**PROJECT DOCUMENTATION**

**SHOPEZ**

**INTRODUCTION:**

* Project Title: SHOPEZ - E-Commerce Application
* Team Members: (Team id: NM2024TMID01287)

Naveena D (311421104053) - Team Leader

Divya Bagya Laxmi A (311421104018)

Harini M (311421104028)

Merlyn Kerenab S (311421104047)

Sanjay Kumar E (311421104310)

**PROJECT OVERVIEW:**

**Purpose:**

The purpose of ShopEZ is to redefine online shopping by offering users a seamless and enjoyable platform to explore a wide variety of products. ShopEZ is dedicated to simplifying the shopping journey with an intuitive interface, personalized recommendations, and access to diverse categories. A key goal of this project is to gain practical experience with MongoDB, showcasing its capabilities as a database solution for modern web applications.

**Why Choose ShopEZ?** We prioritize customer satisfaction, quality, and innovation, redefining the way you shop online.

**Features:**

**Admin Panel:** Admins can log in to efficiently manage the product catalog, including adding, updating, or deleting products to keep the inventory current and relevant.

**Customer Experience:** Customers can create accounts, browse products with detailed descriptions and images, add items to their cart, checkout securely, and track orders in real-time.

**Responsive Design:** The platform is optimized for seamless use across desktops, tablets, and smartphones.

**Shopping Cart & Checkout:** Users can easily manage cart items and quantities, with a streamlined checkout process for convenience.

**Order Tracking:** Customers can view and manage their order history, including status updates and delivery timelines.

**Secure Authentication:** JWT-based secure login ensures proper authorization for admins and customers.

**MongoDB Integration:** Flexible NoSQL database design supports dynamic e-commerce data, efficiently handling user details, inventory, and transactions with scalability.

**ARCHITECTURE:**

The architecture of ShopEZ is crafted to deliver scalability, maintainability, and seamless interaction between the frontend, backend, and database. It follows a modern full-stack development methodology, harnessing the capabilities of React.js, Node.js, Express.js, and MongoDB. Below is a detailed overview:

**Frontend Architecture:**

The frontend leverages React.js, a powerful JavaScript library renowned for creating efficient and dynamic user interfaces.

1. **Component-Based Structure:** The user interface is organized into reusable components such as Header, Product List, Product Detail, Cart, and Checkout, ensuring easier debugging and enhanced code reusability.
2. **State Management:** React's useState and useContext hooks efficiently manage state across components, delivering a responsive and dynamic user experience.
3. **Routing:** Navigation between pages like Home, Product Details, Cart, Checkout, and Admin Dashboard is enabled using React Router, ensuring smooth transitions.
4. **UI/UX Design:** The platform is designed with CSS, focusing on a clean, responsive layout.

The core functionalities of ShopEZ include displaying product catalogs, managing cart operations like adding, updating, and removing items, and enabling user authentication with role-based routing for admins and customers.

**Backend Architecture:**

The backend is developed using Node.js and Express.js, creating a reliable framework to handle API requests and business logic.

1. RESTful APIs: A robust set of RESTful APIs manages CRUD operations for products, users, orders, and authentication.
2. Middleware: Express.js middleware supports tasks like parsing requests, managing errors, and authenticating users.
3. Role-Based Access Control (RBAC): Middleware differentiates between customer and admin roles, restricting sensitive actions such as adding or deleting products to admin users.
4. Security: JSON Web Tokens (JWT) are used for secure authentication and session management.

Key API endpoints include:

Authentication: /login, /register

Product Management: /products, /products/:id

Order Management: /orders, /orders/:id

**Database Architecture:**

ShopEZ employs MongoDB, a NoSQL database known for its scalability and flexibility.

