Final Bootcamp of MYC Node Batch 2024

Proposed project name: ShopSphere

Description: ShopSphere is a modern, scalable, and robust e-commerce platform built using a microservices architecture. Designed to cater to diverse business needs, it integrates cutting-edge technologies like Node.js, Express, MongoDB, and MySQL to ensure high performance and reliability. The platform features separate microservices for critical operations such as user management, item catalog, inventory, sales reporting, and order processing.

Service List:

1. Api Gateway Service.
2. User Service.
3. Inventory Service.
4. Order Management Service.
5. Sales and Stock Reporting Service.

1. User Service

- **Database**: MongoDB
- **Features**:
    - User registration and authentication.
    - Password hashing with **bcrypt**.
    - JWT issuance for access and refresh tokens.
    - Role-based access control (Admin, Customer).
- **Specification**:
    - Use Standard MVC architecture with router, controller, service, repository, config etc modules.
    - Write proper test cases to cover the complete service operations.
    - Follow standard code guidelines

Tech Stack:

1. **Node JS.**
2. **Express JS.**
3. **JWT**
4. **Redis**
5. **Mongo DB**
6. **Node Mailer**
7. **BCrypt**

Table Design:

1.TBL_USER_DETAILS

Field Name	Description	Data Type	Example Value
_id	Unique identifier for the user (auto-generated by MongoDB)	ObjectId	ObjectId("user123")
first_name	First name of the user	String	"John"
last_name	Last name of the user	String	"Doe"
email	User's email address (unique)	String	"johndoe@example.com"
password	Hashed password for authentication	String	"$2b$12$Vxi4H..."
phone	User's phone number	String	"+1234567890"
addresses	Array of address objects associated with the user	Array of Objects	[{...}, {...}]
role	Role of the user (e.g., customer, admin)	String	"customer"
status	t status (e.g., active, inactive, suspended)	String	"active"
created_at	Timestamp when the user was created	ISODate	"2024-11-22T10:00:00Z"

Field Name	Description	Data Type	Example Value
updated_at	Timestamp when the user was last updated	ISODate	"2024-11-22T10:00:00Z"

2.TBL_ADDRESS_DETAILS

Field Name	Description	Data Type	Example Value
_id	Unique identifier for the address	ObjectId	ObjectId("address123")
address_type	Type of the address (e.g., home, office, billing)	String	"Home"
street	Street name and number	String	"123 Elm Street"
city	City name	String	"New York"
state	State or region name	String	"NY"
zip	ZIP or postal code	String	"10001"
country	Country name	String	"USA"
is_primary	Indicates whether this is the user's primary address	Boolean	true

3.TBL_ROLE_DETAILS

Field Name	Description	Data Type	Example Value
_id	Unique identifier for the role	ObjectId	ObjectId("role123")
role_name	Name of the role (e.g., admin, customer, merchant)	String	"admin"
description	User role description	Strings	“Administration”

User Service REST APIs

Endpoint	HTTP Method	Description	Response Code	Error Codes
/api/users/register **ADMIN & CUSTOMER Access**	POST	Register a new user	201 Created	500 Internal Server Error
/api/users/login **ADMIN & CUSTOMER Access**	POST	User login (Generate access and refresh tokens)	200 OK	400 Bad Request (Invalid credentials)
/api/users/logout **ADMIN & CUSTOMER Access**	POST	User logout (Invalidate the refresh token)	200 OK	400 Bad Request (Token expired)

