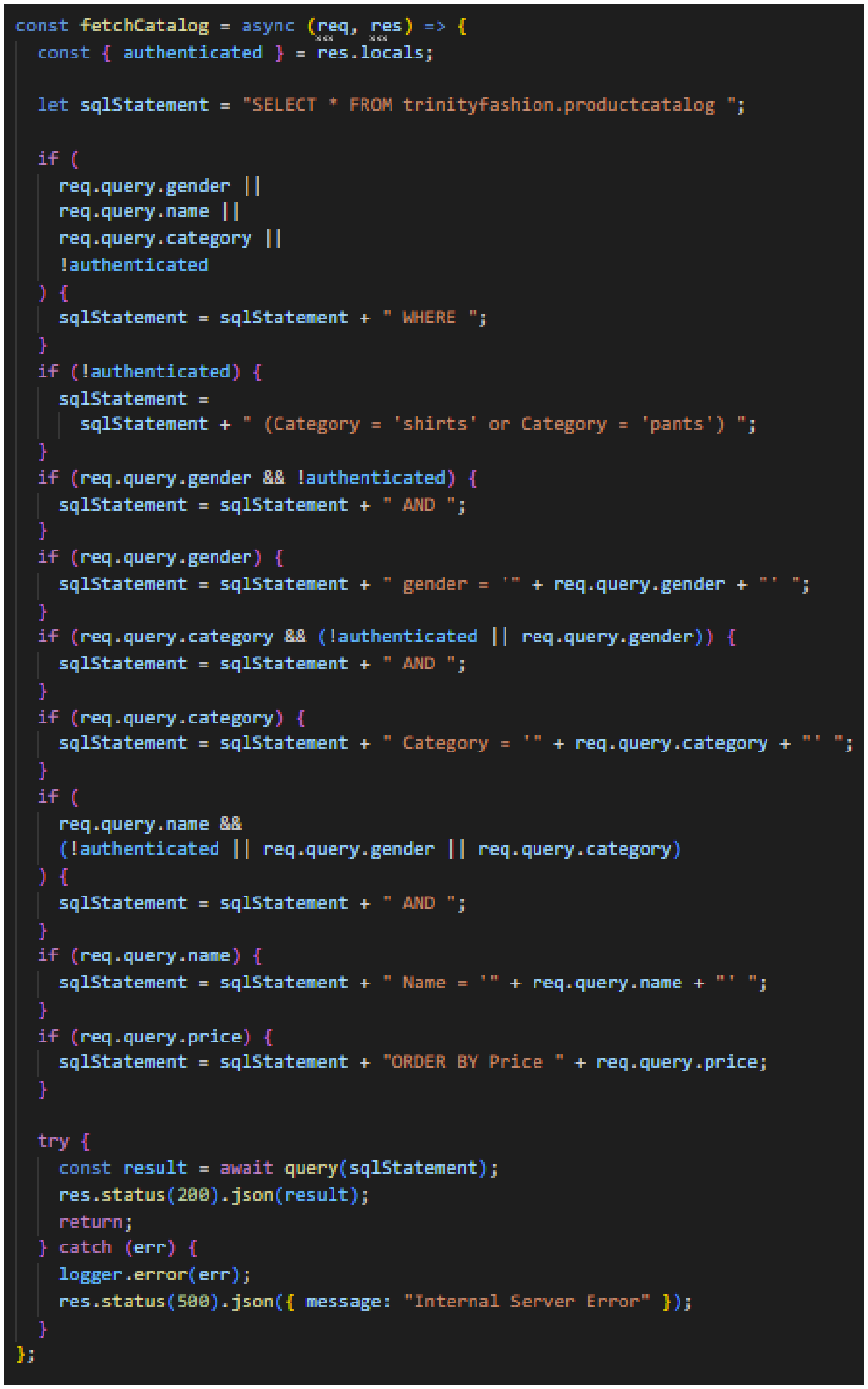
Server Endpoint:

GET http://localhost:8000/products/productCatalog

Function Definition:



```
const verifyID = async (id) => {
  const sql = "SELECT * FROM trinityfashion.productcatalog WHERE pid = ?";
  const result = await query(sql, [id]);
  return result.length > 0;
};
```



```
const fetchCatalog = async (req, res) => {
  const { authenticated } = res.locals;

  let sqlStatement = "SELECT * FROM trinityfashion.productcatalog ";

  if (
    req.query.gender ||
    req.query.name ||
    req.query.category ||
    !authenticated
  ) {
    sqlStatement = sqlStatement + " WHERE ";
  }
  if (!authenticated) {
    sqlStatement =
      sqlStatement + " (Category = 'shirts' or Category = 'pants') ";
  }
  if (req.query.gender && !authenticated) {
    sqlStatement = sqlStatement + " AND ";
  }
  if (req.query.gender) {
    sqlStatement = sqlStatement + " gender = '" + req.query.gender + "' ";
  }
  if (req.query.category && (!authenticated || req.query.gender)) {
    sqlStatement = sqlStatement + " AND ";
  }
  if (req.query.category) {
    sqlStatement = sqlStatement + " Category = '" + req.query.category + "' ";
  }
  if (
    req.query.name &&
    (!authenticated || req.query.gender || req.query.category)
  ) {
    sqlStatement = sqlStatement + " AND ";
  }
  if (req.query.name) {
    sqlStatement = sqlStatement + " Name = '" + req.query.name + "' ";
  }
  if (req.query.price) {
    sqlStatement = sqlStatement + "ORDER BY Price " + req.query.price;
  }

  try {
    const result = await query(sqlStatement);
    res.status(200).json(result);
    return;
  } catch (err) {
    logger.error(err);
    res.status(500).json({ message: "Internal Server Error" });
  }
};
```

RESULT DIFFERENT WHEN NOT LOGGED IN VS LOGGED IN - SHOWN BELOW

<u>NOT LOGGED IN:</u>

Request

```
GET /products/productcatalog HTTP/1.1
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: b11f00a0-a816-4aab-b053-c000bd6fa52b
Host: localhost:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: APIKey=None; vid=eb29e588-e007-42ae-b784-e32d31ad9e05
```

Response

```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Content-Length: 1468
ETag: W/"5bc-+dukFCCiNOf6+f1WSxjzhR+39O8"
Date: Wed, 07 Dec 2022 23:49:16 GMT
Connection: keep-alive
Keep-Alive: timeout=5
```

```
[{"PID":101,"Category":"shirts","Name":"Plain
Tee","Price":"13.99","SubCategory":"T-Shirt","gender":"male","image":"./productImages/
blackPlainTeeM.jpg"},{"PID":102,"Category":"pants","Name":"Ruler Straight
Jeans","Price":"44.99","SubCategory":"Jeans","gender":"male","image":"./productImages/
blueRulerStraightJeanM.jpg"},{"PID":106,"Category":"shirts","Name":"Plain
Vee","Price":"11.99","SubCategory":"V-Neck","gender":"male","image":"./productImages/b
lackPlainVeeM.jpg"},{"PID":107,"Category":"pants","Name":"LazyWear
SweatPants","Price":"24.99","SubCategory":"Sweatpants","gender":"male","image":"./prod
uctImages/blueLWSweatPantsM.jpg"},{"PID":111,"Category":"shirts","Name":"Cotton Crew
Neck Crop
Top","Price":"21.99","SubCategory":"Crew-Neck","gender":"female","image":"./productIma
ges/blackCotCrewW.jpg"},{"PID":112,"Category":"pants","Name":"BootCut
Jeans","Price":"19.99","SubCategory":"jeans","gender":"female","image":"./productImage
s/blueBootCutJeanW.jpg"},{"PID":115,"Category":"pants","Name":"Performance
Leggings","Price":"29.99","SubCategory":"Leggings","gender":"female","image":"./produc
tImages/bluePerformanceLegW.jpg"},{"PID":116,"Category":"shirts","Name":"Wooly
Sweater","Price":"21.99","SubCategory":"Sweater","gender":"female","image":"./productI
mages/whiteWoolSwetrW.jpg"},{"PID":117,"Category":"pants","Name":"Wool Blended Pleat
Pants","Price":"89.99","SubCategory":"pants","gender":"female","image":"./productImage
s/greenWoolPleatPantsW.jpg"}]
```

