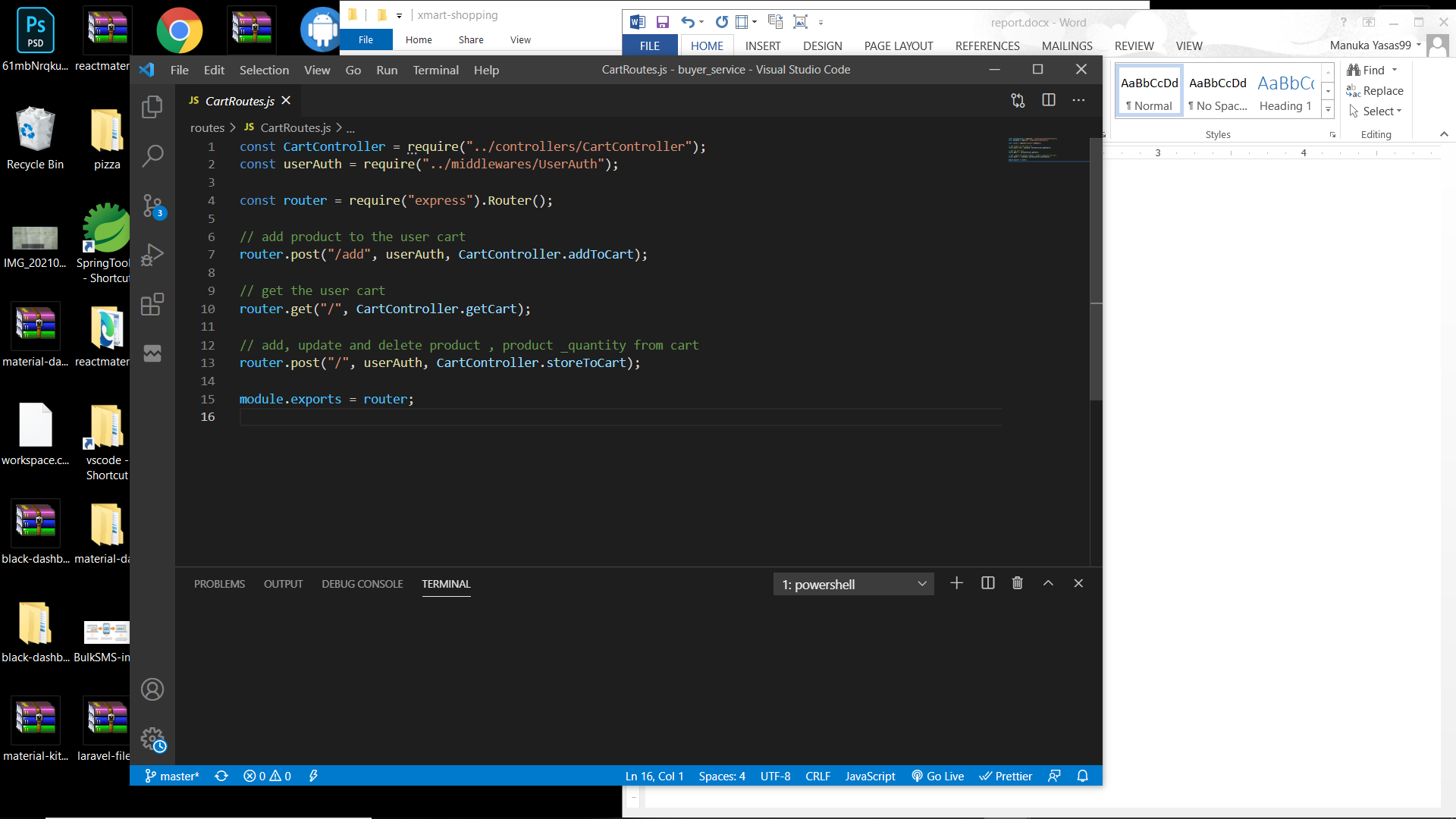
1. Process of login
2. Process of registering
3. Process of adding a product
4. Process of updating a product
5. Process of deleting a product
6. Process of adding product to cart
7. Process of updating products in cart
8. Process of retrieving user cart
9. Process of creating a new order
10. Process of updating / saving order
11. Process of retrieving order details
12. Process of placing cash on delivery order.
13. Process of notifying the payment
14. Process of payment through credit card
15. Process of payment through mobile service provider
16. Process of requesting mobile pin
17. Process of saving delivery details.

**Shopping cart management**

Service functions / interfaces to manage the shopping cart.



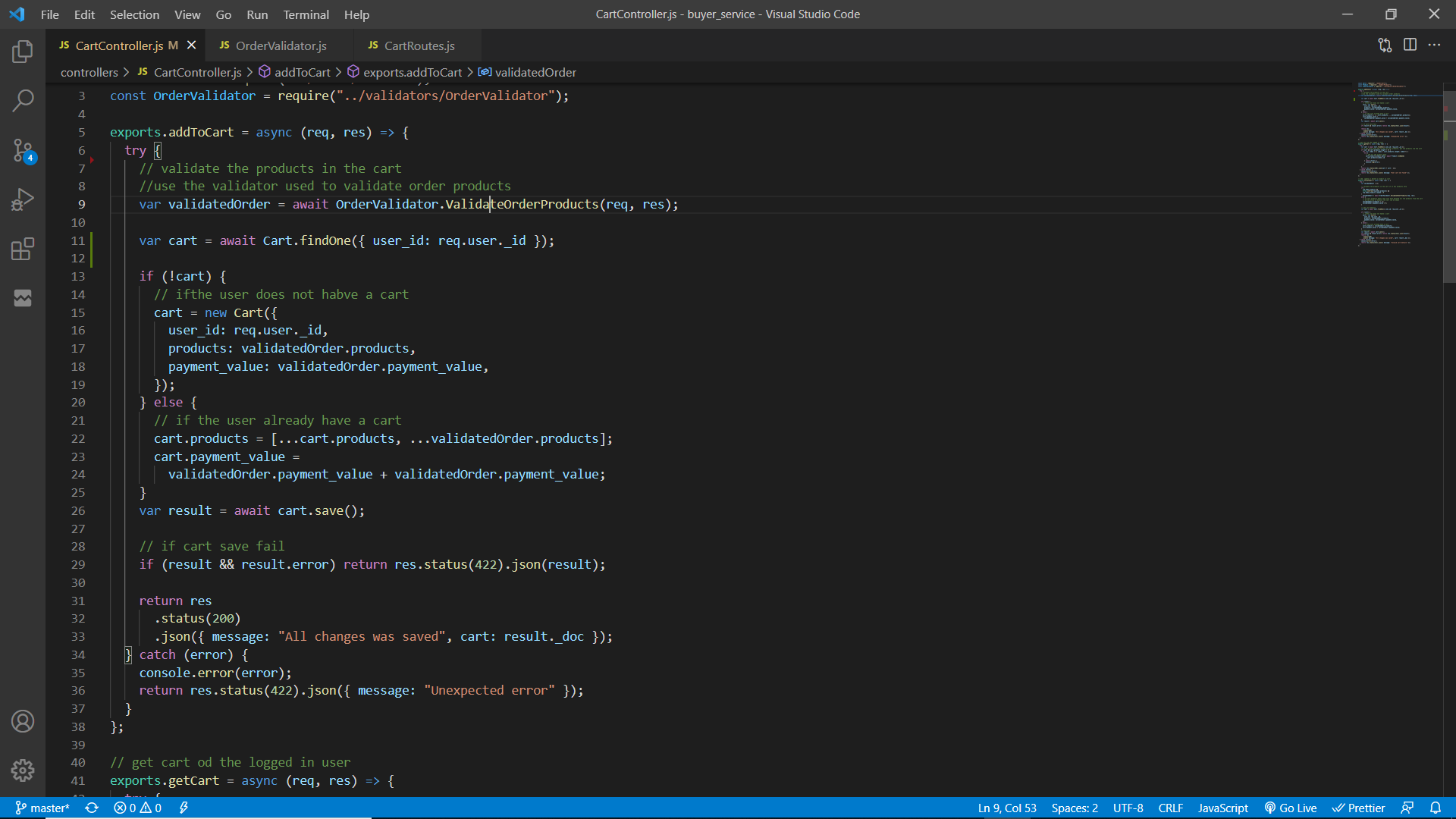
There are three routes in cart management.

1. A POST request with product and its quantity to add a product to shopping cart.
2. A GET request to retrieve user’s cart details with products.
3. A POST request to save / update product quantities in the shopping cart.