1. **Schema Design:** MongoDB’s dynamic schema supports flexible data storage. Major collections include:
2. **Users:** Contains details about customers and admins, including roles, authentication tokens, and profile data.
3. **Products:** Stores product information like name, description, price, category, and images.
4. **Orders:** Tracks order information, such as customer details, product IDs, quantities, order status and timestamps.
5. **Data Relationships:** Relationships between data are efficiently managed using embedded documents and references. For instance, order documents link to the product and user collections.

**Server Configuration and Connections:**

Server Deployment: The backend server, built on Node.js, utilizes Express.js for routing and middleware management.

Database Connectivity: MongoDB integrates with the backend using the MongoDB Node.js driver, with connection pooling enabled to support multiple concurrent requests.

**SETUP INSTRUCTIONS:**

This section provides step-by-step instructions to set up the ShopEZ e-commerce application locally. Please ensure you have all the necessary prerequisites installed before proceeding with the setup.

**Prerequisites:**

Before starting, confirm the following software is installed on your system:

* Node.js (version 14.x or higher): Required for running the frontend and backend servers.
  + Check installation with node -v in the terminal.
  + If not installed, download it from the [Node.js official website.](https://nodejs.org/en)
* MongoDB (version 4.x or higher): Used for storing and managing application data.
  + You can either run MongoDB locally or use a cloud-based service like [MongoDB Atlas](https://www.mongodb.com/products/platform/atlas-database).
  + Verify by running mongod --version in the terminal.
* Git (optional, but recommended): Helps in cloning the repository.
* Check installation with git --version.
* Download from the [Git official website](https://git-scm.com/).
* Code Editor: Visual Studio Code or any other editor
  + Download and install from the [VS Code Official Website](https://code.visualstudio.com/).

**Installation Steps:**

Follow these steps to set up the project locally

1. **Clone the Repository**

Start by cloning the ShopEZ repository to your local machine using Git.

git clone <https://github.com/Dnavee18/ShopEZ_E-commerce.git>

1. **Navigate to the Project Directory**

Once cloned, navigate into the project directory:

cd ShopEZ

1. **Install Backend Dependencies**

Navigate to the server directory (backend folder) and install the required dependencies using npm or yarn:

cd server

npm start

or use

yarn start

This will install dependencies like Express, Mongoose, JWT, etc., required for

the backend.

1. **Install Frontend Dependencies**

Move to the frontend folder and install required packages:

cd ../client

npm start

or use

yarn start

1. **Start the Application**

Run the backend and frontend servers:

Start the backend server

cd server

npm start

This starts the backend server at <http://localhost:6001>.

Start the frontend server in a new terminal:

cd client

npm start

or use

yarn start

This starts the React app at http://localhost:3000.

1. **Configure the Database**

For local MongoDB, ensure the MongoDB service is running.

1. **Verify the Setup**

To verify that everything is set up correctly, open your browser and go to:

▪ http://localhost:3000: To access the ShopEZ frontend.

▪ http://localhost:6001: You can use tools like Postman to test backend

API endpoints if required (e.g., /products, /orders, etc.).

**FOLDER STRUCTURE**

**Explanation of Client Folder and Files:**

**• public/:** Contains static assets like the index.html file, which serves as the entry

point to the application, as well as icons and manifest files.

**• src/:** All the source code for the frontend is here. It is further organized into subdirectories:

**components/:** Reusable components that are used throughout the app. For example, ProductCard.js displays individual products, and CartItem.js is used to show items in the shopping cart.

**context/:** Implements the React Context API for global state management. This is where the application’s global state (authentication, cart, and product data) is handled.

**pages/:** Contains components for different pages or views in the application, such as the homepage, product detail page, and checkout page.

* **styles/:** Contains global CSS files or styled-components for styling the frontend.

**Explanation of Server Folder and Files:**

**• config/:** Contains configuration files like schema.js, where MongoDB connection logic resides, and config.js for storing environment variables (e.g., JWT secret).

**• controllers/:** Functions responsible for handling incoming HTTP requests and applying business logic. Controllers handle different operations like user authentication, product management, and order processing.