LOGGED IN (Without Filters):

Request

```
GET /products/productcatalog HTTP/1.1
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: 6b02f8c4-c944-4096-b372-48c208826168
Host: localhost:8000
Accept-Encoding: gzip, deflate, br
```

```
Connection: keep-alive
Cookie: APIKey=1944a8c3-be7d-491c-a520-c00b1fbca27f; vid=001
```

Response

```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Content-Length: 3268
ETag: W/"cc4-mdVtOtBRGOvFG3aQmmwXE+LU0dw"
Date: Wed, 07 Dec 2022 23:51:50 GMT
Connection: keep-alive
Keep-Alive: timeout=5
```

```
[{"PID":101,"Category":"shirts","Name":"Plain
Tee","Price":"13.99","SubCategory":"T-Shirt","gender":"male","image":"./productImages/
blackPlainTeeM.jpg"},{"PID":102,"Category":"pants","Name":"Ruler Straight
Jeans","Price":"44.99","SubCategory":"Jeans","gender":"male","image":"./productImages/
blueRulerStraightJeanM.jpg"},{"PID":103,"Category":"shoes","Name":"Zoomer
7","Price":"29.99","SubCategory":"Sneakers","gender":"male","image":"./productImages/w
hiteZoomer7M.jpg"},{"PID":104,"Category":"hats","Name":"Solid Bold
Cap","Price":"19.99","SubCategory":"Baseball
Cap","gender":"male","image":"./productImages/blackBaseBCapM.jpg"},{"PID":105,"Categor
y":"socks","Name":"Faded Mellow Work
Sock","Price":"12.99","SubCategory":"socks","gender":"male","image":"./productImages/g
reenFMSockM.jpg"},{"PID":106,"Category":"shirts","Name":"Plain
Vee","Price":"11.99","SubCategory":"V-Neck","gender":"male","image":"./productImages/b
lackPlainVeeM.jpg"},{"PID":107,"Category":"pants","Name":"LazyWear
SweatPants","Price":"24.99","SubCategory":"Sweatpants","gender":"male","image":"./prod
uctImages/blueLWSweatPantsM.jpg"},{"PID":108,"Category":"shoes","Name":"Steel Toe
Black Smoke
Boots","Price":"39.99","SubCategory":"Boots","gender":"male","image":"./productImages/
blackSteelToeBlackSmokeBootsM.jpg"},{"PID":109,"Category":"hats","Name":"Straw
Fidora","Price":"21.99","SubCategory":"Fedora","gender":"male","image":"./productImage
s/whiteStrawFidoraM.jpg"},{"PID":110,"Category":"socks","Name":"Athletic
Socks","Price":"19.99","SubCategory":"socks","gender":"male","image":"./productImages/
blueAthleticSockM.jpg"},{"PID":111,"Category":"shirts","Name":"Cotton Crew Neck Crop
Top","Price":"21.99","SubCategory":"Crew-Neck","gender":"female","image":"./productIma
ges/blackCotCrewW.jpg"},{"PID":112,"Category":"pants","Name":"BootCut
Jeans","Price":"19.99","SubCategory":"jeans","gender":"female","image":"./productImage
s/blueBootCutJeanW.jpg"},{"PID":113,"Category":"shoes","Name":"Running/Tennis
Shoes","Price":"39.99","SubCategory":"sneakers","gender":"female","image":"./productIm
ages/whiteRunTenSnekrsW.jpg"},{"PID":114,"Category":"hats","Name":"Wool Pom Pom
hat","Price":"14.99","SubCategory":"hat","gender":"female","image":"./productImages/wh
itePomPomHatW.jpg"},{"PID":115,"Category":"pants","Name":"Performance
Leggings","Price":"29.99","SubCategory":"Leggings","gender":"female","image":"./produc
tImages/bluePerformanceLegW.jpg"},{"PID":116,"Category":"shirts","Name":"Wooly
Sweater","Price":"21.99","SubCategory":"Sweater","gender":"female","image":"./productI
mages/whiteWoolSwetrW.jpg"},{"PID":117,"Category":"pants","Name":"Wool Blended Pleat
Pants","Price":"89.99","SubCategory":"pants","gender":"female","image":"./productImage
s/greenWoolPleatPantsW.jpg"},{"PID":118,"Category":"shoes","Name":"Pointed Toe
Heels","Price":"119.99","SubCategory":"Heels","gender":"female","image":"./productImag
es/whitePointedToeHeelsW.jpg"},{"PID":119,"Category":"hats","Name":"Sherpa Bucket
Hat","Price":"26.99","SubCategory":"BucketHat","gender":"female","image":"./productIma
```

ges/whiteSherpaBucketHatW.jpg"},{"PID":120,"Category":"socks","Name":"Fuzzy Festive Snowflake Sock","Price":"4.99","SubCategory":"socks","gender":"female","image":"./productImages/redFuzzyFestiveSockW.jpg"}]

LOGGED IN (With Category Filter):

```
GET /products/productcatalog?category=shirts HTTP/1.1
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: deee2833-bd60-49c1-a3eb-4d0ef8ade3b7
Host: localhost:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: APIKey=46a892bc-1846-4ebe-a02a-c4e334e850ea; vid=001
```

Request

```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Content-Length: 634
ETag: W/"27a-nYqak0dmcCUUsXZpLOJcNif0YU4"
Date: Thu, 08 Dec 2022 01:30:48 GMT
Connection: keep-alive
Keep-Alive: timeout=5
```

Response

[{"PID":101,"Category":"shirts","Name":"Plain Tee","Price":"13.99","SubCategory":"T-Shirt","gender":"male","image":"./productImages/blackPlainTeeM.jpg"},{"PID":106,"Category":"shirts","Name":"Plain Vee","Price":"11.99","SubCategory":"V-Neck","gender":"male","image":"./productImages/blackPlainVeeM.jpg"},{"PID":111,"Category":"shirts","Name":"Cotton Crew Neck Crop Top","Price":"21.99","SubCategory":"Crew-Neck","gender":"female","image":"./productImages/blackCotCrewW.jpg"},{"PID":116,"Category":"shirts","Name":"Wooly Sweater","Price":"21.99","SubCategory":"Sweater","gender":"female","image":"./productImages/whiteWoolSwetrW.jpg"}]

LOGGED IN (With Category and Gender Filter):

Request

```
GET /products/productcatalog?category=shirts&gender=male HTTP/1.1
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: 12dde6d4-34a5-4717-a8b7-6ebef63cbdaa
Host: localhost:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: APIKey=46a892bc-1846-4ebe-a02a-c4e334e850ea; vid=001
```