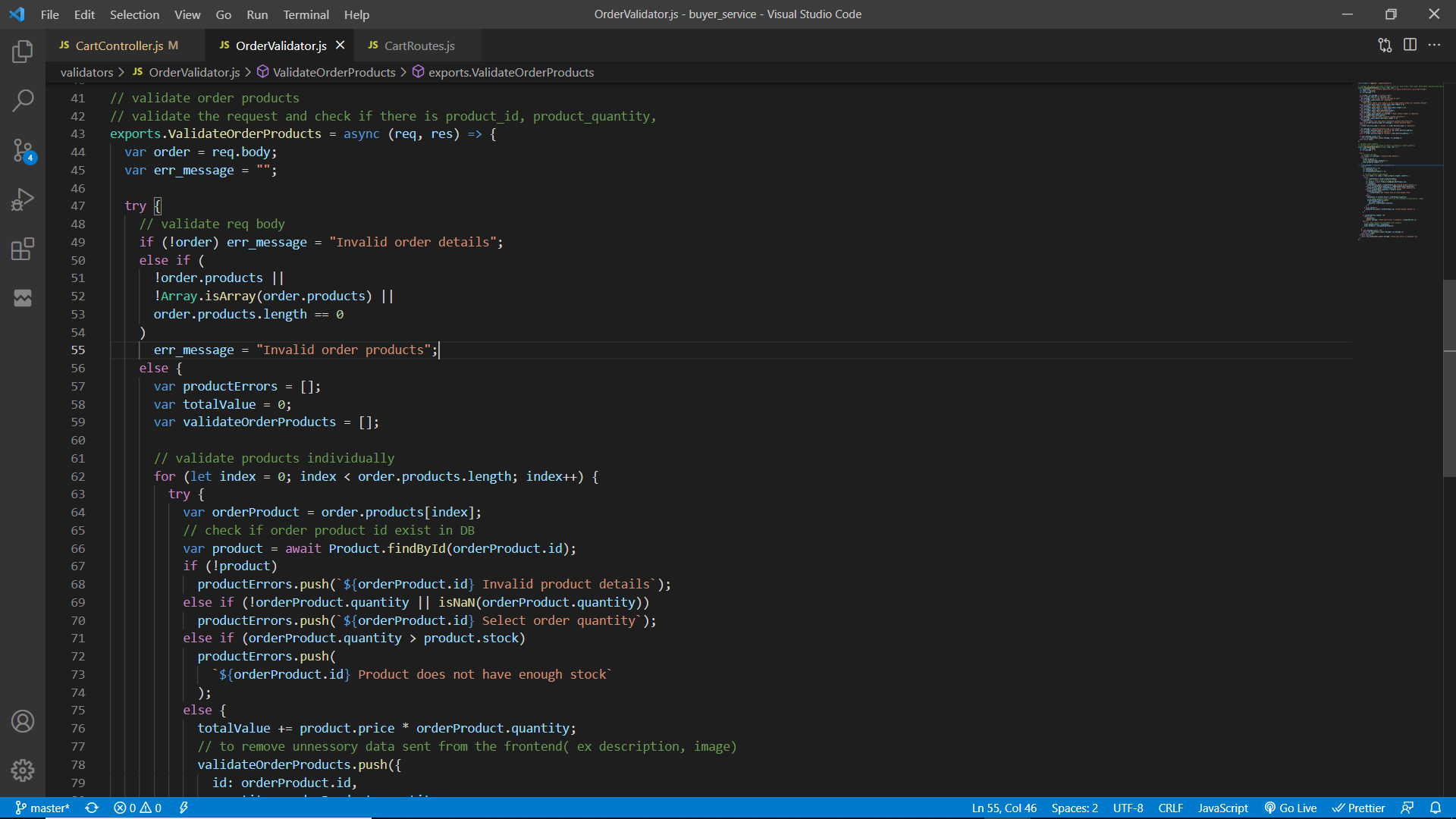
All these routes are mapped using the WSO2 Enterprise Integrator.

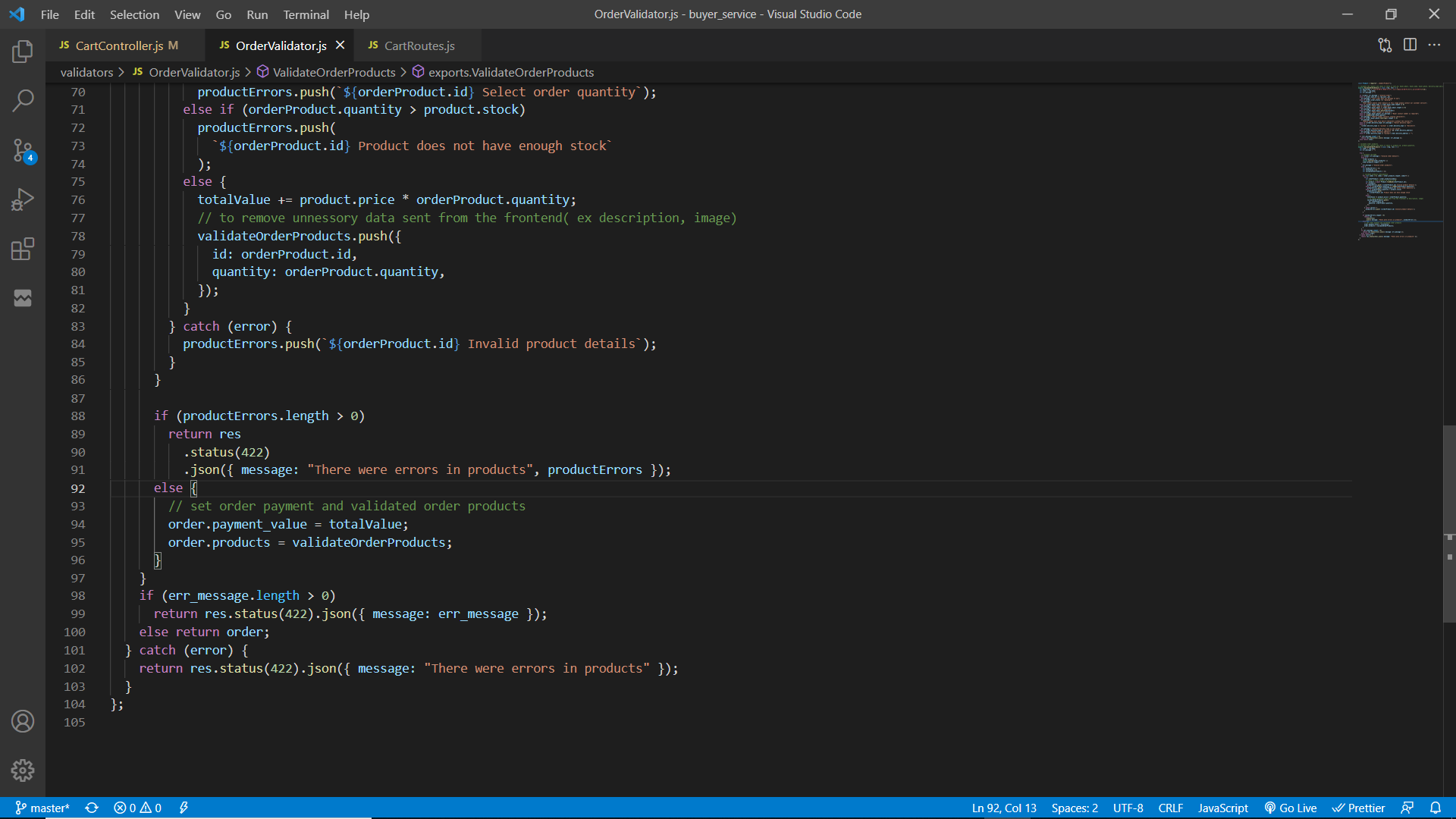
**Add a product to shopping cart**

Service function to add a product to cart



Validator function to validate request body such as valid products, products have stock (validateOrderProducts)





The initial POST request will be sent to the main application server through ESB. (WSO2 EI).

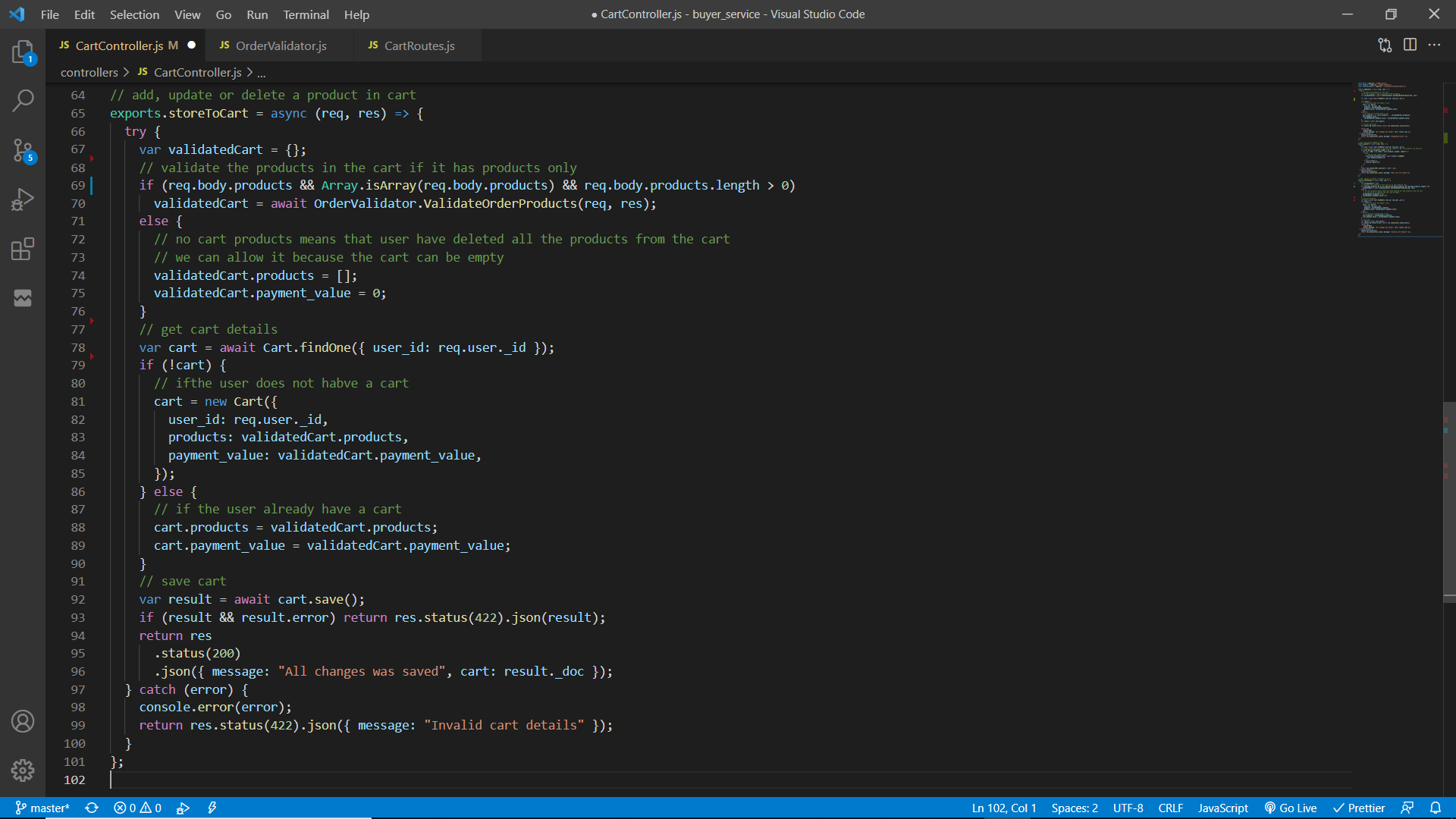
To add a product to cart, user must be authenticated for this “userAuth” middleware is used to filter the request and allow only authenticated users. Request body must have product id and its quantity. Then the request is validated to check if it contains product data that exist in database also the product quantity requested is less than or equal with its stock else relevant error messages are sent back to the client through ESB. If the request body is validated, it will return the total cart value(value of products \* quantity) with the products.