Endpoint	HTTP Method	Description	Response Code	Error Codes
/api/users/{userId} **ADMIN & CUSTOMER Access**	GET	Get user details by user ID. Should come from cache if present. Cache lifetime 1 hour	200 OK	404 Not Found (User not found)
/api/users/ **ADMIN Access**	GET	Get all users	200 OK	500
/api/users/{userId} **CUSTOMER Access**	PUT	Update user details (email, phone, address, etc.) Have impact on running cache.	200 OK	400 Bad Request (Invalid data)
/api/users/{userId} **ADMIN & CUSTOMER Access**	DELETE	Delete user account. Also delete the existing cache.	200 OK	404 Not Found (User not found)
/api/users/roles **ADMIN Access**	GET	Get all available roles	200 OK	500 Internal Server Error
/api/users/forgot-password **Need to mail the existing pwd to user** **CUSTOMER Access**	POST	Request password reset email	200 OK	400 Bad Request (Invalid email format)
/api/users/reset-password	PUT	Reset user password using token	200 OK	400 Bad Request (Invalid token)

2. Inventory Service

- **Database**: My SQL
- **Features**:
    - Add and manage products by Admin.
    - Retrieve product listings with filters and sorting.
- **Specification**:
    - Use Standard MVC architecture with router, controller, service, repository, config etc modules.
    - Write proper test cases to cover the complete service operations.
    - Follow standard code guidelines

Tech Stack: Node JS,
Express,
My SQL,
File Management (fs)

Need to manage item bulk upload by excel format (Secondary task)

Table Design:

1.TBL_ITEM_DETAILS

Attribute Name	**Data Type**	**Description**	**Constraints**
item_id	INT (AUTO_INCREMENT)	Unique identifier for the item.	PRIMARY KEY
item_name	VARCHAR (255)	Name of the item.	NOT NULL
Description	TEXT	Detailed description of the item.	NULL
Price	DECIMAL (10, 2)	Price of the item.	NOT NULL
stock_quantity	INT	Number of items available in stock.	NOT NULL
Status	ENUM ('available', 'out_of_stock', 'discontinued')	Current status of the item.	NOT NULL
size_id	INT	Size of the item (foreign key to tbl_item_size).	FOREIGN KEY (size_id)
brand_id	INT	Brand of the item (foreign key to tbl_item_brands).	FOREIGN KEY (brand_id)
color_id	INT	Color of item (foreign key to tbl_item_color)	FOREIGN KEY (color_id)
created_at	DATETIME	Timestamp when the item was added.	NOT NULL

Attribute Name	Data Type	Description	Constraints
updated_at	DATETIME	Timestamp when the item was last updated.	NULL

2.TBL_ITEM_BRAND

Attribute Name	Data Type	Description	Constraints
brand_id	INT (AUTO_INCREMENT)	Unique identifier for the brand.	PRIMARY KEY
brand_name	VARCHAR(100)	Name of the brand (e.g., Apple, Nike).	NOT NULL
created_at	DATETIME	Timestamp when the item was added.	NOT NULL
updated_at	DATETIME	Timestamp when the item was last updated.	NULL

3.TBL_ITEM_COLOR

Attribute Name	Data Type	Description	Constraints
color_id	INT (AUTO_INCREMENT)	Unique identifier for the color.	PRIMARY KEY
color_name	VARCHAR(100)	Name of color	NOT NULL
created_at	DATETIME	Timestamp when the item was added.	NOT NULL
updated_at	DATETIME	Timestamp when the item was last updated.	NULL

4.TBL_ITEM_SIZE

Attribute Name	Data Type	Description	Constraints
size_id	INT (AUTO_INCREMENT)	Unique identifier for the color.	PRIMARY KEY
size_types	VARCHAR(100)	Size Type	NOT NULL
Size_value	VARCHAR (100)	Size value	NOT NULL
created_at	DATETIME	Timestamp when the item was added.	NOT NULL
updated_at	DATETIME	Timestamp when the item was last updated.	NULL

5.TBL_ITEM_IMAGES

Attribute Name	Data Type	Description	Constraints
image_id	INT (AUTO_INCREMENT)	Unique identifier for the image.	PRIMARY KEY
image_name	VARCHAR (100)	Size Type	NOT NULL
image_location	VARCHAR (500)	Size value	NOT NULL
created_at	DATETIME	Timestamp when the item was added.	NOT NULL
updated_at	DATETIME	Timestamp when the item was last updated.	NULL

REST APIs for Inventory Management Service

Here is a list of the REST services for the **Inventory Management Service** along with their HTTP methods, response codes, and error codes.