Response

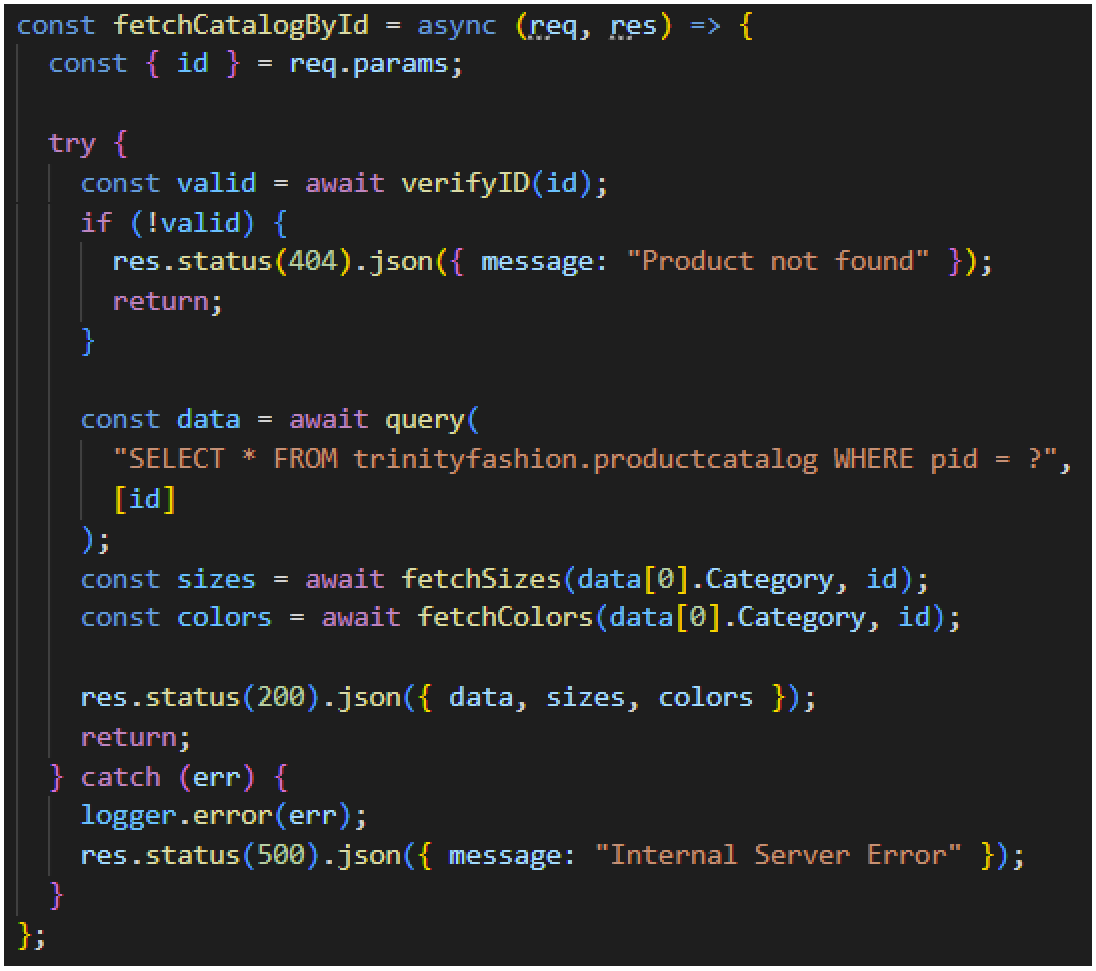
```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Content-Length: 304
ETag: W/"130-wFi+OU99vHwb03d8aX9Zh3+Lgqs"
Date: Thu, 08 Dec 2022 01:34:39 GMT
Connection: keep-alive
Keep-Alive: timeout=5

[{"PID":101,"Category":"shirts","Name":"Plain
Tee","Price":"13.99","SubCategory":"T-Shirt","gender":"male","image":"./productImages/
blackPlainTeeM.jpg"},{"PID":106,"Category":"shirts","Name":"Plain
Vee","Price":"11.99","SubCategory":"V-Neck","gender":"male","image":"./productImages/b
lackPlainVeeM.jpg"}]
```

Get Product Details: (Query and display the content of a collection)

Server Endpoint:

GET http://localhost:8000/products/productCatalog/:id

Function Definition:



```
const fetchCatalogById = async (req, res) => {
  const { id } = req.params;

  try {
    const valid = await verifyID(id);
    if (!valid) {
      res.status(404).json({ message: "Product not found" });
      return;
    }

    const data = await query(
      "SELECT * FROM trinityfashion.productcatalog WHERE pid = ?",
      [id]
    );
    const sizes = await fetchSizes(data[0].Category, id);
    const colors = await fetchColors(data[0].Category, id);

    res.status(200).json({ data, sizes, colors });
    return;
  } catch (err) {
    logger.error(err);
    res.status(500).json({ message: "Internal Server Error" });
  }
};
```

Request

```
GET /products/productcatalog/101 HTTP/1.1
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: 1de266e1-78bf-496b-ba55-c723af80503e
Host: localhost:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

Response

```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Content-Length: 230
ETag: W/"e6-ZbTpaYED4DQWdWcsozgFBmr74GI"
Date: Thu, 08 Dec 2022 00:19:12 GMT
Connection: keep-alive
Keep-Alive: timeout=5
```

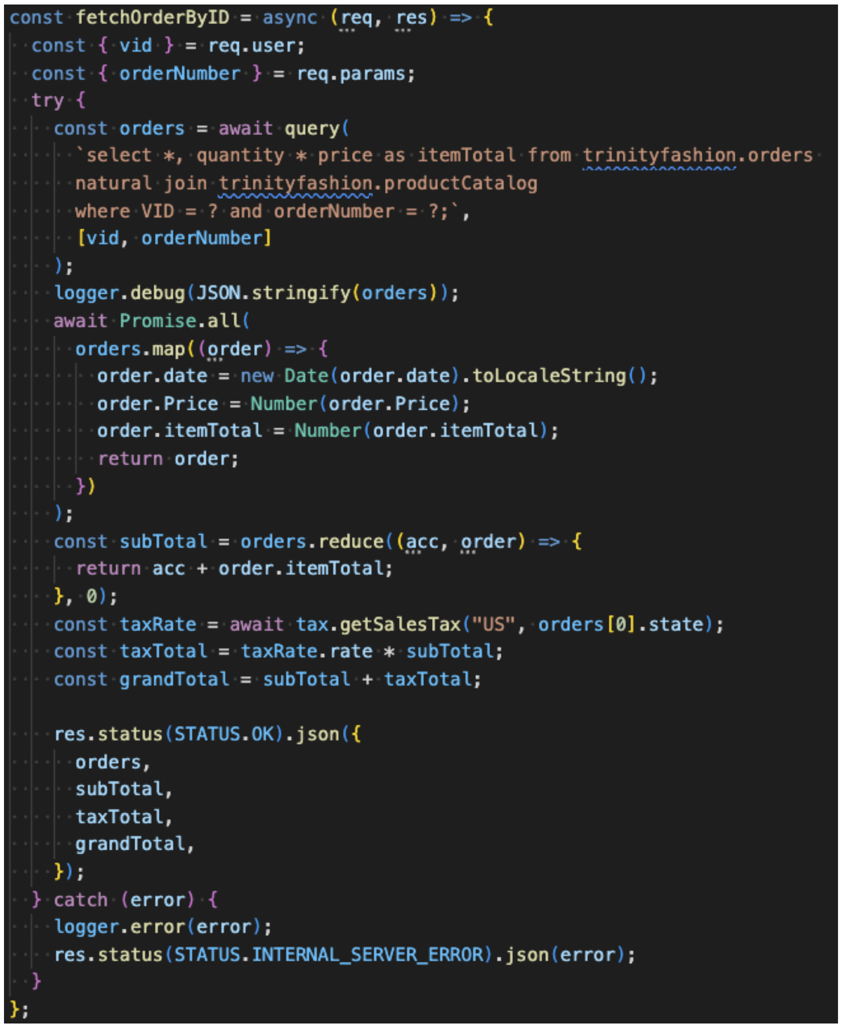
{"data":[{"PID":101,"Category":"shirts","Name":"Plain Tee","Price":"13.99","SubCategory":"T-Shirt","gender":"male","image":"./productImages/blackPlainTeeM.jpg"}],"sizes":["L","M","S","XL"],"colors":["black","green","red","white"]}

Fetch Order Details: (Query and display the content of a collection and linking collections)

Server Endpoint:

GET http://localhost:8000/orders/details/:id

Function Definition:



```
const fetchOrderByID = async (req, res) => {
  const { vid } = req.user;
  const { orderNumber } = req.params;
  try {
    const orders = await query(
      `select *, quantity * price as itemTotal from trinityfashion.orders
      natural join trinityfashion.productCatalog
      where VID = ? and orderNumber = ?;`,
      [vid, orderNumber]
    );
    logger.debug(JSON.stringify(orders));
    await Promise.all(
      orders.map((order) => {
        order.date = new Date(order.date).toLocaleString();
        order.Price = Number(order.Price);
        order.itemTotal = Number(order.itemTotal);
        return order;
      })
    );
    const subTotal = orders.reduce((acc, order) => {
      return acc + order.itemTotal;
    }, 0);
    const taxRate = await tax.getSalesTax("US", orders[0].state);
    const taxTotal = taxRate.rate * subTotal;
    const grandTotal = subTotal + taxTotal;

    res.status(STATUS.OK).json({
      orders,
      subTotal,
      taxTotal,
      grandTotal,
    });
  } catch (error) {
    logger.error(error);
    res.status(STATUS.INTERNAL_SERVER_ERROR).json(error);
  }
};
```

Request:

```
GET /orders/details/048159eb-a0bc-4905-aa9c-b3186913ec6e HTTP/1.1
User-Agent: PostmanRuntime/7.29.2
```

```
Accept: /
Postman-Token: 0b45613e-ee47-489c-856b-4c0fdbeba44d
Host: localhost:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: APIKey=9a9a838d-1800-4b26-9305-036e56a974e3;
cart=%5B%7B%22VID%22%3A%22001%22%2C%22PID%22%3A101%2C%22Size%22%3A%22S%22%2C%22color%2
2%3A%22white%22%2C%22quantity%22%3A1%7D%2C%7B%22VID%22%3A%22001%22%2C%22PID%22%3A102%2
C%22Size%22%3A%22L%22%2C%22color%22%3A%22green%22%2C%22quantity%22%3A1%7D%2C%7B%22VID%
22%3A%22001%22%2C%22PID%22%3A105%2C%22Size%22%3A%22S%22%2C%22color%22%3A%22black%22%2C
%22quantity%22%3A1%7D%5D
```

Response:

```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Content-Length: 1058
ETag: W/"422-YSd6xjFKo5PUIO0TwdqwMie9CDI"
Date: Thu, 08 Dec 2022 04:20:32 GMT
Connection: keep-alive
Keep-Alive: timeout=5
```

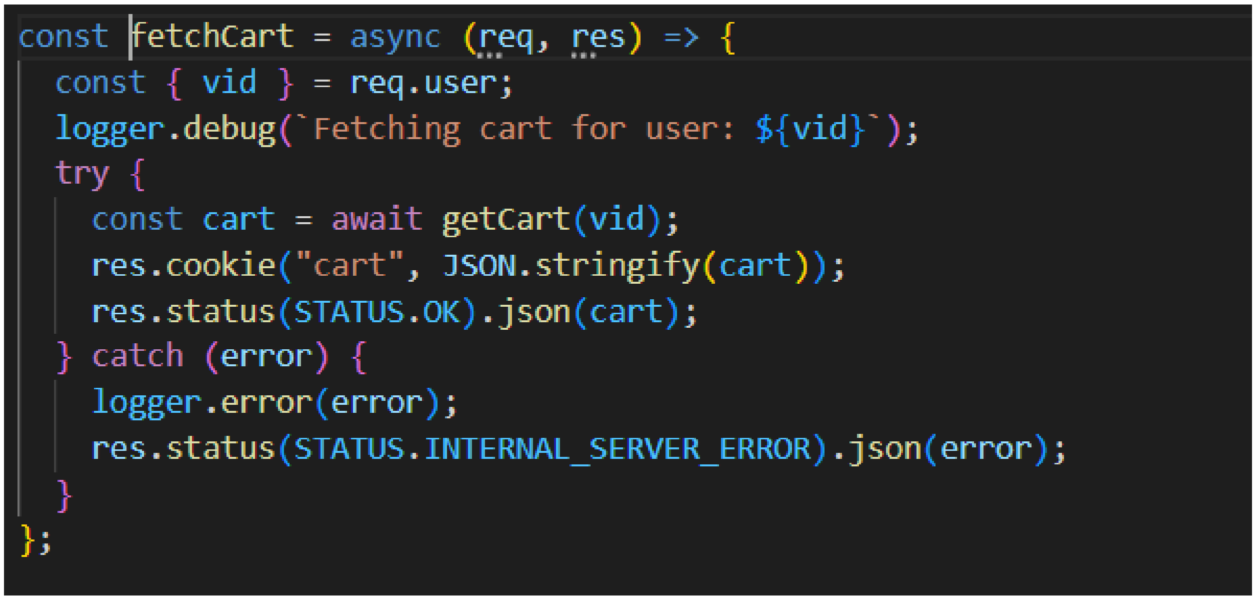
```
{"orders":[{"PID":101,"VID":"001","color":"white","Size":"S","quantity":1,"orderNumber
":"048159eb-a0bc-4905-aa9c-b3186913ec6e","state":"NJ","date":"12/7/2022, 10:36:24
PM","Category":"shirts","Name":"Plain
Tee","Price":13.99,"SubCategory":"T-Shirt","gender":"male","image":"./productImages/bl
ackPlainTeeM.jpg","itemTotal":13.99},{"PID":102,"VID":"001","color":"green","Size":"L"
,"quantity":1,"orderNumber":"048159eb-a0bc-4905-aa9c-b3186913ec6e","state":"NJ","date"
:"12/7/2022, 10:36:24 PM","Category":"pants","Name":"Ruler Straight
Jeans","Price":44.99,"SubCategory":"Jeans","gender":"male","image":"./productImages/bl
ueRulerStraightJeanM.jpg","itemTotal":44.99},{"PID":105,"VID":"001","color":"black","S
ize":"S","quantity":1,"orderNumber":"048159eb-a0bc-4905-aa9c-b3186913ec6e","state":"NJ
","date":"12/7/2022, 10:36:24 PM","Category":"socks","Name":"Faded Mellow Work
Sock","Price":12.99,"SubCategory":"socks","gender":"male","image":"./productImages/gre
enFMSockM.jpg","itemTotal":12.99}],"subTotal":71.97,"taxTotal":5.0379000000000005,"gra
ndTotal":77.0079}
```

Get Cart For Member (Query and display the content of a collection)

Server Endpoint:

GET http://localhost:8000/orders/cart

Function Definition:



```
const fetchCart = async (req, res) => {
  const { vid } = req.user;
  logger.debug(`Fetching cart for user: ${vid}`);
  try {
    const cart = await getCart(vid);
    res.cookie("cart", JSON.stringify(cart));
    res.status(STATUS.OK).json(cart);
  } catch (error) {
    logger.error(error);
    res.status(STATUS.INTERNAL_SERVER_ERROR).json(error);
  }
};
```

Response:

```
GET /orders/cart HTTP/1.1
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: 9aa1b1d8-bf5e-45ef-b93c-d1a4e67803b3
Host: localhost:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: APIKey=94e2967e-334d-432b-bade-f5a0872bb38b;
cart=%5B%7B%22VID%22%3A%22001%22%2C%22PID%22%3A101%2C%22Size%22%3A%22S%22%2C%22color%2
2%3A%22white%22%2C%22quantity%22%3A1%7D%2C%7B%22VID%22%3A%22001%22%2C%22PID%22%3A102%2
C%22Size%22%3A%22L%22%2C%22color%22%3A%22green%22%2C%22quantity%22%3A1%7D%2C%7B%22VID%
22%3A%22001%22%2C%22PID%22%3A105%2C%22Size%22%3A%22S%22%2C%22color%22%3A%22black%22%2C
%22quantity%22%3A1%7D%5D; vid=1
```

Request:

```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
Set-Cookie:
cart=%5B%7B%22VID%22%3A%22001%22%2C%22PID%22%3A101%2C%22Size%22%3A%22S%22%2C%22color%2
2%3A%22white%22%2C%22quantity%22%3A1%7D%2C%7B%22VID%22%3A%22001%22%2C%22PID%22%3A102%2
C%22Size%22%3A%22L%22%2C%22color%22%3A%22green%22%2C%22quantity%22%3A1%7D%2C%7B%22VID%
```

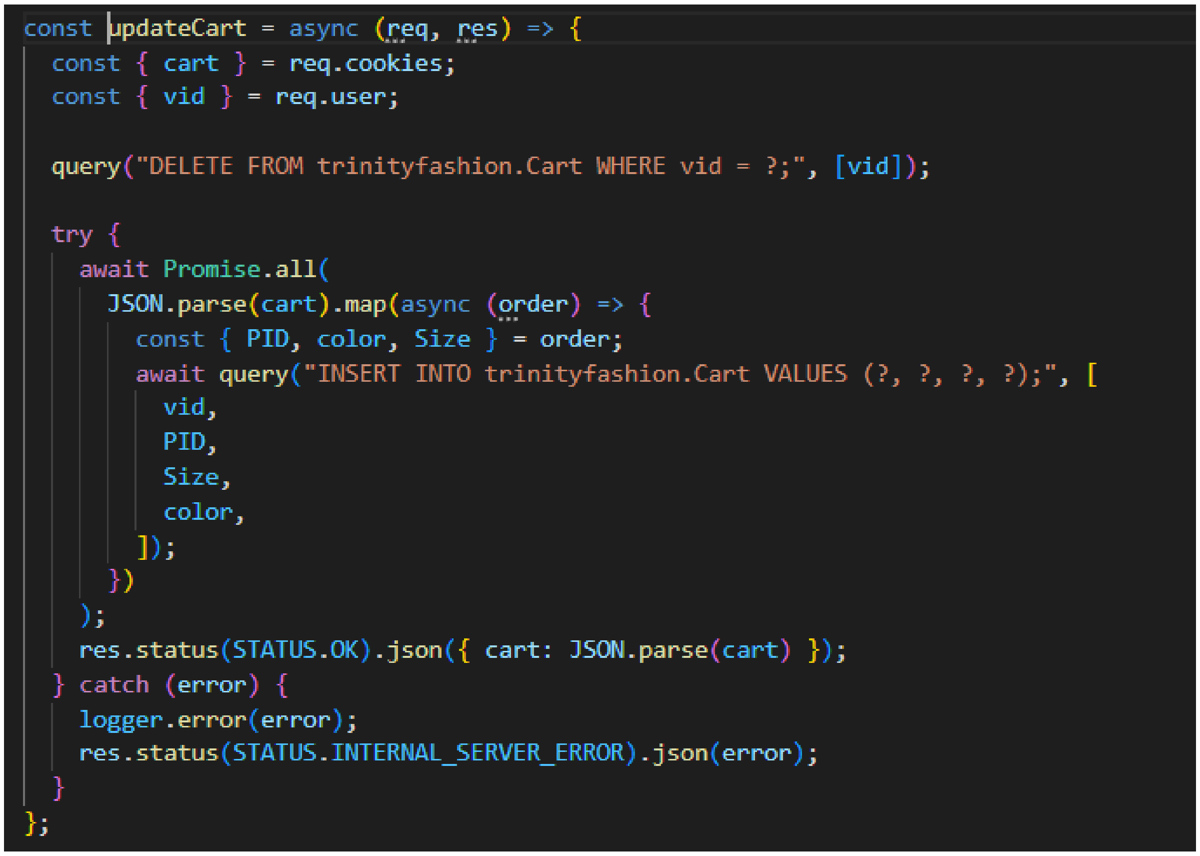
```
22%3A%22001%22%2C%22PID%22%3A105%2C%22Size%22%3A%22S%22%2C%22color%22%3A%22black%22%2C
%22quantity%22%3A1%7D%5D; Path=/
Content-Type: application/json; charset=utf-8
Content-Length: 193
ETag: W/"c1-Fwxnnorp7H2qsJhTXpik5Sw5Ejs"
Date: Thu, 08 Dec 2022 04:41:44 GMT
Connection: keep-alive
Keep-Alive: timeout=5

[{"VID":"001","PID":101,"Size":"S","color":"white","quantity":1},{"VID":"001","PID":10
2,"Size":"L","color":"green","quantity":1},{"VID":"001","PID":105,"Size":"S","color":"
black","quantity":1}]
```

Post Cart For Member (Link an item in one collection to an item in another collection)

Server Endpoint:

POST http://localhost:8000/orders/cart

Function Definition:



```
const updateCart = async (req, res) => {
  const { cart } = req.cookies;
  const { vid } = req.user;

  query("DELETE FROM trinityfashion.Cart WHERE vid = ?;", [vid]);

  try {
    await Promise.all(
      JSON.parse(cart).map(async (order) => {
        const { PID, color, Size } = order;
        await query("INSERT INTO trinityfashion.Cart VALUES (?, ?, ?, ?);", [
          vid,
          PID,
          Size,
          color,
        ]);
      })
    );
    res.status(STATUS.OK).json({ cart: JSON.parse(cart) });
  } catch (error) {
    logger.error(error);
    res.status(STATUS.INTERNAL_SERVER_ERROR).json(error);
  }
};
```

Response:

```
POST /orders/cart HTTP/1.1
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: fad3594e-703f-427e-adee-b872ec1b415b
Host: localhost:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: APIKey=94e2967e-334d-432b-bade-f5a0872bb38b;
cart=%5B%7B%22VID%22%3A%22001%22%2C%22PID%22%3A101%2C%22Size%22%3A%22S%22%2C%22color%2
2%3A%22white%22%2C%22quantity%22%3A1%7D%2C%7B%22VID%22%3A%22001%22%2C%22PID%22%3A102%2
C%22Size%22%3A%22L%22%2C%22color%22%3A%22green%22%2C%22quantity%22%3A1%7D%2C%7B%22VID%
22%3A%22001%22%2C%22PID%22%3A105%2C%22Size%22%3A%22S%22%2C%22color%22%3A%22black%22%2C
%22quantity%22%3A1%7D%5D; vid=1
Content-Length: 0
```

Response:

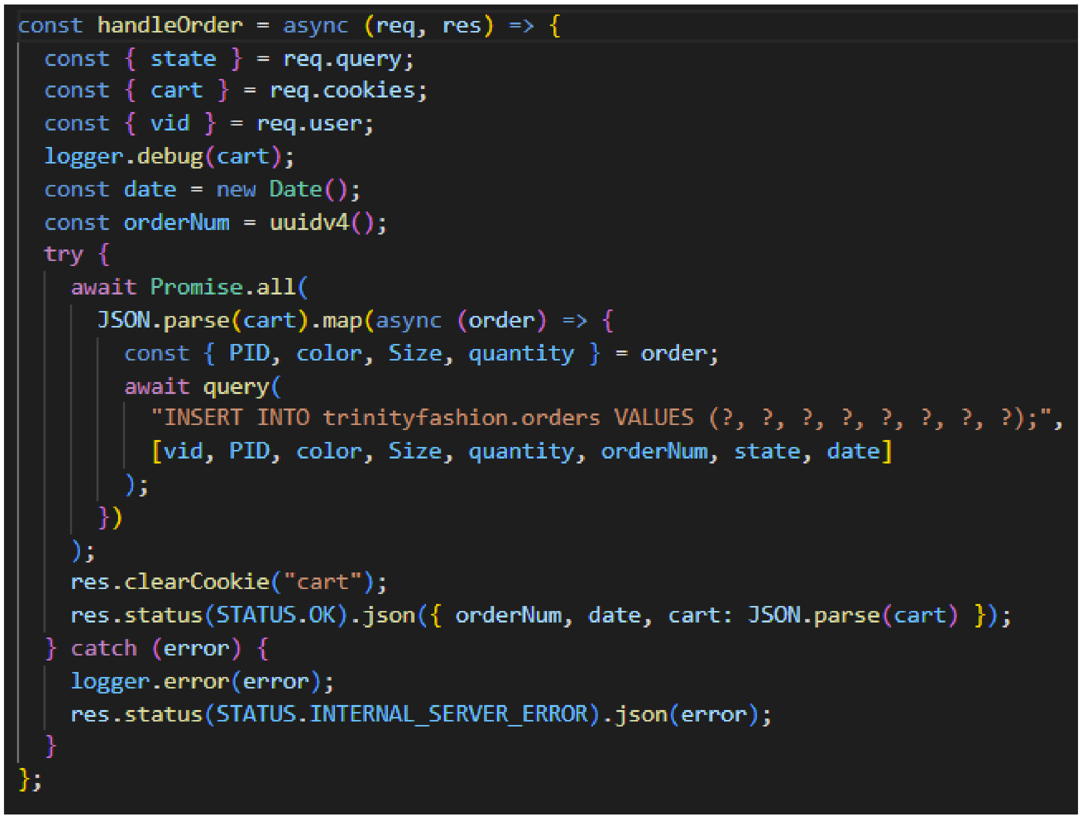
```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Content-Length: 202
ETag: W/"ca-ZOTTrkHS/RFUcl4a9rUQj8TG/B8"
Date: Thu, 08 Dec 2022 04:47:38 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"cart":[{"VID":"001","PID":101,"Size":"S","color":"white","quantity":1},{"VID":"001",
"PID":102,"Size":"L","color":"green","quantity":1},{"VID":"001","PID":105,"Size":"S","
color":"black","quantity":1}]}
```

Create Order (Add to the content of a collection and linking collections)

Server Endpoint:

POST http://localhost:8000/orders

Function Definition:



```
const handleOrder = async (req, res) => {
  const { state } = req.query;
  const { cart } = req.cookies;
  const { vid } = req.user;
  logger.debug(cart);
  const date = new Date();
  const orderNum = uuidv4();
  try {
    await Promise.all(
      JSON.parse(cart).map(async (order) => {
        const { PID, color, Size, quantity } = order;
        await query(
          "INSERT INTO trinityfashion.orders VALUES (?, ?, ?, ?, ?, ?, ?, ?);",
          [vid, PID, color, Size, quantity, orderNum, state, date]
        );
      })
    );
    res.clearCookie("cart");
    res.status(STATUS.OK).json({ orderNum, date, cart: JSON.parse(cart) });
  } catch (error) {
    logger.error(error);
    res.status(STATUS.INTERNAL_SERVER_ERROR).json(error);
  }
};
```

Request:

```
POST /orders?state=NJ HTTP/1.1
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: 7164a0e3-3c20-41ee-bd86-2b5e02c16148
Host: localhost:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: APIKey=cf8a839c-0801-4d59-8bb8-8265614055b6;
cart=%5B%7B%22VID%22%3A%22001%22%2C%22PID%22%3A101%2C%22Size%22%3A%22S%22%2C%22color%2
2%3A%22white%22%2C%22quantity%22%3A1%7D%2C%7B%22VID%22%3A%22001%22%2C%22PID%22%3A102%2
C%22Size%22%3A%22L%22%2C%22color%22%3A%22green%22%2C%22quantity%22%3A1%7D%2C%7B%22VID%
22%3A%22001%22%2C%22PID%22%3A105%2C%22Size%22%3A%22S%22%2C%22color%22%3A%22black%22%2C
%22quantity%22%3A1%7D%5D
Content-Length: 0
```

Response:

```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
Set-Cookie: cart=; Path=/; Expires=Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 286
ETag: W/"11e-BVzQXXgzxP9YklnlOT+Be/kVh1o"
Date: Thu, 08 Dec 2022 04:51:03 GMT
```

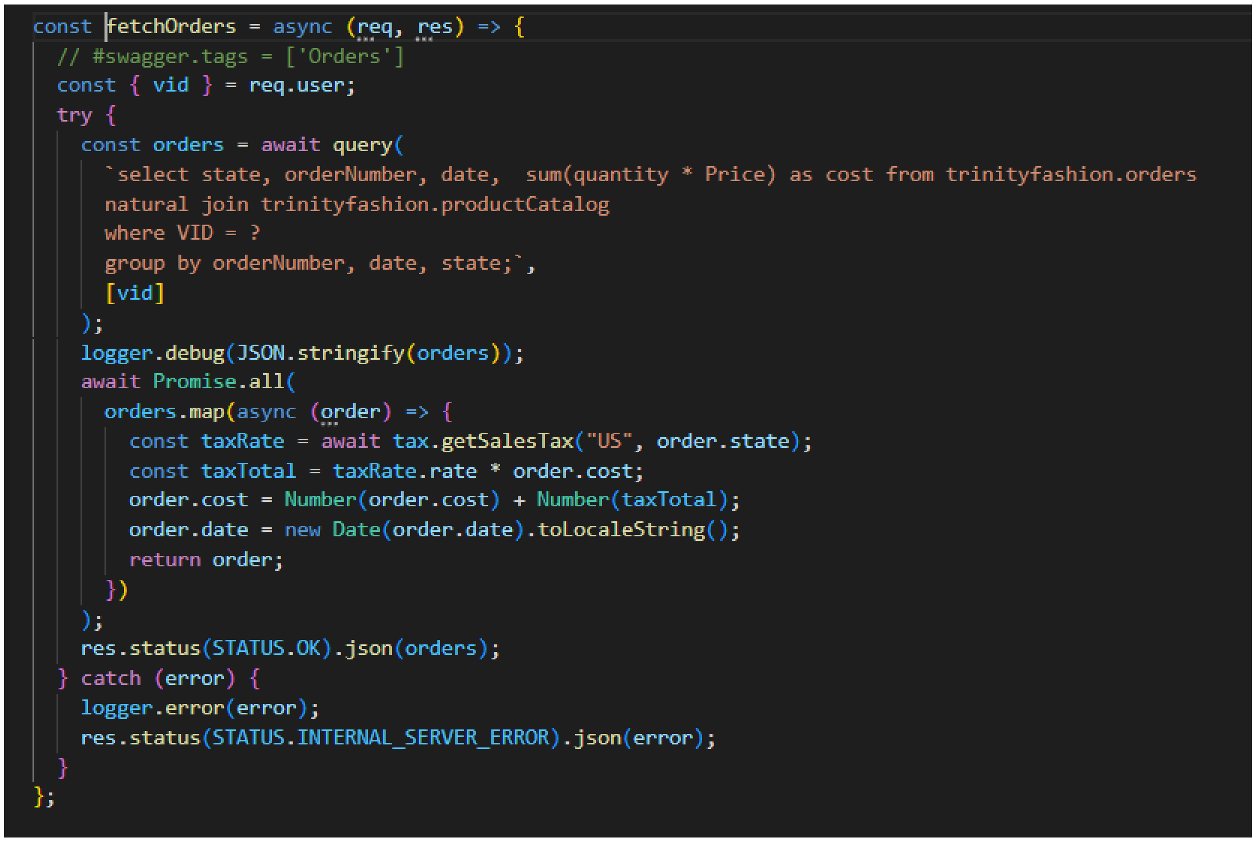
```
Connection: keep-alive
Keep-Alive: timeout=5

{"orderNum":"fee536e1-ab19-4761-a058-75eb58470d1a","date":"2022-12-08T04:51:03.856Z","
cart":[{"VID":"001","PID":101,"Size":"S","color":"white","quantity":1},{"VID":"001","P
ID":102,"Size":"L","color":"green","quantity":1},{"VID":"001","PID":105,"Size":"S","co
lor":"black","quantity":1}]}
```

Fetch Orders (Add to the content of a collection and linking collections)

Server Endpoint:

GET http://localhost:8000/orders

Function Definition:



```
const fetchOrders = async (req, res) => {
  // #swagger.tags = ['Orders']
  const { vid } = req.user;
  try {
    const orders = await query(
      `select state, orderNumber, date,  sum(quantity * Price) as cost from trinityfashion.orders
      natural join trinityfashion.productCatalog
      where VID = ?
      group by orderNumber, date, state;`,
      [vid]
    );
    logger.debug(JSON.stringify(orders));
    await Promise.all(
      orders.map(async (order) => {
        const taxRate = await tax.getSalesTax("US", order.state);
        const taxTotal = taxRate.rate * order.cost;
        order.cost = Number(order.cost) + Number(taxTotal);
        order.date = new Date(order.date).toLocaleString();
        return order;
      })
    );
    res.status(STATUS.OK).json(orders);
  } catch (error) {
    logger.error(error);
    res.status(STATUS.INTERNAL_SERVER_ERROR).json(error);
  }
};
```

Request:

```
GET /orders HTTP/1.1
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: e1fa6a9f-8d4c-4898-9240-e783d98d708a
Host: localhost:8000
```

```
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: APIKey=bf51a305-29ad-4880-9d1c-113431a9f6c4
```

Response:

```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: http://localhost:3000
Vary: Origin
Access-Control-Allow-Credentials: true
Content-Type: application/json; charset=utf-8
Content-Length: 117
ETag: W/"75-ai/1QvIUFaAq1iTC4m5ORHsaIAs"
Date: Sun, 18 Dec 2022 01:43:47 GMT
Connection: keep-alive
Keep-Alive: timeout=5
```

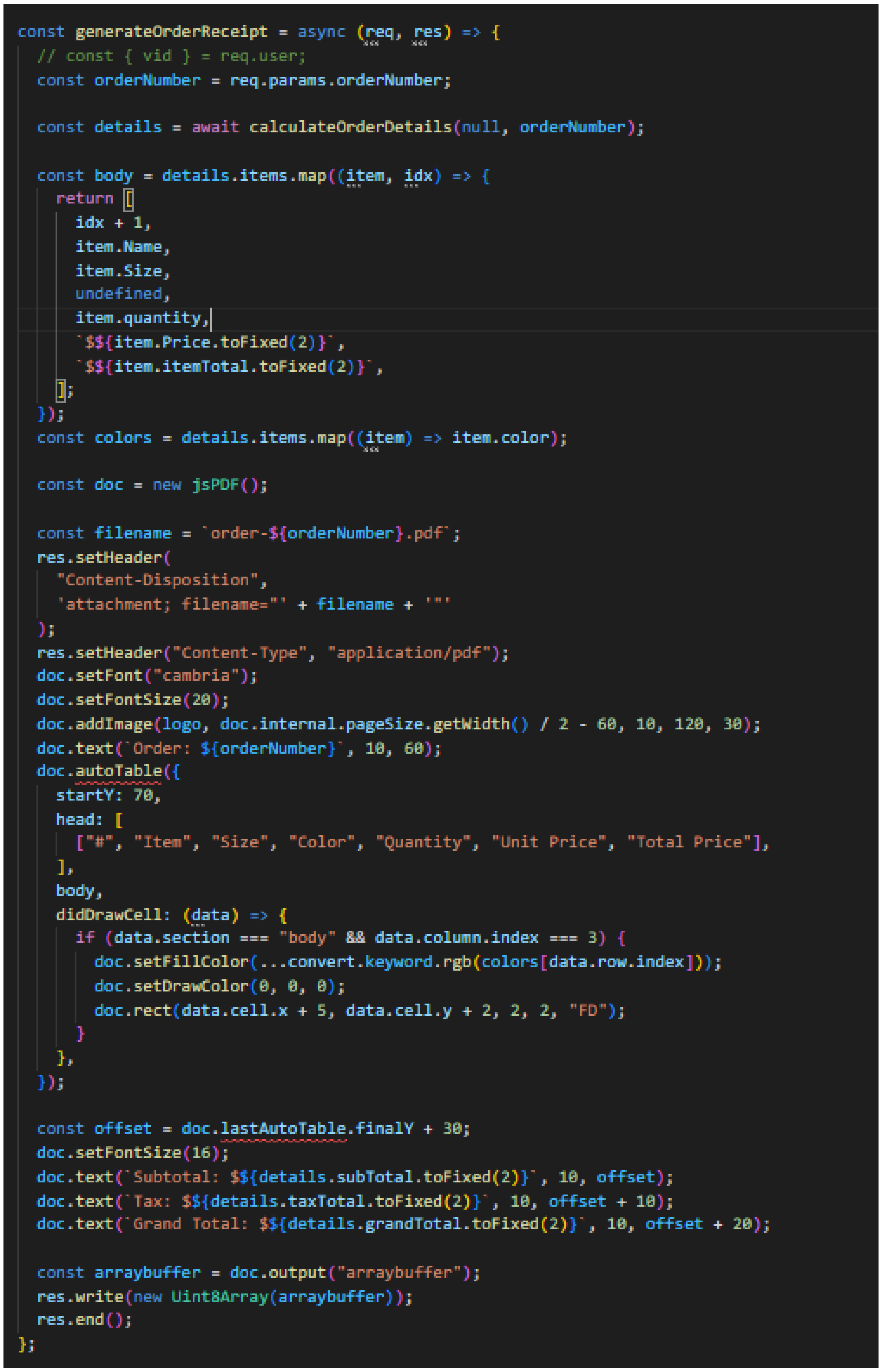
```
[{"state":"NJ","orderNumber":"fd746114-3f31-4c96-877d-53eda1afa8f3","date":"12/17/2022
, 8:43:42 PM","cost":173.2865}]
```

Create Receipt (Add to the content of a collection and linking collections)

Server Endpoint:

GET http://localhost:8000/orders/receipts/:orderNumber

Function Definition:



```
const generateOrderReceipt = async (req, res) => {
  // const { vid } = req.user;
  const orderNumber = req.params.orderNumber;

  const details = await calculateOrderDetails(null, orderNumber);

  const body = details.items.map((item, idx) => {
    return [
      idx + 1,
      item.Name,
      item.Size,
      undefined,
      item.quantity,
      `$${item.Price.toFixed(2)}`,
      `$${item.itemTotal.toFixed(2)}`,
    ];
  });
  const colors = details.items.map((item) => item.color);

  const doc = new jsPDF();

  const filename = `order-${orderNumber}.pdf`;
  res.setHeader(
    "Content-Disposition",
    'attachment; filename="' + filename + '"'
  );
  res.setHeader("Content-Type", "application/pdf");
  doc.setFont("cambria");
  doc.setFontSize(20);
  doc.addImage(logo, doc.internal.pageSize.getWidth() / 2 - 60, 10, 120, 30);
  doc.text(`Order: ${orderNumber}`, 10, 60);
  doc.autoTable({
    startY: 70,
    head: [
      ["#", "Item", "Size", "Color", "Quantity", "Unit Price", "Total Price"],
    ],
    body,
    didDrawCell: (data) => {
      if (data.section === "body" && data.column.index === 3) {
        doc.setFillColor(...convert.keyword.rgb(colors[data.row.index]));
        doc.setDrawColor(0, 0, 0);
        doc.rect(data.cell.x + 5, data.cell.y + 2, 2, 2, "FD");
      }
    },
  });

  const offset = doc.lastAutoTable.finalY + 30;
  doc.setFontSize(16);
  doc.text(`Subtotal: $${details.subTotal.toFixed(2)}`, 10, offset);
  doc.text(`Tax: $${details.taxTotal.toFixed(2)}`, 10, offset + 10);
  doc.text(`Grand Total: $${details.grandTotal.toFixed(2)}`, 10, offset + 20);

  const arraybuffer = doc.output("arraybuffer");
  res.write(new Uint8Array(arraybuffer));
  res.end();
};
```

Request:

```
GET /orders/receipts/fd746114-3f31-4c96-877d-53eda1afa8f3 HTTP/1.1
User-Agent: PostmanRuntime/7.29.2
```

```
Accept: */*
Postman-Token: 043b3570-119f-4859-a582-6a63fa2cce5a
Host: localhost:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: APIKey=bf51a305-29ad-4880-9d1c-113431a9f6c4
```

Response:

```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: http://localhost:3000
Vary: Origin
Access-Control-Allow-Credentials: true
Content-Disposition: attachment;
filename="order-fd746114-3f31-4c96-877d-53eda1afa8f3.pdf"
Content-Type: application/pdf
Date: Sun, 18 Dec 2022 01:44:17 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Transfer-Encoding: chunked
The console only shows response bodies smaller than 10 KB inline.
```

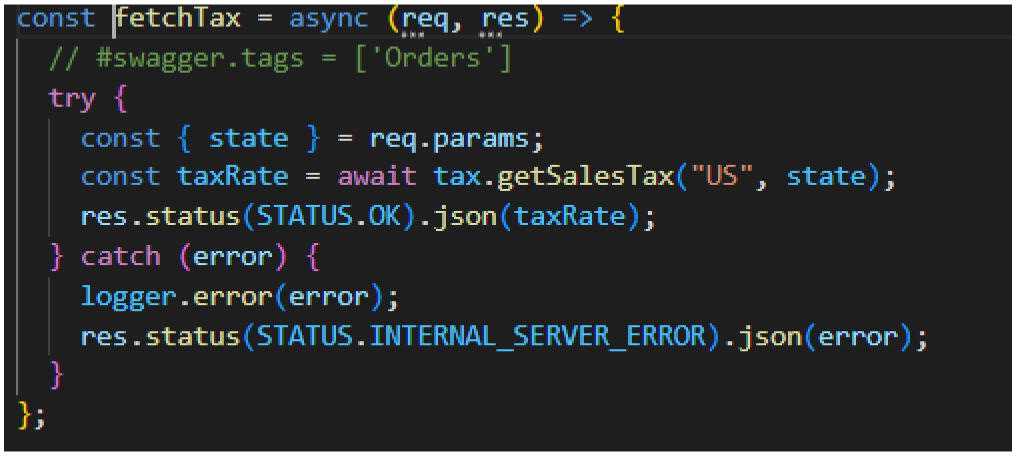
{PDF of receipt also returned, but too large to show in large log console of Postman}

Get Tax Rate

Server Endpoint:

GET http://localhost:8000/orders/tax/:state

Function Definition:



```
const fetchTax = async (req, res) => {
  // #swagger.tags = ['Orders']
  try {
    const { state } = req.params;
    const taxRate = await tax.getSalesTax("US", state);
    res.status(STATUS.OK).json(taxRate);
  } catch (error) {
    logger.error(error);
    res.status(STATUS.INTERNAL_SERVER_ERROR).json(error);
  }
};
```

Response:

```
GET /orders/tax/NJ HTTP/1.1
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Postman-Token: e5e45435-bca1-4443-b5a8-eeeec8282cc6
Host: localhost:8000
```

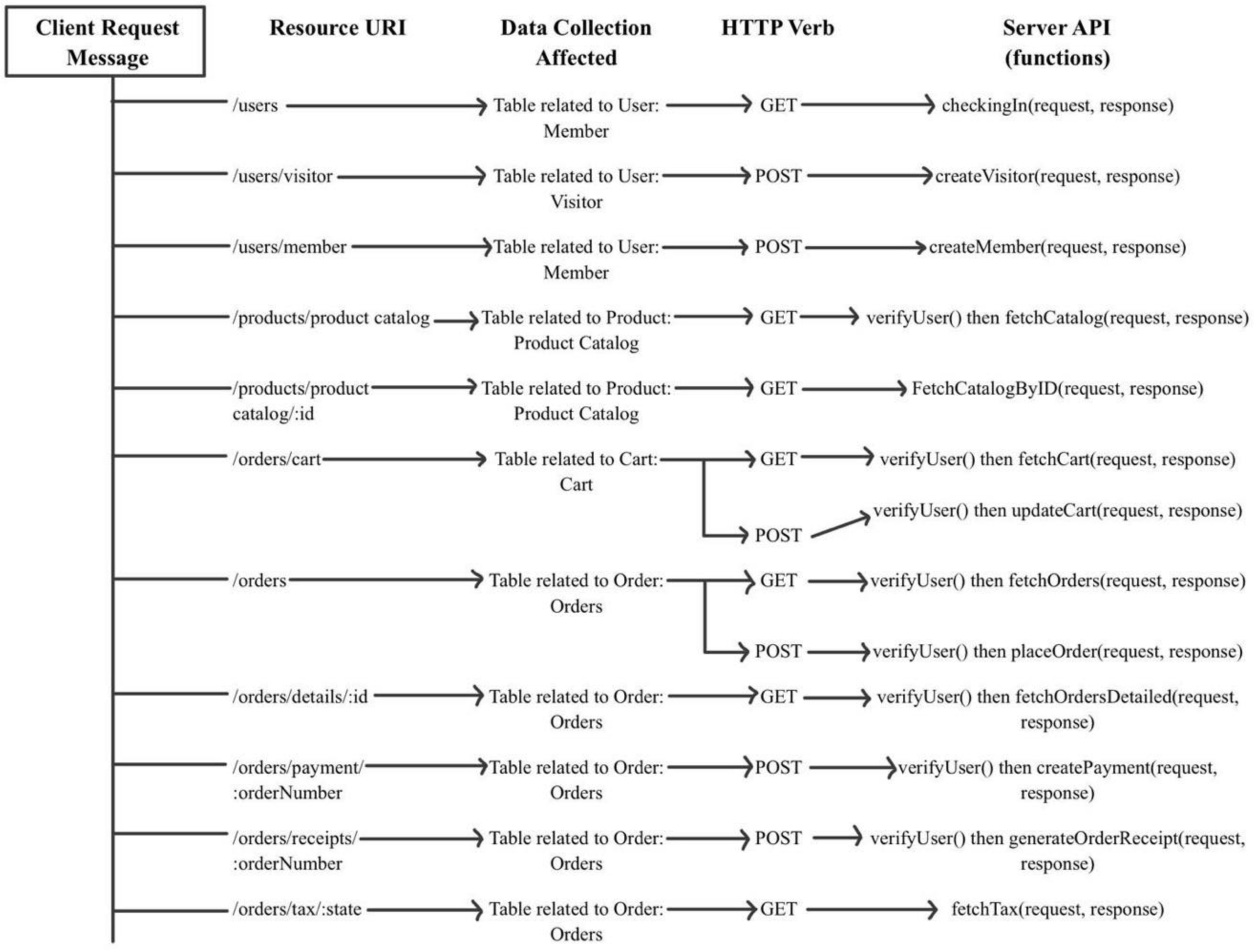
```
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

Request:

```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: http://localhost:3000
Vary: Origin
Access-Control-Allow-Credentials: true
Content-Type: application/json; charset=utf-8
Content-Length: 147
ETag: W/"93-1VS3K9g4ENbdeZI5t9mgUyFg1wQ"
Date: Sun, 18 Dec 2022 02:18:15 GMT
Connection: keep-alive
Keep-Alive: timeout=5
```

{"type":"vat","rate":0.07,"area":"worldwide","exchange":"consumer","charge":{"direct":
true,"reverse":false},"details":[{"type":"vat","rate":0.07}]}

Routing of HTTP Requests To API


Client Request Message
Resource URI
Data Collection Affected
HTTP Verb
Server API (functions)
/users
Table related to User: Member
GET
checkingIn(request, response)
/users/visitor
Table related to User: Visitor
POST
createVisitor(request, response)
/users/member
Table related to User: Member
POST
createMember(request, response)
/products/product catalog
Table related to Product: Product Catalog
GET
verifyUser() then fetchCatalog(request, response)
/products/product catalog/:id
Table related to Product: Product Catalog
GET
FetchCatalogByID(request, response)
/orders/cart
Table related to Cart: Cart
GET
verifyUser() then fetchCart(request, response)
POST
verifyUser() then updateCart(request, response)
/orders
Table related to Order: Orders
GET
verifyUser() then fetchOrders(request, response)
POST
verifyUser() then placeOrder(request, response)
/orders/details/:id
Table related to Order: Orders
GET
verifyUser() then fetchOrdersDetailed(request, response)
/orders/payment/:orderNumber
Table related to Order: Orders
POST
verifyUser() then createPayment(request, response)
/orders/receipts/:orderNumber
Table related to Order: Orders
POST
verifyUser() then generateOrderReceipt(request, response)
/orders/tax/:state
Table related to Order: Orders
GET
fetchTax(request, response)