Then user cart will be retrieved from the database using the user’s id. If the cart does not exist, create a new cart by providing user’s ID, products and the total cart value. Else, update the existing cart by adding the new product to cart products and the total cart value.

The cart will be saved/updated and if it fails then an error message will be sent else, the updated user cart object will be sent to the client through ESB.

**Update products quantity in shopping cart**

Service function



The initial POST request will be sent to the main application server through ESB. (WSO2 EI).

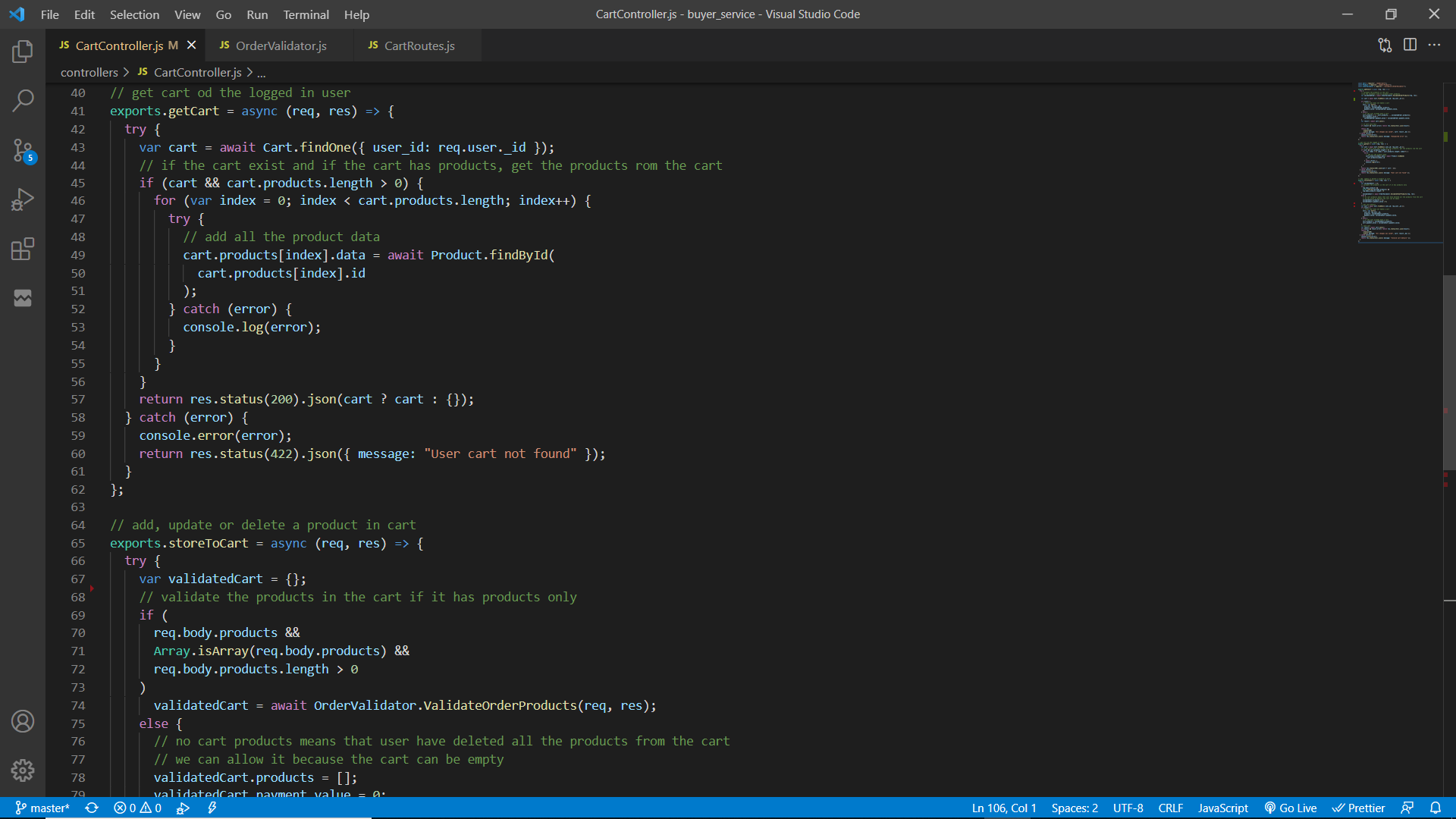
To update product quantity user must be authenticated. Therefore the “userAuth” middleware is used to filter the authenticated users/ requests . Then the request body is validated to check if it contains products. If no products were found that means user has deleted all products from cart, but if products are found then these product data must be validated for this the previous function “validateOrderProducts” will be used.

If all the data are validated then retrieve the user cart using the user’s ID and then if there is a cart, then replacing the previous products with new products along with their quantity and update the cart payment value. Else, create a new cart by providing the user’s id, products data and the total payment/cart value.

The cart will be saved/updated and if it fails then an error message will be sent else, the updated user cart object will be sent to the client through ESB.

**Retrieve user’s shopping cart**

Service function



The initial GET request will be sent to the main application server through ESB. (WSO2 EI).

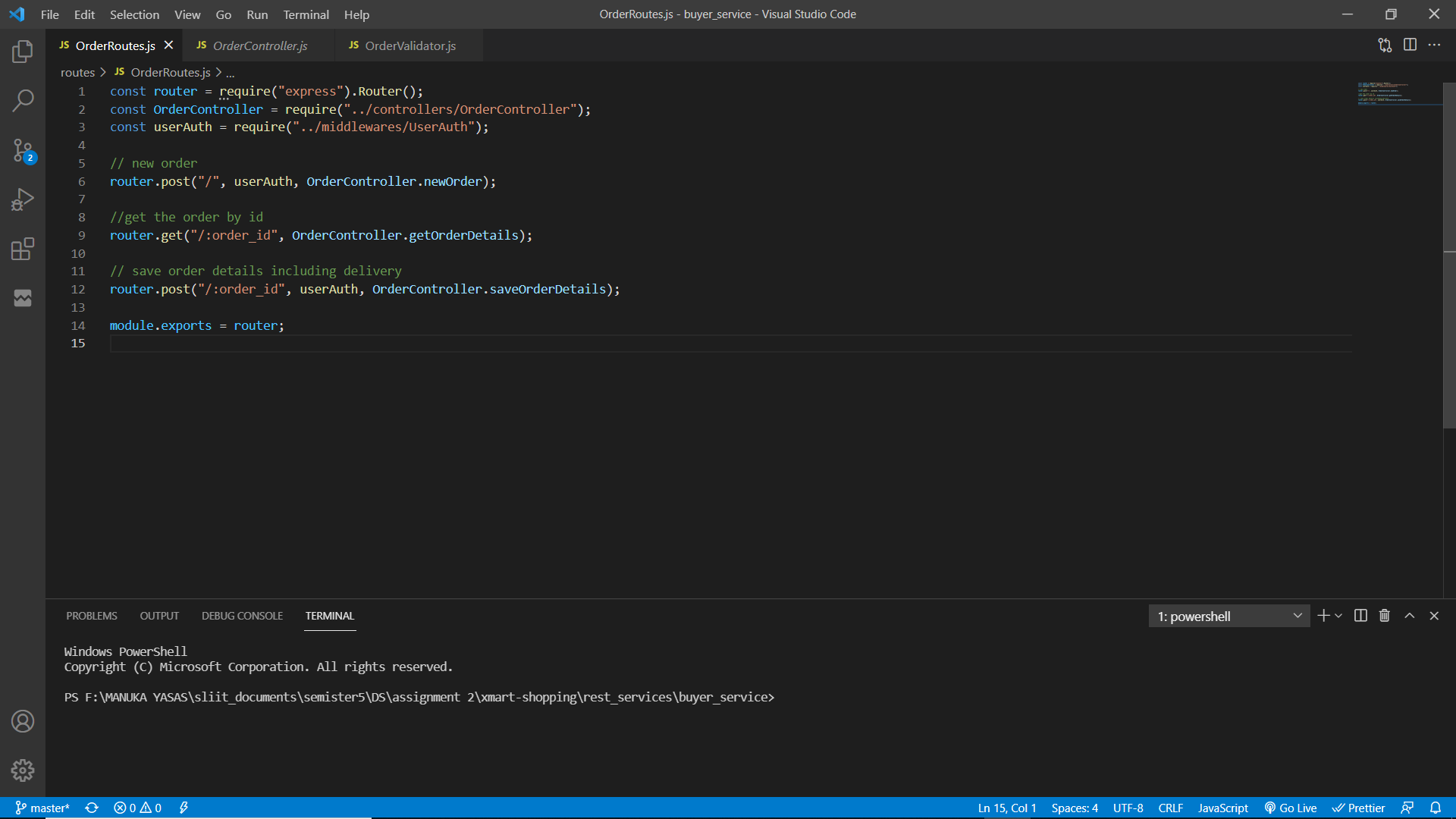
To retrieve user’s shopping cart, user must be authenticated. Therefore “userAuth” middleware is used to filter the authenticated users/ requests .