Endpoint	HTTP Method	Description	Response Code	Error Codes
/api/inventory/items **ADMIN & CUSTOMER Access**	GET	Get all items in the inventory.	200 OK	404 Not Found (No items found)
/api/inventory/items/{itemId} **ADMIN & CUSTOMER Access**	GET	Get details of a specific item by item ID. Get from Cache if present. Cache lifetime 1 hour.	200 OK	404 Not Found (Item not found)
/api/inventory/items **ADMIN Access**	POST	Add a new item to the inventory.	201 Created	500 Internal Server Error (Invalid data)
/api/inventory/items/{itemId} **ADMIN Access**	PUT	Update an existing item in the inventory.	200 OK	404 Not Found (Item not found), 400 Bad Request (Invalid data)
/api/inventory/items/{itemId} **ADMIN Access**	DELETE	Delete an item from the inventory.	200 OK	404 Not Found (Item not found)
/api/inventory/items/{itemId}/images **ADMIN Access**	POST	Add an image to a product.	201 Created	400 Bad Request (Invalid image format)

Endpoint	HTTP Method	Description	Response Code	Error Codes
/api/inventory/items/{itemId}/images/{imageId} **ADMIN Access**	DELETE	Delete an image associated with a product.	200 OK	404 Not Found (Image not found)
/api/brands **ADMIN Access**	GET	Get all available brands.	200 OK	404 Not Found (No brands found)
/api/brands/{brandId} **ADMIN Access**	GET	Get details of a specific brand by brand ID. Get from Cache if present. Cache lifetime 1 hour.	200 OK	404 Not Found (Brand not found)
/api/brands **ADMIN Access**	POST	Add a new brand to the system.	201 Created	400 Bad Request (Invalid data), 500 Internal Server Error
/api/brands/{brandId} **ADMIN Access**	PUT	Update an existing brand in the system.	200 OK	404 Not Found (Brand not found), 400 Bad Request (Invalid data)
/api/brands/{brandId} **ADMIN Access**	DELETE	Delete a brand from the system.	200 OK	404 Not Found (Brand not found)

Endpoint ADMIN Access	HTTP Method	Description	Response Code	Error Codes
/api/sizes	GET	Get all available sizes.	200 OK	404 Not Found (No sizes found)
/api/sizes/{sizeId}	GET	Get details of a specific size by size ID. Get from Cache if present. Cache lifetime 1 hour.	200 OK	404 Not Found (Size not found)
/api/sizes	POST	Add a new size to the system.	201 Created	400 Bad Request (Invalid data), 500 Internal Server Error
/api/sizes/{sizeId}	PUT	Update an existing size in the system.	200 OK	404 Not Found (Size not found), 400 Bad Request (Invalid data)

Endpoint ADMIN Access	HTTP Method	Description	Response Code	Error Codes
/api/sizes/{sizeId}	DELETE	Delete a size from the system.	200 OK	404 Not Found (Size not found)

Endpoint ADMIN Access	HTTP Method	Description	Response Code	Error Codes
/api/colors	GET	Get all available colors.	200 OK	404 Not Found (No colors found)
/api/colors/{colorId}	GET	Get details of a specific color by color ID. Get from Cache if present. Cache lifetime 1 hour.	200 OK	404 Not Found (Color not found)
/api/colors	POST	Add a new color to the system.	201 Created	400 Bad Request (Invalid data), 500 Internal Server Error
/api/colors/{colorId}	PUT	Update an existing color in the system.	200 OK	404 Not Found (Color not found), 400 Bad Request (Invalid data)
/api/colors/{colorId}	DELETE	Delete a color from the system.	200 OK	404 Not Found (Color not found)

