API Development and Integration Report

Date	14 April 2025
Team ID	SWTID1743680479
Project Name	SB Foods - Food Ordering App
Maximum Marks	

Project Name: SB Foods - Food Ordering App

Project Description: A full-stack food delivery web application allows users to browse restaurant menus, place orders, and track real-time food status. The platform supports user account management, order history, and live updates via external APIs.

Project Version: v1.0.0 (Version 1)

Prepared by: Deeksha Kushwaha, Rujula Malhotra, Utkarsha Aherrao, and Anirban Mukherjee.

Objective
This report's objective is to document the API development progress and key aspects of the backend services implementation for the SB Foods Web App project.

Technologies Used

- **Frontend Framework:** React.js
- **Backend Framework:** Node.js, Express.js
- **Database:** MongoDB
- **Authentication:** JSONWebToken

Key Directories and Files

Backend:

1. **/controllers**
   - Contains functions to handle requests and responses.
   - Integrated in the server.js file
2. **/models**
   Includes Mongoose schemas and models for MongoDB collections. These Models/Schemas are:
   - userModel/userSchema
   - orderModel/orderSchema
   - foodModel/foodSchema
3. **/routes**
   - Defines the API endpoints and links them to controller functions.
   - Integrated in the server.js file in the server directory.

4. **/middlewares**
   - Custom middleware functions for request processing.

    - Cross-Origin Resource Sharing (CORS) : It is a way to enable cross-domain communication in web applications.
    - Contain middleware for user authentication
5. **/config**
    - Configuration files for database connections.
6. **/uploads**
    - Contains all the images of food uploaded by restaurants.

API Endpoints
A summary of the main API endpoints and their purposes:

User Authentication

- **POST /api/user/register** - Registers a new user and checks if user already exists. It also saves data in the MongoDB Collection with hashed password.
- **POST /api/user/login** - Authenticates the user by comparing entered credentials with stored data. If valid, returns a JWT token.

Food Items

- **POST /api/food/add-** Adds a new food item to the database. Accepts form-data including an image file using multer middleware
- **GET /api/food/list**- Retrieves and returns a list of all available food items. Typically used to display products to users in the frontend.
- **POST /api/food/remove-** Removes a food item from the database.

Cart Management

- **POST /api/cart/add** – Adds an item to the user's cart. Checks if the item exists in the user's cart, if not, adds the item with quantity 1 and if it already exists, increments the quantity by 1.
- **POST /api/cart/get** – Retrieves the cart data for the logged-in user. It fetches and returns the cart object associated with the user from the database
- **POST /api/cart/remove-** Removes an item from the user's cart. Decreases the quantity of the item by 1 and if quantity becomes 0, the item is removed from the cart.

Order Management

- **POST /api/order/place** - Places a new order for the logged-in user. It saves order details (items, amount, address) to the database linked to the authenticated user's ID.
- **POST / api/order/userorders** - Retrieves all previous orders for the logged-in user. It returns an array of the user's past orders from the database.

Frontend Integration:
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:
Describe the error-handling strategy and validation mechanisms:

- **Error Handling:** Centralized error handling using middleware. Every function has a try catch block with specific error prompts in the console.

Data Security:
outline the security measures implemented:

- **Data Encryption:** Encrypt sensitive data at rest and in transit. Using bcrypt, the passwords stored in MongoDB are hashed and safe from being spied upon by anyone who has access to the Database.