User’s cart will be retrieved from the database by using user id. If the cart contains products, loop through each product and retrieve it from the database using product id and append the product data to the cart product with a new property “data”. This is required to display product image, category, price and other data in the frontend other than only displaying the product ID.

Return the new cart with all product data and if there is no cart found then return an empty object via ESB to the client.

**Order management**

Service functions / interfaces to manage orders.



There are three routes to manage orders.

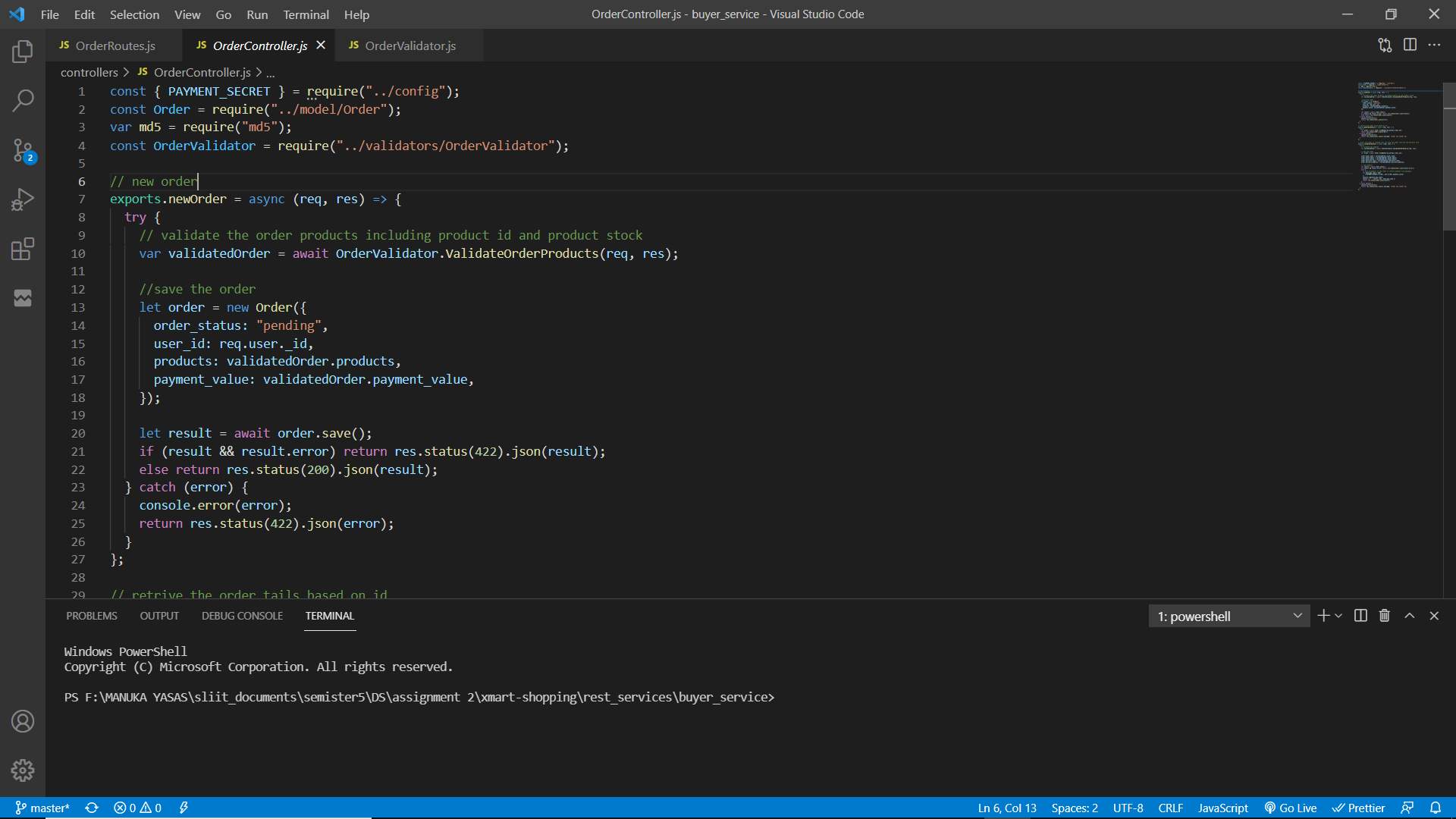
1. A POST request to create a new order with products and its quantities.
2. A GET request to retrieve an order by the order ID sent as a parameter in the request.
3. A POST request to save / update the order buyer details and delivery information.

All these routes are mapped using the WSO2 Enterprise Integrator.

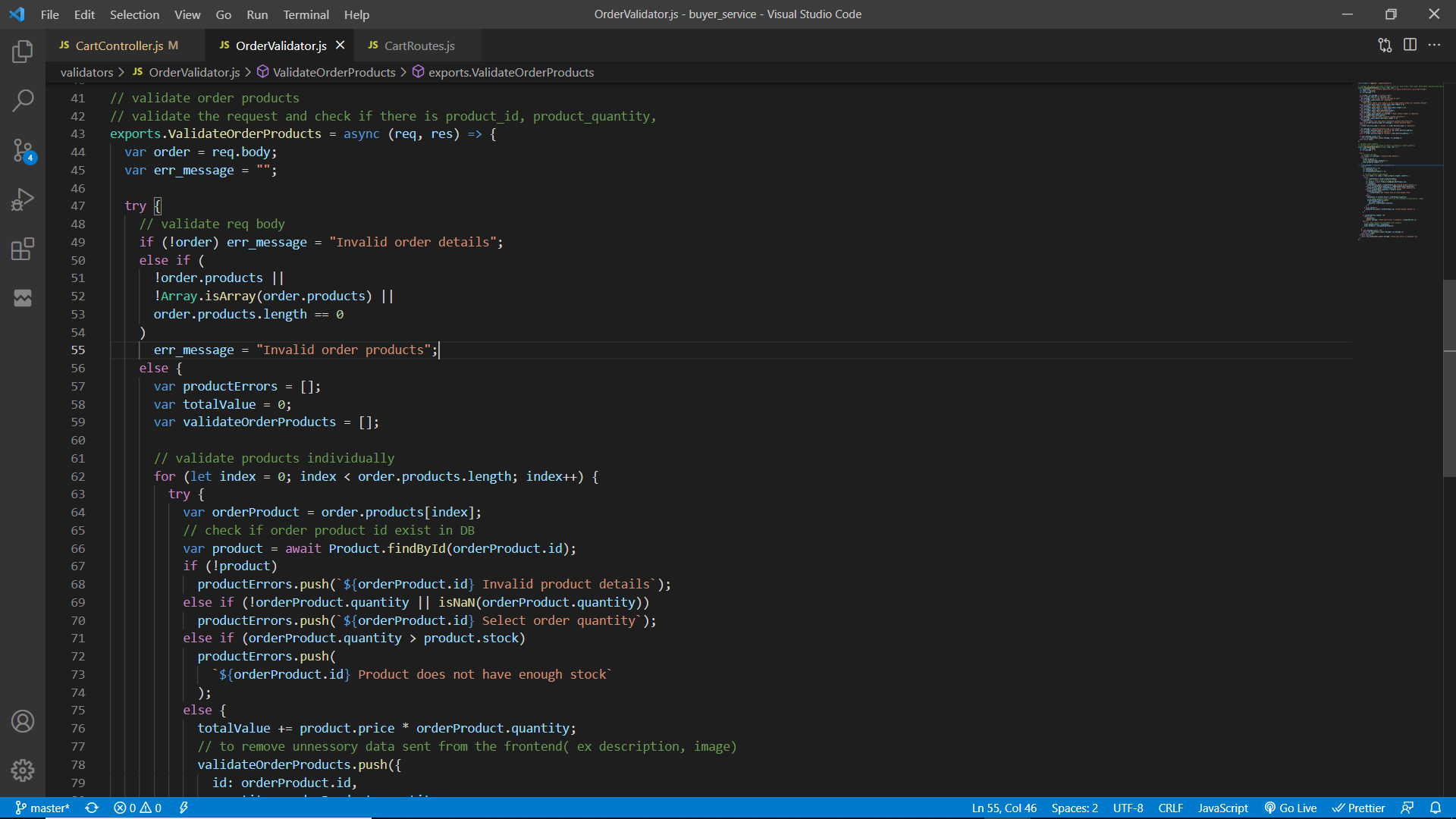
To place an order there are few steps. Initially products with their quantity will be sent to the server and then it will save it and return an object with the payment value and a unique order id. User will navigate to save buyer information and delivery information through the order id. After saving the above details user will navigate for the payments. Payments can be made through cash on delivery, credit / debit card (dummy service) or though mobile service provider (dummy service). After a successful payment user will get and email (through Gmail) and a SMS (though Twillio).