3. Order Service

- **Database**: **My SQL**
- **Features**:
    - Customer manage Wishlist
    - Customer manage Cart.
    - Customer placed order and process the payment.
    - Role-based access control (Admin, Customer).
- **Specification**:
    - Use Standard MVC architecture with router, controller, service, repository, config etc modules.
    - Write proper test cases to cover the complete service operations.
    - Follow standard code guidelines

1. TBL_ORDERS

Attribute Name	**Data Type**	**Description**	**Constraints**
order_id	INT (AUTO_INCREMENT)	Unique identifier for the order.	PRIMARY KEY
user_id	INT	Foreign key to the user placing the order.	FOREIGN KEY (references users.user_id)
order_date	DATETIME	Date and time when the order was placed.	NOT NULL
status	VARCHAR(50)	Current status of the order (e.g., pending, shipped, delivered).	NOT NULL
total_amount	DECIMAL(10,2)	Total cost of the order.	NOT NULL
shipping_address_id	VARCHAR(255)	Shipping address id	NOT NULL
billing_address_id	VARCHAR(255)	Billing address id for the order.	NOT NULL
payment_status	VARCHAR(50)	Payment status (e.g., pending, paid).	NOT NULL

2. TBL_ORDER_ITEMS

Attribute Name	Data Type	Description	Constraints
order_item_id	INT (AUTO_INCREMENT)	Unique identifier for the order item.	PRIMARY KEY
order_id	INT	Foreign key to the associated order.	FOREIGN KEY (references orders.order_id)
item_id	INT	Foreign key to the item being purchased.	FOREIGN KEY (references items.item_id)
quantity	INT	Quantity of the item in the order.	NOT NULL
price	DECIMAL(10,2)	Price of a single unit of the item.	NOT NULL
total_price	DECIMAL(10,2)	Total price for the order item (quantity * price).	NOT NULL

3. TBL_ORDER_PAYMEMNT

Attribute Name	Data Type	Description	Constraints
payment_id	INT (AUTO_INCREMENT)	Unique identifier for the payment.	PRIMARY KEY
order_id	INT	Foreign key to the associated order.	FOREIGN KEY (references orders.order_id)
payment_method	VARCHAR(50)	Payment method used (e.g., credit card, PayPal).	NOT NULL
payment_date	DATETIME	Date and time when the payment was made.	NOT NULL
amount	DECIMAL(10,2)	Amount paid for the order.	NOT NULL
payment_status	VARCHAR(50)	Payment status (e.g., success, failed).	NOT NULL

4. TBL_SHIPPING

Attribute Name	Data Type	Description	Constraints
shipping_id	INT (AUTO_INCREMENT)	Unique identifier for the shipping entry.	PRIMARY KEY
order_id	INT	Foreign key to the associated order.	FOREIGN KEY (references orders.order_id)

Attribute Name	Data Type	Description	Constraints
shipping_method	VARCHAR(50)	Shipping method (e.g., standard, express).	NOT NULL
shipping_date	DATETIME	Date and time when the order was shipped.	NOT NULL
delivery_date	DATETIME	Date and time when the order was delivered.	NULL
shipping_status	VARCHAR(50)	Current status of shipping (e.g., pending, in transit, delivered).	NOT NULL

5. TBL_CART

Attribute Name	Data Type	Description	Constraints
cart_id	INT (AUTO_INCREMENT)	Unique identifier for the cart.	PRIMARY KEY
user_id	INT	Foreign key to the user who owns the cart.	FOREIGN KEY (references users.user_id)
created_at	DATETIME	Timestamp when the cart was created.	NOT NULL
updated_at	DATETIME	Timestamp when the cart was last updated.	NOT NULL

6.TBL_CART_ITEMS

Attribute Name	Data Type	Description	Constraints
cart_item_id	INT (AUTO_INCREMENT)	Unique identifier for the cart item.	PRIMARY KEY
cart_id	INT	Foreign key to the associated cart.	FOREIGN KEY (references cart.cart_id)
item_id	INT	Foreign key to the item added to the cart.	FOREIGN KEY (references items.item_id)
Quantity	INT	Quantity of the item in the cart.	NOT NULL

