**Full Stack Development with MERN**

**API Development and Integration Report**

|  |  |
| --- | --- |
| Date | 16 April, 2025 |
| Team ID | SWTID1745000177 |
| Project Name | ShopEZ – Seamless Online Shopping Platform |
| Maximum Marks |  |

**Project Title:** ShopEZ – Seamless Online Shopping Platform  
**Date:** 15/04/25  
**Prepared by:** Anushka Singh, Ayush Kumar Jha

**Objective**  
The objective of this report is to document the API development progress and backend services implementation for **ShopEZ**, an e-commerce web application offering streamlined online shopping for buyers and efficient management for sellers.

**Technologies Used**

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

**Key Directories and Files**

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.
   * Schema.js file contains the 4 schemas of MongoDB collections:
   * UserSchema
   * ProductSchema
   * OrderSchema
   * CartSchema
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.
5. **/config**
   * Configuration files for database connections, environment variables, etc.

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

**User Authentication**

* **POST /api/user/register** - Registers a new user. Checks if user already exists. Saves data in the MongoDB Collection. The password is hashed.
* **POST /api/user/login** - Authenticates a user and returns a token. Checks if the data is present in the MongoDB Collection and compares the data input.

**User Management**

* **GET /api/user/-** Retrieves user information by ID. Using the ID, we fetch the rest of user data like email, username and password.
* **PUT /api/user/**- Updates user information by ID.

**Product Management**

* GET /api/products – Get all available products
* GET /api/products/:id – Get a product by ID
* POST /api/products – Add a new product (admin/seller only)
* PUT /api/products/:id – Update product details

**Cart & Checkout**

* POST /api/cart/add – Add product to cart
* GET /api/cart/:userId – View cart
* DELETE /api/cart/:productId – Remove item
* POST /api/checkout – Complete purchase

**Orders**

* GET /api/orders/:userId – Get user order history
* POST /api/orders – Place a new order
* PUT /api/orders/:orderId – Update order status

**Admin APIs**

* GET /api/admin/dashboard – View analytics
* GET /api/admin/orders – View all orders

**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**

* All routes are wrapped with try-catch blocks.
* Errors are logged and sent back with meaningful HTTP status codes.
* Inputs are validated with middleware to prevent malformed requests**.**

**Security Considerations**

* **Authentication:** All protected routes use JWT authentication.
* **Data Protection:** Passwords are hashed using **bcrypt** before storing.
* **CORS:** Enabled for frontend-backend communication across domains.
* **Role-based Access:** Admin and Seller roles have restricted routes.