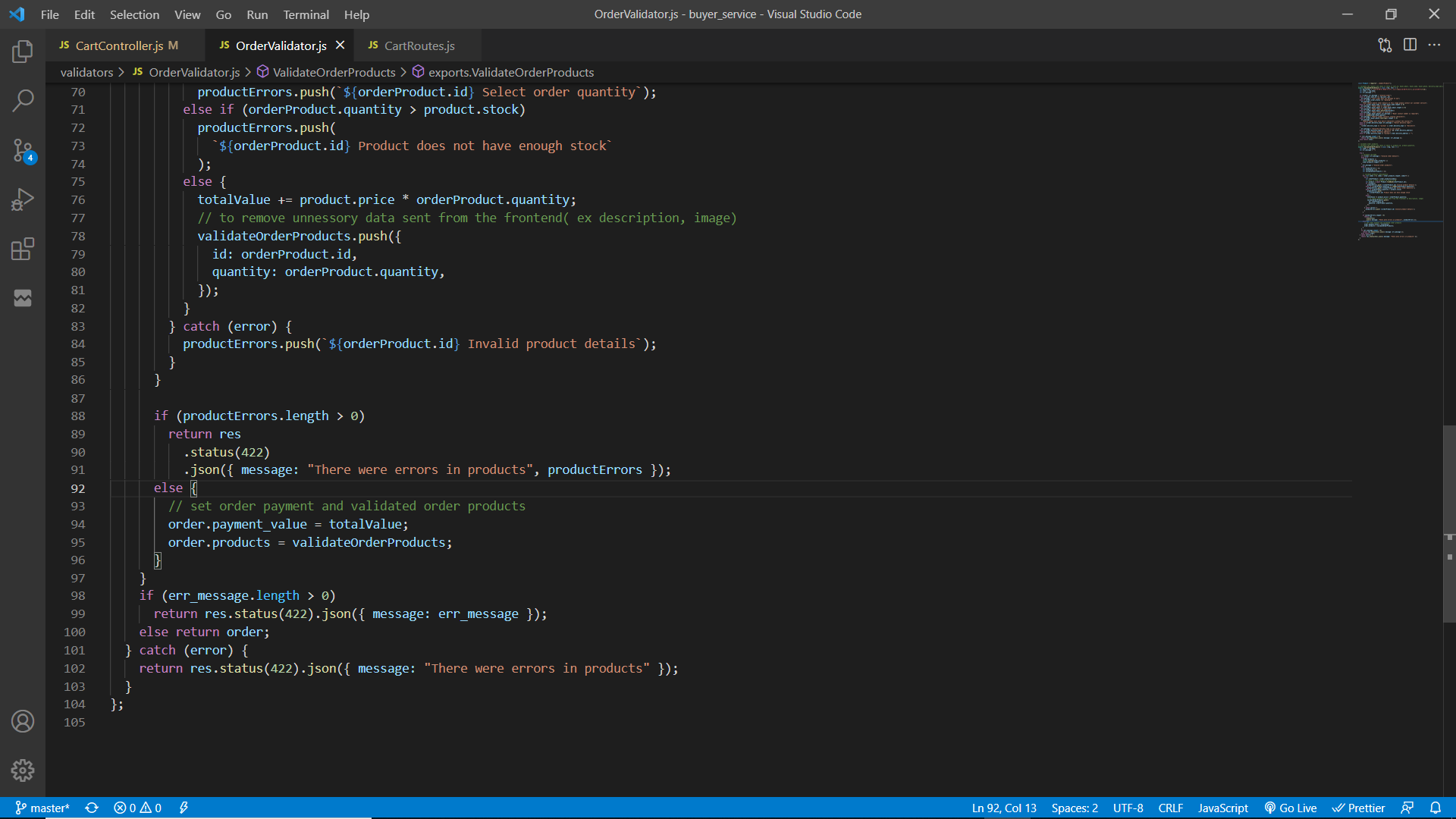
**Process of creating a new order**

Service function



Validator function to validate request body such as valid products, products have stock (validateOrderProducts)



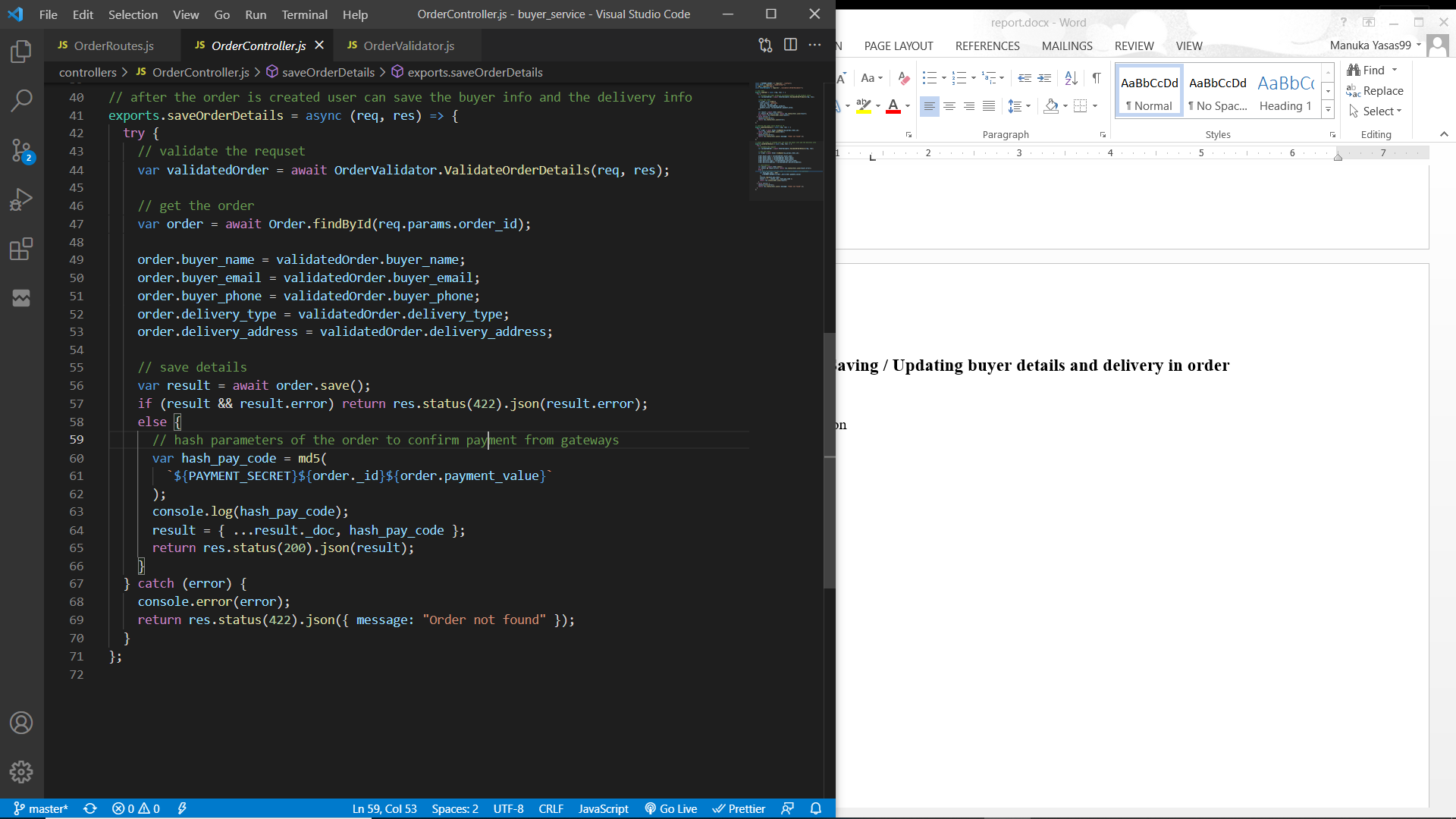


The initial POST request will be sent to the main application server through ESB. (WSO2 EI). To create a new order, user must be authenticated. Therefore the “userAuth” middleware is used to filter the authenticated users/ requests. Request body must have an array of objects where each object must have the product id and its quantity. This request is validated using the above “validateOrderProducts” function. This function will iterate the array of objects and check if each object has a product id and then it checks if there is a product in the database with the product id in the object additionally, it validates that the product has enough stock to supply. If any of the validations fail, respective error messages will be sent back to the client through ESB. Else, an object with the validated products array and total payment value (product price \* quantity) will be returned.