7.TBL_WISHLIST

Attribute Name	Data Type	Description	Constraints
wishlist_id	INT (AUTO_INCREMENT)	Unique identifier for the wishlist.	PRIMARY KEY
user_id	INT	Foreign key to the user who owns the wishlist.	FOREIGN KEY (references users.user_id)
created_at	DATETIME	Timestamp when the wishlist was created.	NOT NULL

8.TBL_WISHLIST_ITEMS

Attribute Name	Data Type	Description	Constraints
wishlist_item_id	INT (AUTO_INCREMENT)	Unique identifier for the wishlist item.	PRIMARY KEY
wishlist_id	INT	Foreign key to the associated wishlist.	FOREIGN KEY (references wishlist.wishlist_id)
item_id	INT	Foreign key to the item in the wishlist.	FOREIGN KEY (references items.item_id)

Order Management REST APIs

Endpoint CUSTIMER Access	HTTP Method	Description	Response Code	Error Codes
/api/orders	GET	Get all orders for the authenticated user.	200 OK	404 Not Found (No orders found)
/api/orders/{orderId}	GET	Get details of a specific order by order ID.	200 OK	404 Not Found (Order not found)
/api/orders	POST	Place a new order.	201 Created	400 Bad Request (Invalid data), 500 Internal Server Error
/api/orders/{orderId}	PUT	Update an existing order (e.g., status change).	200 OK	404 Not Found (Order not found), 400 Bad Request (Invalid data)

Endpoint CUSTIMER Access	HTTP Method	Description	Response Code	Error Codes
/api/orders/{orderId}	DELETE	Cancel an existing order.	200 OK	404 Not Found (Order not found)
/api/orders/{orderId}/items	GET	Get all items in a specific order.	200 OK	404 Not Found (Order not found)
/api/orders/{orderId}/payment	POST	Make a payment for an order.	201 Created	400 Bad Request (Invalid data), 500 Internal Server Error
/api/orders/{orderId}/shipping ADMIN Access	POST	Add shipping details for the order.	201 Created	400 Bad Request (Invalid data), 404 Not Found (Order not found)
/api/orders/{orderId}/shipping/status ADMIN Access	PUT	Update shipping status for an order.	200 OK	404 Not Found (Order not found), 400 Bad Request (Invalid status)

Cart Management REST APIs

Endpoint	HTTP Method	Description	Response Code	Error Codes
/api/cart	GET	Get all items in the cart for the authenticated user.	200 OK	404 Not Found (No items found in cart)
/api/cart	POST	Create a new cart for the authenticated user.	201 Created	400 Bad Request (Invalid data)
/api/cart/{cartId}/items	GET	Get all items in a specific cart.	200 OK	404 Not Found (Cart not found)
/api/cart/{cartId}/items	POST	Add an item to the cart.	201 Created	400 Bad Request (Invalid data)
/api/cart/{cartId}/items/{itemId}	DELETE	Remove an item from the cart.	200 OK	404 Not Found (Item or Cart not found)
/api/cart/{cartId}/checkout	POST	Checkout the cart and place an order.	200 OK	404 Not Found (Cart not found), 400 Bad Request (Invalid data)

Wishlist Management REST APIs

Endpoint	HTTP Method	Description	Response Code	Error Codes
/api/wishlist	GET	Get all items in the wishlist for the authenticated user.	200 OK	404 Not Found (No items found in wishlist)
/api/wishlist	POST	Create a new wishlist for the authenticated user.	201 Created	400 Bad Request (Invalid data)
/api/wishlist/{wishlistId}/items	GET	Get all items in a specific wishlist.	200 OK	404 Not Found (Wishlist not found)
/api/wishlist/{wishlistId}/items	POST	Add an item to the wishlist.	201 Created	400 Bad Request (Invalid data)
/api/wishlist/{wishlistId}/items/{itemId}	DELETE	Remove an item from the wishlist.	200 OK	404 Not Found (Item or Wishlist not found)

