**Full Stack Development with MERN**

**API Development and Integration Report**

|  |  |
| --- | --- |
| Date | 12 July 2024 |
| Team ID | SWTID1720171927 |
| Project Name | SBFoods- Food Ordering Website |
| Maximum Marks | 10 Marks |

**Project Title:** SBFoods- Food Ordering Website  
**Date:** 12 July 2024  
**Prepared by:** SWTID1720171927

**Objective**  
The objective of this report is to document the API development progress and key aspects of the backend services implementation for the SBFood project.

**Technologies Used**

* **Backend Framework:** Node.js with Express.js
* **Database:** MongoDB
* **Authentication:** OAuth

**Project Structure**

**Key Directories and Files**

1. **/controllers**

Functions used to handle requests and responses:

**-**addToCart: Add the dish selected to Cart

**-**removeFromCart: Remove a dish from the Cart

**-**getCart: Get the details of dishes added to the Cart

**-**addFood: Add a new dish along with it’s details to the menu

**-**listFood: Give the list of dishes in menu(database)

**-**removeFood: Remove a food item from the menu

- placeOrder: User placing order

-verifyOrder: Verifying the order placed

-userOrders: List of orders placed by a particular user

-listOrders: List of all orders placed on the platform

-updateStatus: Updating order status(Food Processing/Out for Delivery/Delivered)

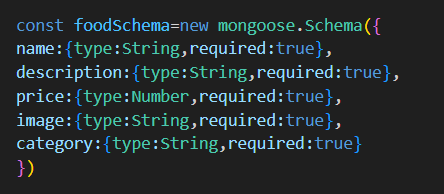
-loginUser: User logging in with email and password

-registerUser: Creating an account for new user

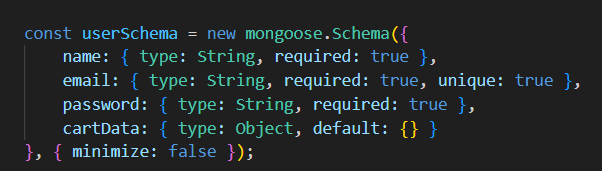
**2./models**

Mongoose schemas used for MongoDB Collections:

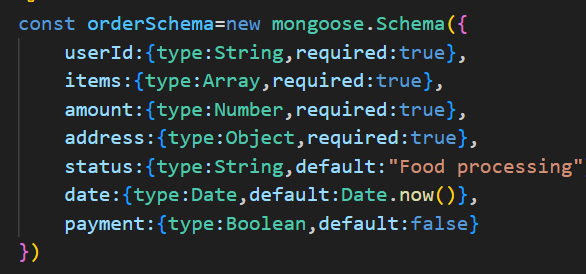
-foodSchema



-userSchema:



-orderSchema:



**3./routes**

API endpoints used for building the website:

Client-side:

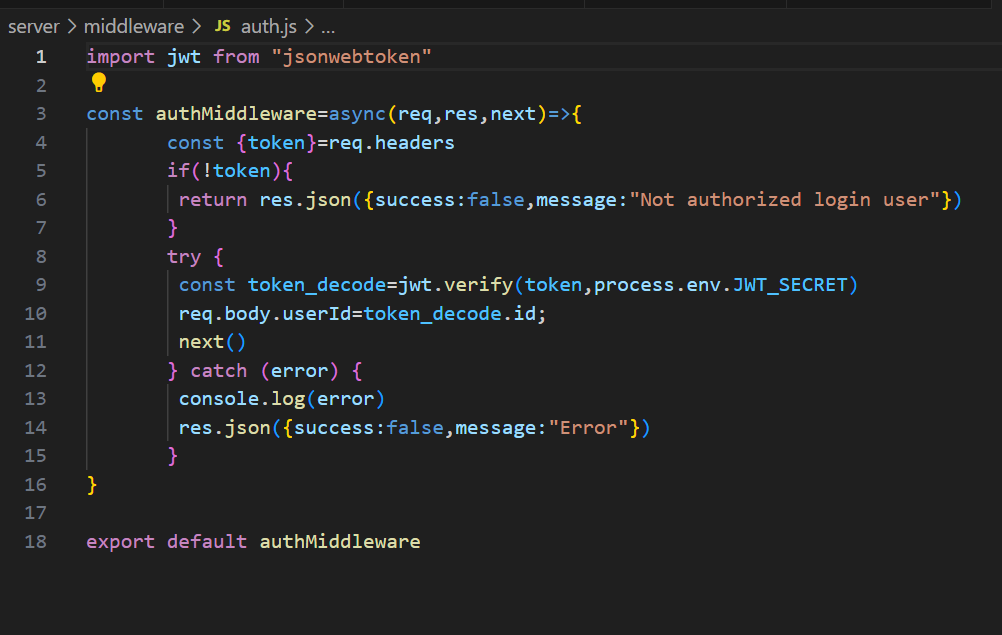
* **/** - Landing page of the application.
* **/ cart**- displays the items added in to the cart
* **/order** – getting delivery address information and calculating the total amount to be paid
* **/verify** - verify the payment being processed
* **/myOrders** – allows user to track their order status(Processing/Out for Delivery/Delivered)

Admin-side

* **/add**- uploading new dish to the menu
* **/list**- the available dishes in the menu are displayed
* **/orders –** display user orders

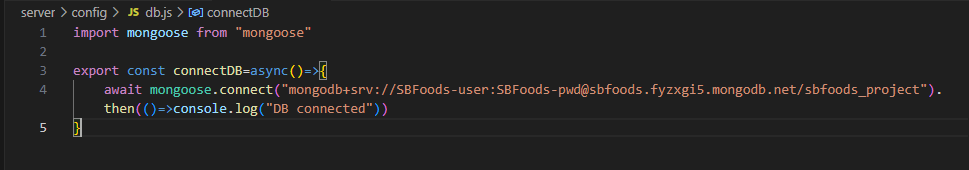
**4./middlewares**

Custom middleware functions for request processing:



**5./config**

Database Connection:



**API Endpoints**

The frontend communicates with the backend APIs hosted on backend URL. Key endpoints include:

* userRouter.post("/register", registerUser) – user registration
* userRouter.post("/login", loginUser) -user login
* foodRouter.post("/add",upload.single("image"),addFood) – updating menu by uploading image of new dish
* foodRouter.get("/list",listFood) – display dishes present in the database
* foodRouter.post("/remove",removeFood) – remove a dish from menu
* cartRouter.post('/add',authMiddleware,addToCart) – add the dish to cart
* cartRouter.post('/remove',authMiddleware,removeFromCart) – remove dish from the cart
* cartRouter.post('/get',authMiddleware,getCart) – display the items in cart
* orderRouter.post('/place',authMiddleware,placeOrder) – order being placed by user
* orderRouter.post('/verify',verifyOrder) - verify the order made by user
* orderRouter.post('/userorders',authMiddleware,userOrders) – get the orders made by a specific user
* orderRouter.get('/list',listOrders) - get the list of orders that were made
* orderRouter.post("/status",updateStatus) – update the order status(Processing/Out for Delivery/Delivered)

**Integration with Frontend**  
The backend communicates with the frontend via RESTful APIs. Key points of integration include:

* **User Authentication:** Tokens are passed between frontend and backend to handle authentication.
* **Data Fetching:** Frontend components make API calls to fetch necessary data for display and interaction.

**Error Handling and Validation**  
Describe the error handling strategy and validation mechanisms:

* **Error Handling:** Centralized error handling using middleware.
* **Validation:** Input validation using libraries like Joi or express-validation