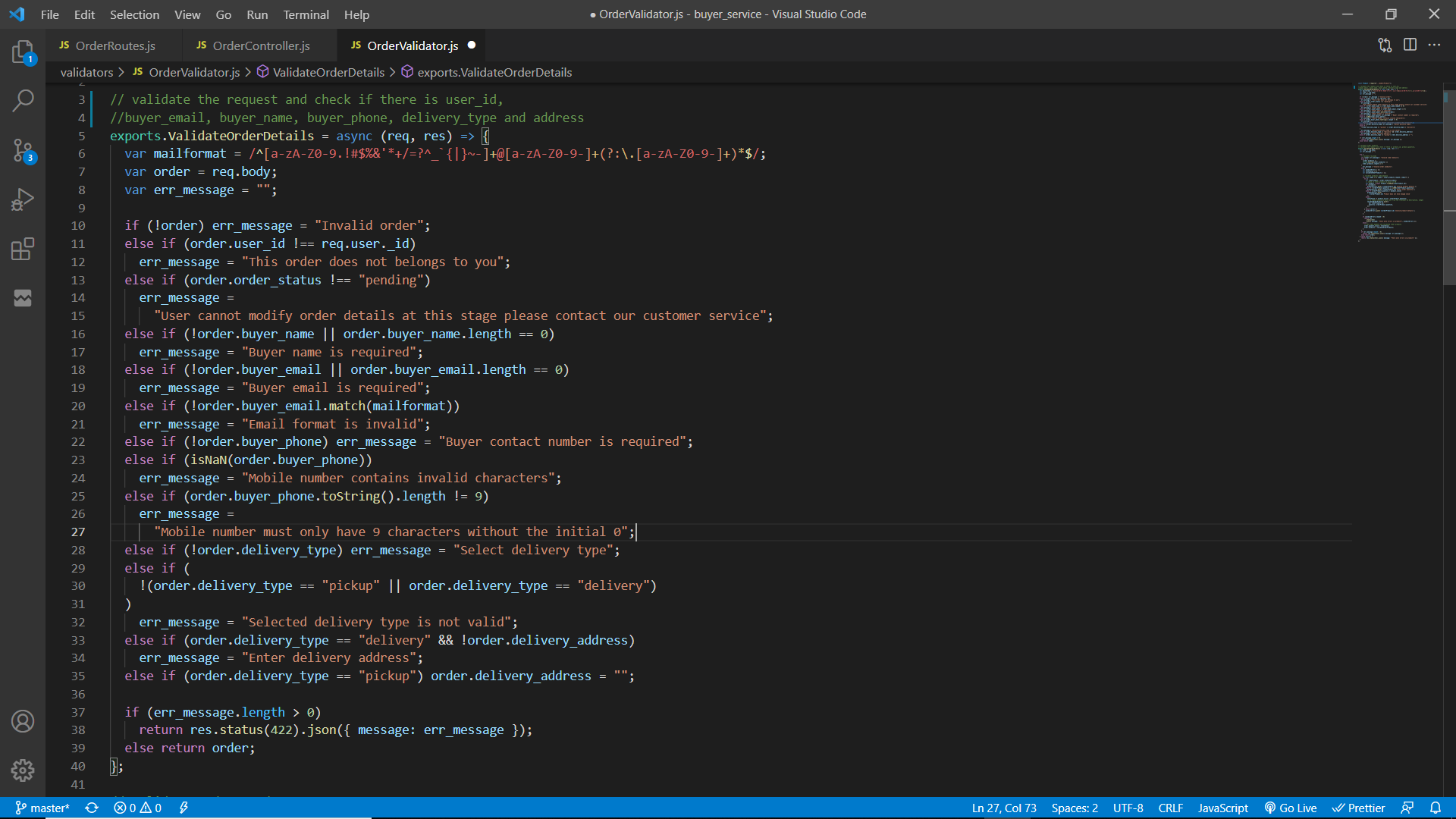
A new order will be created by setting the user’s id, products and payment value fields. The order status will be set to “pending”. This order object is saved and if it fails to save an error message is sent back to the client through ESB. Else, the saved object will be returned back to the client through ESB which will be required to save the delivery details and payment.

**Process of Saving / Updating buyer details and delivery in order**

Service function



Function “ValidateOrderDetails” will be used to validate buyer details and delivery information.



The initial POST request will be sent to the main application server through ESB. (WSO2 EI). To save / update the order, user must be authenticated. Therefore the “userAuth” middleware is used to filter the authenticated users/ requests. Request body must have the order id, buyer information such as buyer name, contact number and email; also, it must contain the delivery type mentioned. If the order needs to be delivered then delivery address field must not be empty. Delivery type can only be one of pick up or delivery. Mobile number must be a valid nine digit (Integer) number and the user email must be a valid email. All the above are validated using the function “ValidateOrderDetails” and if errors are found then respective error messages will be sent back to the client through ESB. Else, the validate request body will be returned.

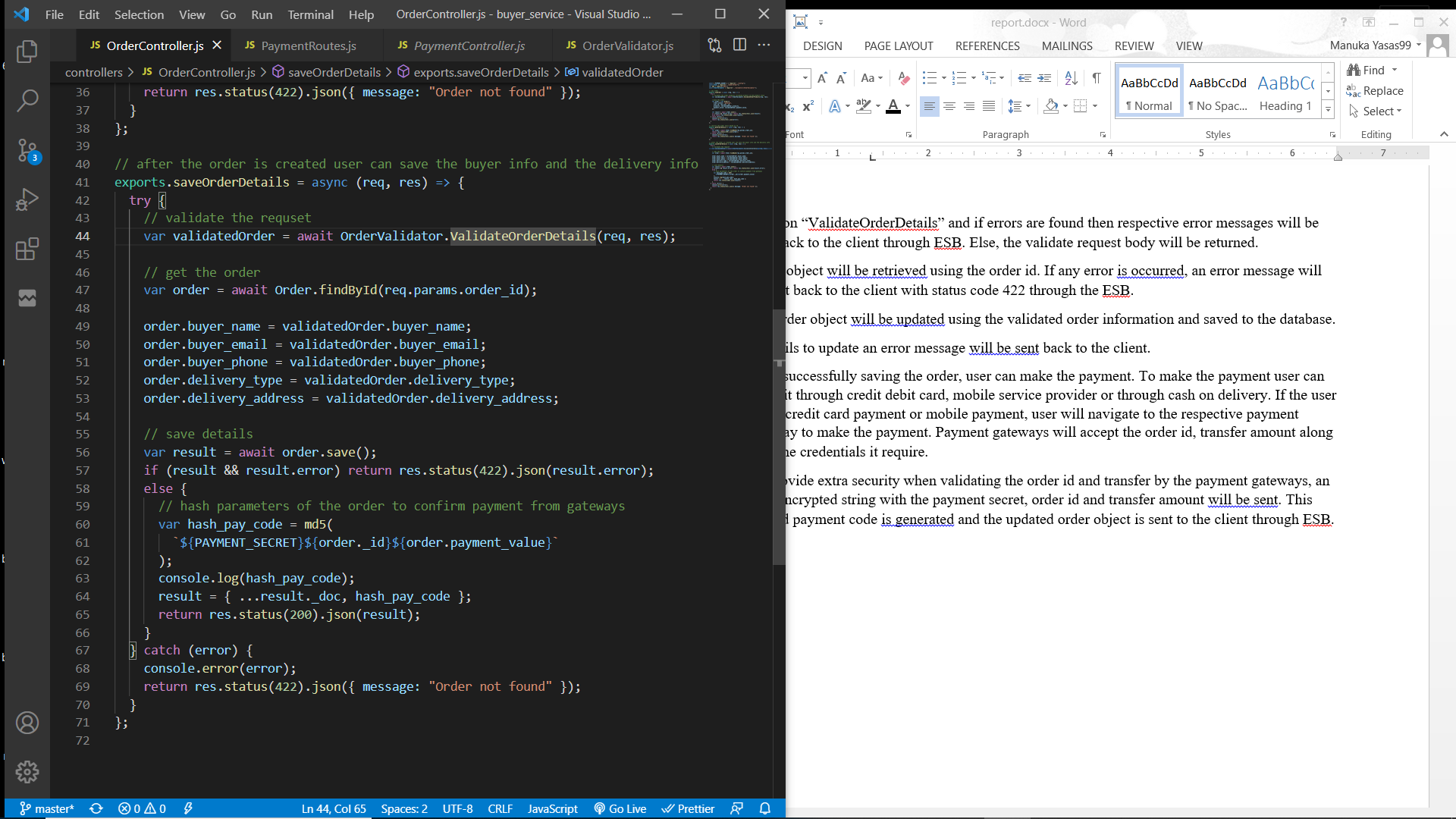
Order object will be retrieved using the order id. If any error is occurred, an error message will be sent back to the client with status code 422 through the ESB.

The order object will be updated using the validated order information and saved to the database.

If it fails to update an error message will be sent back to the client.