**• models/:** Mongoose models that define the structure of MongoDB collections. For example, User.js defines the user schema, Product.js defines the product schema, and Order.js defines the order schema.

**• routes/:** Contains Express routes that map API endpoints to specific controller functions. For example, authRoutes.js defines routes like /login and /register.

**• middleware/:** Includes middleware functions for handling common tasks such as authentication (e.g., authMiddleware.js ensures the user is authenticated before accessing certain routes) and error handling (e.g., errorMiddleware.js catches and returns errors).

**• utils/:** Helper functions like generateToken.js, which is responsible for generating JWT tokens used in authentication.

**• index.js**: The entry point for the backend server where Express is configured and routes are connected.

**RUNNING THE APPLICATION**

Commands to start the frontend and backend servers locally.

1. **Frontend:** npm start in the client directory.
2. **Backend:** npm run start in the server directory.

**API DOCUMENTATION**

The ShopEZ backend exposes several RESTful APIs that facilitate communication

between the frontend and the database. These APIs are structured to support key features like

authentication, product management, and order processing. Below is a comprehensive list of

the available endpoints along with their descriptions, methods, parameters, and example

responses.

1.POST /api/auth/register

Registers a new user (admin or customer). The request body should contain the user's details

for creating an account.

**Request Body:**

{

"name": "John Doe",

"email": "john.doe@example.com",

"password": "password123",

"role": "customer" // 'admin' or 'customer'

}

**Response:**

• **200 OK**: Successfully registered.

{

"message": "User registered successfully!"

}

• **400 Bad Request**: Invalid or missing fields.

{

"message": "Missing required fields."

}

2.POST /api/auth/login

Logs in an existing user and returns a JWT token for subsequent requests.

**Request Body:**

{

"email": "john.doe@example.com",

"password": "password123"

}

**Response:**

**• 200 OK**: Successfully logged in.

{

"token": "jwt.token.string",

"user": {

"id": "userId",

"name": "John Doe",

"email": "john.doe@example.com",

"role": "customer"

}

}

• **401 Unauthorized**: Incorrect credentials.

{

"message": "Invalid credentials."

}

3.GET/api/Fetch-products

Retrieve all the products that are available.

**Response:**

**• 200 OK:** Products fetched

{

##Shows list of all products

}

4.GET/API/orders

Retrieve all orders of the logged-in user.

**Response:**

[

{

“id”: 6738c69064199272c65aec66

“title”: "Necklace"

“description”: "Gold Pated Classic Temple Necklace set"

“quantity”: "1"

“price”: 3000

“discount”: 50

}

]

**AUTHENTICATION**

Authentication is a crucial aspect of the ShopEZ e-commerce application, as it ensures secure access to both customer and admin features. All authentication-related actions are handled through JWT (JSON Web Tokens), which provide a stateless, secure method of user authentication and authorization.

**User Registration API**

POST /api/auth/register

Registers a new user (either customer or admin). This API allows the creation of a new account by sending the user's details in the request body. The role parameter determines whether the user is an admin or a customer.

**Request Body:**

{

"name": "John Doe",

"email": "john.doe@example.com",

"password": "password123",

"role": "customer" // 'admin' or 'customer'

}

• name: The name of the user.

• email: The user’s email address (must be unique).

• password: The user’s password (must be encrypted before storing).

• role: The role of the user. Can be either admin or customer.

**Response:**

**• 200 OK**: Successfully registered the user.

**• 400 Bad Request:** Invalid or missing fields, or if the email is already taken.

The application uses JWT for secure authentication. Users receive a token upon successful login, which is stored in cookies or local storage for session management. Role-based access control ensures only authorized users can access specific resources.

**USER INTERFACE**

The User Interface (UI) of the ShopEZ application is designed to offer a smooth and intuitive shopping experience for customers, as well as efficient management tools for admins. Built using modern web technologies, the UI is fully responsive, ensuring accessibility across various devices, from desktops to smartphones. Below is a summary of the main features and design elements.