---

4. Sales and Stock Reporting Service.

- **Database**: **MongoDB**
- **Features**:
    - Sales Tracking: Provides insights into revenue and top-performing items.
    - Stock Monitoring: Ensures sufficient inventory levels, preventing stockouts or overstocking.
    - Stock Movements: Logs every change in stock, enabling better accountability.
    - Top-Selling Items: Helps identify high-demand products for better marketing strategies.
    - Real-Time Updates: Updates sales and stock information dynamically for accurate reporting.
- **Specification**:
    - Use Standard MVC architecture with router, controller, service, repository, config etc modules.
    - Write proper test cases to cover the complete service operations.
    - Follow standard code guidelines

TBL_SALES

Attribute Name	**Data Type**	**Description**	**Constraints**
sale_id	INT (AUTO_INCREMENT)	Unique identifier for each sale.	PRIMARY KEY
item_id	INT	Foreign key to the sold item.	FOREIGN KEY (references items.item_id)
quantity_sold	INT	Number of items sold.	NOT NULL
sale_date	DATETIME	Timestamp of the sale.	NOT NULL
total_revenue	DECIMAL(10,2)	Total revenue generated from the sale.	NOT NULL

TBL_STOCK

Attribute Name	**Data Type**	**Description**	**Constraints**
stock_id	INT (AUTO_INCREMENT)	Unique identifier for stock entries.	PRIMARY KEY
item_id	INT	Foreign key to the associated item.	FOREIGN KEY (references items.item_id)
quantity_in_stock	INT	Current quantity available in stock.	NOT NULL
last_updated	DATETIME	Timestamp of the last stock update.	NOT NULL

TBL_STOCK_MOVEMENT

Attribute Name	Data Type	Description	Constraints
movement_id	INT (AUTO_INCREMENT)	Unique identifier for stock movement.	PRIMARY KEY
item_id	INT	Foreign key to the associated item.	FOREIGN KEY (references items.item_id)
movement_type	VARCHAR(50)	Type of stock movement (e.g., "restock", "sale").	NOT NULL
Quantity	INT	Quantity involved in the stock movement.	NOT NULL
movement_date	DATETIME	Timestamp of the stock movement.	NOT NULL

REST APIs for Sales and Stock Reporting

Endpoint ADMIN	HTTP Method	Description	Response Code	Error Codes
/api/sales	GET	Get all sales records.	200 OK	404 Not Found (No sales found)
/api/sales/{saleId}	GET	Get details of a specific sale by sale ID.	200 OK	404 Not Found (Sale not found)
/api/sales	POST	Record a new sale.	201 Created	400 Bad Request (Invalid data)
/api/stock	GET	Get the current stock levels for all items.	200 OK	404 Not Found (No stock data available)
/api/stock/{itemId}	GET	Get stock details for a specific item.	200 OK	404 Not Found (Item not found)
/api/stock	PUT	Update stock levels (e.g., after restocking).	200 OK	400 Bad Request (Invalid data)
/api/stock/movements	GET	Get all stock movements.	200 OK	404 Not Found (No movements found)
/api/stock/movements/{movementId}	GET	Get details of a specific stock movement.	200 OK	404 Not Found (Movement not found)

Endpoint ADMIN	HTTP Method	Description	Response Code	Error Codes
/api/stock/movements	POST	Record a new stock movement (e.g., restock or sale).	201 Created	400 Bad Request (Invalid data)

Key Features of Sales and Stock Reporting Service

1. **Sales Tracking**: Provides insights into revenue and top-performing items.
2. **Stock Monitoring**: Ensures sufficient inventory levels, preventing stockouts or overstocking.
3. **Stock Movements**: Logs every change in stock, enabling better accountability.
4. **Real-Time Updates**: Updates sales and stock information dynamically for accurate reporting.