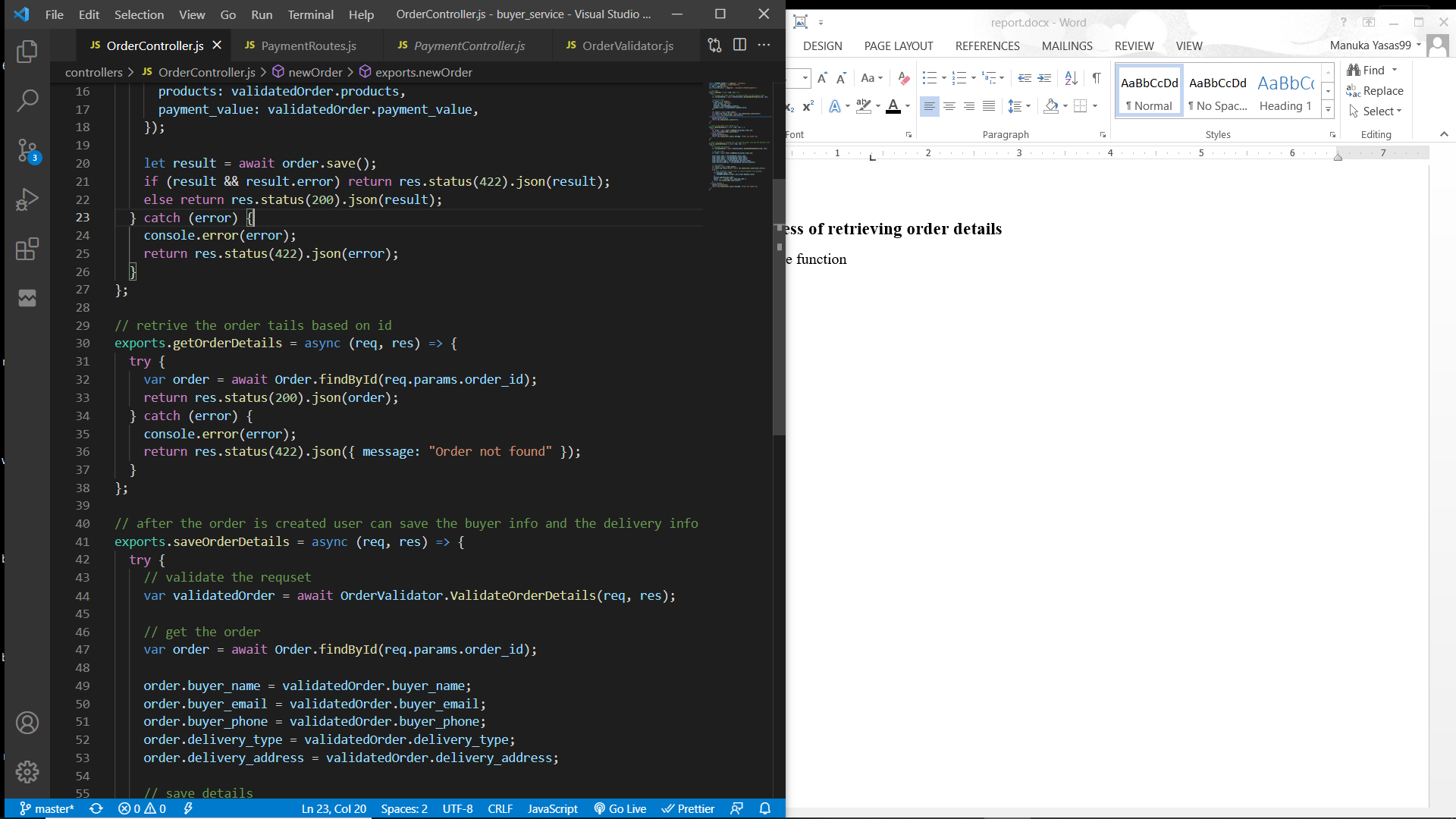
After successfully saving the order, user can make the payment. To make the payment user can make it through credit debit card, mobile service provider or through cash on delivery. If the user select credit card payment or mobile payment, user will navigate to the respective payment gateway to make the payment. Payment gateways will accept the order id, transfer amount along with the credentials it require.

To provide extra security when validating the order id and transfer by the payment gateways, an md5 encrypted string with the payment secret, order id and transfer amount will be sent. This hashed payment code is generated and the updated order object is sent to the client through ESB.



**Process of retrieving order details**

Service function



The initial GET request will be sent to the main application server through ESB. The order id is sent as a parameter and the order is retrieved using the order id. The retrieved order will be sent back to the client through the ESB. If an error is occurred then a response is sent back to the client with 422 status code.

**Process of placing cash on delivery order.**

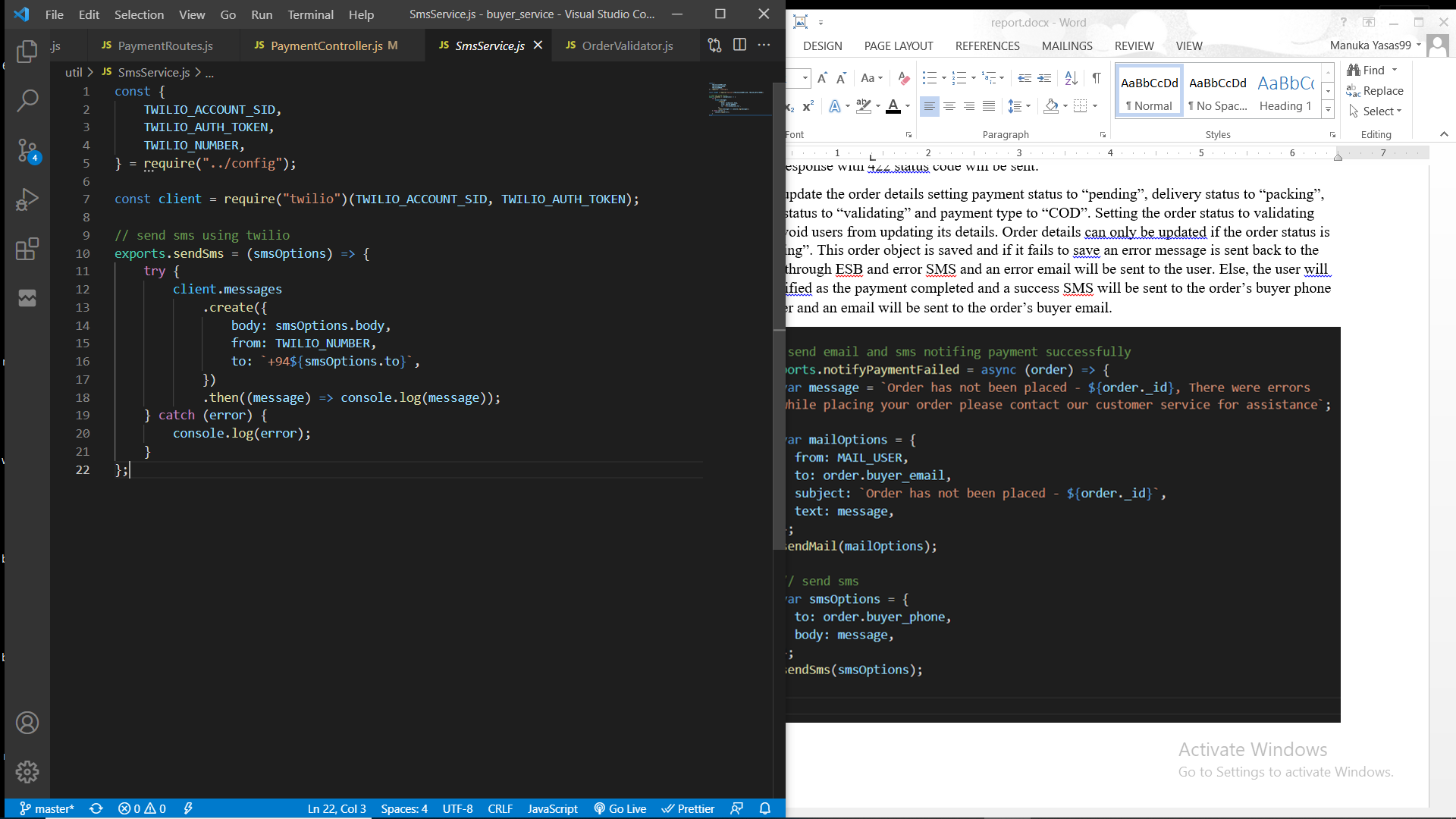
Service function



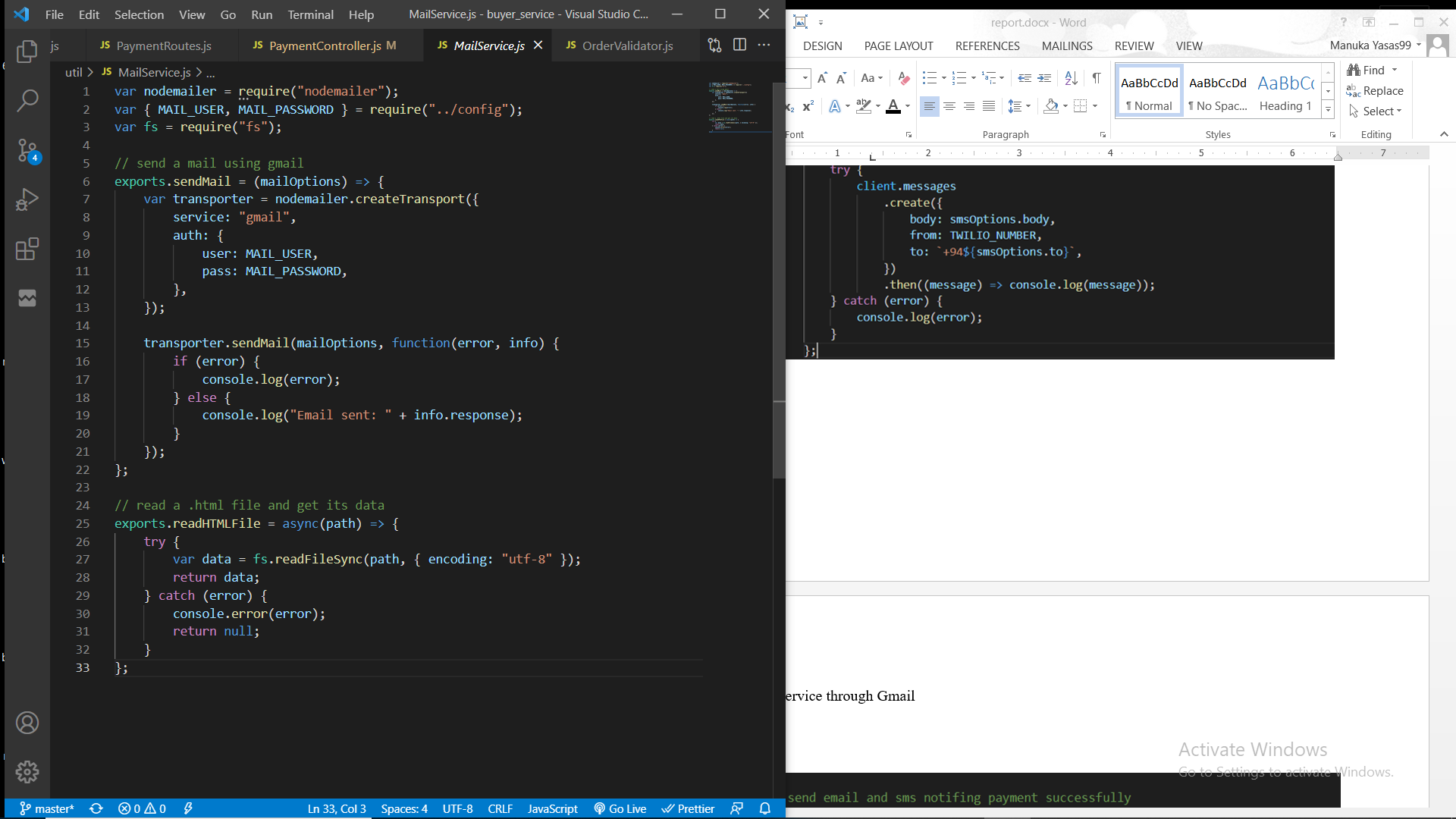
The initial POST request will be sent to the main application server through ESB. The request is required to send the order ID. Then the order object will be retrieved. Order details including the buyer and delivery information must be validated before placing the order. As in above functions “ValidateOrderDetails” function is used to validate the request. If any errors are found then an error response with 422 status code will be sent.