**1. Landing Page (Home)**

The Home Page serves as the first point of interaction for users, presenting a clean and user-friendly layout with easy navigation to various sections of the application. It highlights products, promotions, and special offers.

* **Category Filters**: Enables users to explore products by category (e.g., electronics, clothing, accessories).
* **Search Bar**: Allows users to quickly search for products by name or category.

**2. Product Listing Page**

The Product Listing Page allows customers to browse available products, filter by category, and view detailed information about each product.

**3. Product Detail Page**

On the Product Detail Page, users can access detailed information about a specific product. This page offers a comprehensive view to assist customers in making informed purchasing decisions.

**Key Features:**

* **Product Images**: Multiple images of the product, including a zoom feature for a better view.
* **Product Information**: Includes the product name, description, price, and specifications.
* **Add to Cart**: A button enabling customers to add the product to their shopping cart.
* **Reviews and Ratings**: A section where users can view customer feedback and ratings.
* **Quantity Selection**: A dropdown or input box to select the quantity of items to add to the cart.

**4. Shopping Cart**

The Shopping Cart page displays all the items a customer has added. It allows users to review their choices and make changes before proceeding to checkout.

**Key Features:**

* **Item List**: Shows each item with its name, quantity, price, and total price.
* **Quantity Control**: Users can modify the quantity or remove items.
* **Total Price**: Displays the overall cost of the items in the cart.
* **Proceed to Checkout**: A button to begin the checkout process.

**5. Checkout Page**

The Checkout Page is where customers enter their shipping and payment information to complete their purchase. Its simple design ensures a smooth experience.  
**Key Features:**

* **Shipping Information**: Users provide their address details, including name, address, and phone number.
* **Payment Options**: Multiple payment methods (e.g., credit card, PayPal, Cash on delivery) are available.
* **Order Summary**: A detailed breakdown of the cart items, total price, and estimated shipping cost.
* **Place Order Button**: The final button to submit the order.

**6. User Account Dashboard (for Logged-in Users)**

The User Account Dashboard enables customers to manage their profile, track orders, and update account settings.

**7. Admin Dashboard (for Admin Users)**

The Admin Dashboard is designed for managing the ShopEZ platform. Admins can add, update, or delete products, view user information, and manage orders.

**TESTING**

Manual testing involves evaluating the application's functionality by mimicking real-user actions without relying on automated testing tools. The goal is to ensure that the application performs correctly across all features and edge cases, validating its overall performance, usability, and security.

**Testing Levels:**

* **Unit Testing**: Verifies that individual components work correctly (e.g., product display, cart operations).
* **Integration Testing**: Confirms that different modules of the application interact smoothly (e.g., frontend working with backend APIs, database).
* **System Testing**: Validates that the entire application functions properly as a whole (e.g., the end-to-end process from browsing products to completing checkout).
* **User Acceptance Testing (UAT)**: Ensures the application satisfies user needs and expectations, focusing on both customer and admin workflows.

**SCREENSHOTS OR DEMO**

**Demo link:**

**Screenshots:**

**KNOWN ISSUES**

• Admin Dashboard Performance

• Mobile Checkout Experience

• Order History Pagination

• Product Image Zoom on Mobile

• Browser Compatibility for Admin Panel

**FUTURE ENHANCEMENTS**

Looking ahead, ShopEZ plans to roll out a range of new features and improvements to elevate both the user experience and admin capabilities. Key planned upgrades include introducing multi-currency support for international customers, implementing product reviews and ratings for greater product transparency, and enhancing search and filter functionalities to improve browsing efficiency.

Furthermore, AI-powered personalized recommendations, improved mobile compatibility, and a referral program for customers are also in the works. These additions will enhance shopping convenience, boost customer engagement, and simplify administrative processes, setting ShopEZ up for continued growth and customer satisfaction.