Then update the order details setting payment status to “pending”, delivery status to “packing”, order status to “validating” and payment type to “COD”. Setting the order status to validating will avoid users from updating its details. Order details can only be updated if the order status is “pending”. This order object is saved and if it fails to save an error message is sent back to the client through ESB and error SMS and an error email will be sent to the user. Else, the user will be notified as the payment completed and a success SMS will be sent to the order’s buyer phone number and an email will be sent to the order’s buyer email.

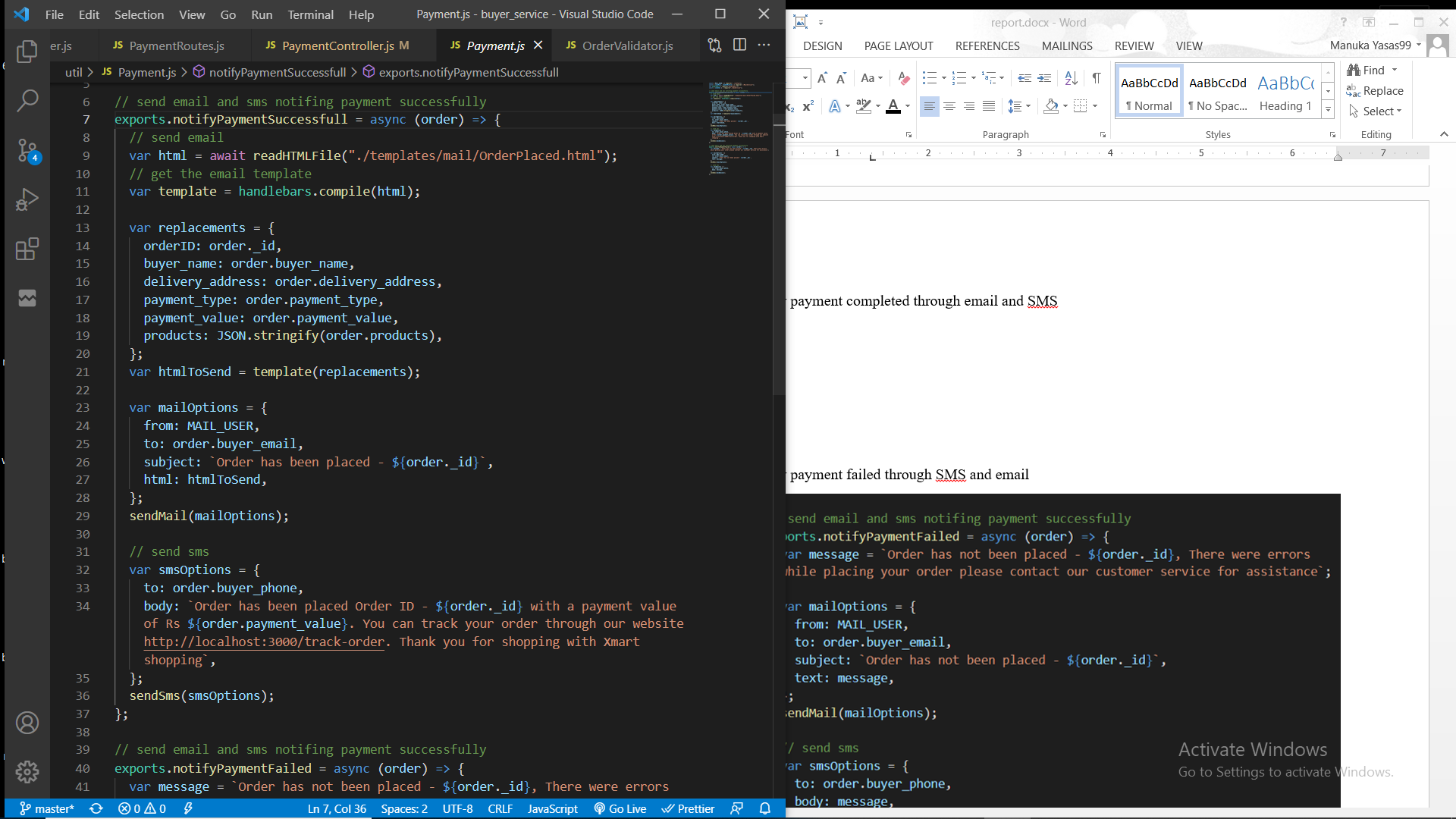
SMS service through TWILIO

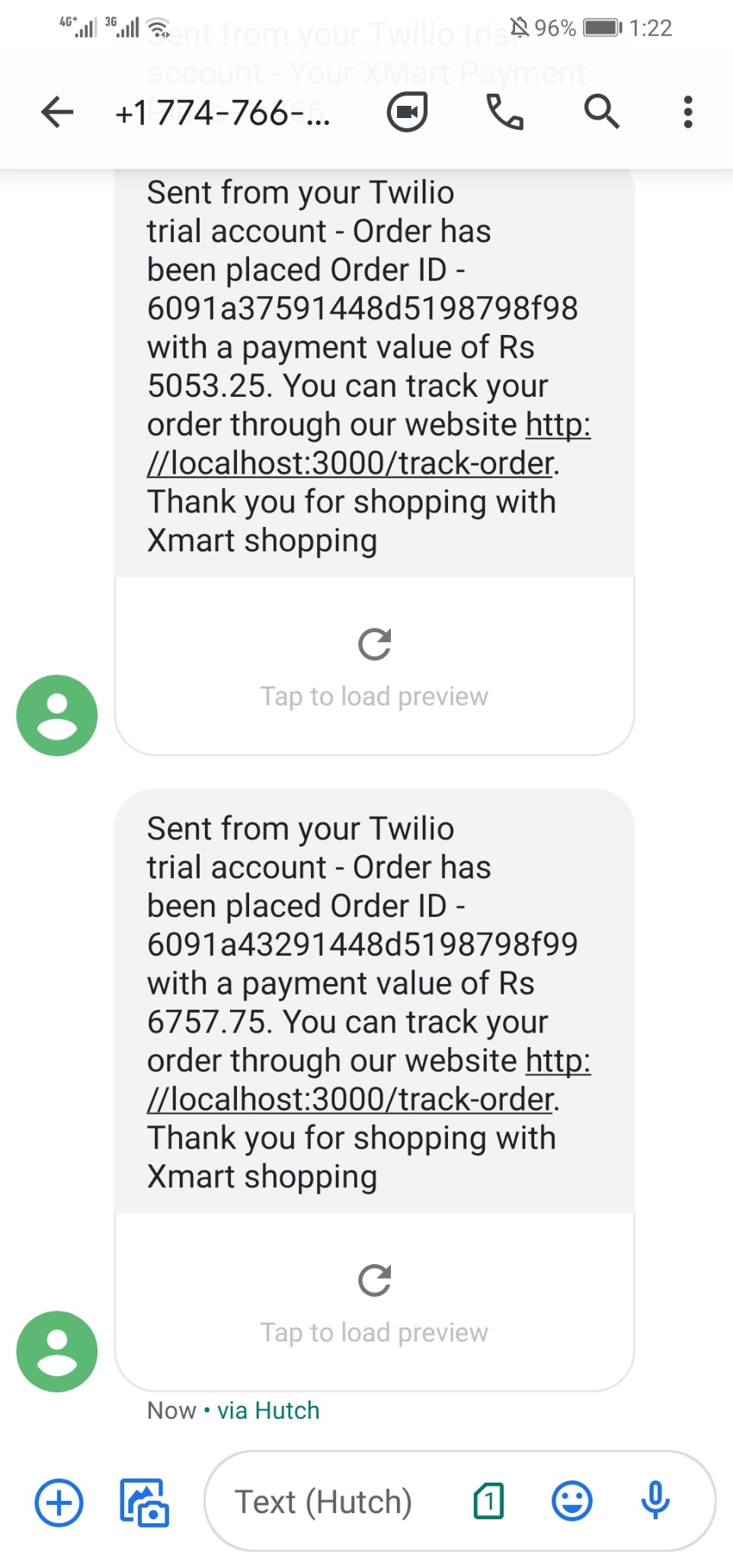


Mail service through Gmail



Notify payment completed through email and SMS





Notify payment failed through SMS and email

