# Project Title: ToyStation

## Deliverable 3 – Phase 1

CSCE 5430 (Fall 2024)

**Group Name: Group-3**

**Core Functionalities: Code Inspection Document**

**1. User Registration and Authentication**

**Description:** This module handles user registration, login, and authentication. It ensures secure login by encrypting user passwords using bcrypt.js and authenticates users via JWT tokens.

**Source Code:**

```
const mongoose = require('mongoose');

const UserSchema = new mongoose.Schema({
  username: {
    type: String,
    required: true,
  },
  email: {
    type: String,
    required: true,
    unique: true,
  },
  password: {
    type: String,
    required: true,
  },
});

module.exports = mongoose.model('User', UserSchema);
```

**Description:**

- The user schema stores user information.
- Passwords are encrypted using bcrypt before saving them.
- JWT tokens are generated to authenticate users securely.

---

## 2. Product Management

**Description:** This module allows users to view, search, and filter products. It interacts with the MongoDB database to retrieve product information.

**Source Code:**

```
import React, { useState } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { Button, Card, Col, Container, Row, Form, Dropdown } from 'react-bootstrap';
import { addToCart } from '../redux/actions/cartActions';
import NavbarTop from './NavbarTop';
import Footer from './Footer';

const ProductList = [
  {
    id: 1,
    name: 'Toy Car',
    description: 'A cool toy car for kids',
    price: 15,
    image:
'https://isakaabengaluru.s3.ap-south-1.amazonaws.com/wp-content/uploads/2022/04/29125018/71dEW
XwH0cL._SL1500_.jpg',
    category: 'Toys',
  },
  {
    id: 2,
    name: 'Toy Train',
    description: 'A toy train for kids',
    price: 20,
    image: 'https://m.media-amazon.com/images/I/511sKmv7JHL.jpg',
    category: 'Toys',
  },
  {
    id: 3,
    name: 'Toy Plane',
    description: 'A toy plane for kids',
    price: 25,
    image:
'https://rukminim2.flixcart.com/image/850/1000/kfk0e4w0/vehicle-pull-along/z/d/h/musical-air-p
lane-running-with-music-not-flying-white-topup-original-imafvzfvkhxu5apr.jpeg?q=90&crop=false'
,
    category: 'Toys',
  },
  {
    id: 4,
    name: 'Toy Robot',
    description: 'A toy robot for kids',
    price: 30,
    image: 'https://m.media-amazon.com/images/I/71j6OP1zHPL.jpg',
    category: 'Toys',
  },
  {
    id: 5,
    name: 'Toy Doll',
    description: 'A toy doll for kids',
    price: 35,
    image: 'https://images.meesho.com/images/products/393169341/ueedc_512.webp',
    category: 'Toys',
  },
  {
    id: 6,
    name: 'Toy Truck',
    description: 'A toy truck for kids',
    price: 40,
    image: 'https://toyzone.in/cdn/shop/products/72522-02_2048x.jpg?v=1662612960',
    category: 'Toys',
  },
  {
    id: 7,
    name: 'Football',
    description: 'Football for kids',
    price: 45,
    image:
'https://mmtoyworld.com/cdn/shop/files/mm-toys-high-quality-pvc-football-3-no-size-for-age-3-t
o-10-years-pack-of-1-pc-free-inflating-pin-multicolor_1.jpg?v=1684300546',
    category: 'Sports',
  },
  {
    id: 8,
    name: 'Basketball',
    description: 'Basketball for kids',
    price: 50,
    image:
'https://cdn.pixelbin.io/v2/black-bread-289bfa/HrdP6X/original/hamleys-product/491603946/665/4
91603946-1_3253.webp',
    category: 'Sports',
  },
  {
    id: 9,
    name: 'Cricket Bat',
    description: 'Cricket bat for kids',
    price: 55,
    image:
'https://gmcricket.in/media/catalog/product/cache/757ea7d2b7282843694bdb6de7a23598/d/i/diamond
-606-english-willow-cricket-bat_9.jpg',
    category: 'Sports',
  }
]

const Products = () => {
  const dispatch = useDispatch();
  const cartItems = useSelector((state) => state.cart.cartItems);

  // States for search and filter
  const [searchTerm, setSearchTerm] = useState('');
  const [selectedCategory, setSelectedCategory] = useState('All');

  const handleAddToCart = (product, quantity) => {
    dispatch(addToCart(product, quantity));
  };

  const handleQuantityChange = (product, change) => {
    const existingItem = cartItems.find((item) => item.id === product.id);
    const newQty = existingItem ? existingItem.quantity + change : 1;

    if (newQty > 0) {
      dispatch(addToCart(product, newQty));
    }
  };

  // Filter products by search term and category
  const filteredProducts = ProductList.filter((product) => {
    const matchesCategory = selectedCategory === 'All' || product.category ===
selectedCategory;
    const matchesSearch = product.name.toLowerCase().includes(searchTerm.toLowerCase());
    return matchesCategory && matchesSearch;
  });

  return (
    <>
      <NavbarTop />
      <Container>
        {/* Search and Filter Section */}
        <Row className="mt-4 mb-4">
          <Col md={6}>
            <Form.Control
              type="text"
              placeholder="Search for products..."
              value={searchTerm}
              onChange={(e) => setSearchTerm(e.target.value)}
            />
          </Col>
          <Col md={6}>
            <Form.Control as="select" value={selectedCategory} onChange={(e) =>
setSelectedCategory(e.target.value)}>
              <option value="All">All Categories</option>
              <option value="Toys">Toys</option>
              <option value="Sports">Sports</option>
            </Form.Control>
          </Col>
        </Row>

        {/* Products Display Section */}
        <Row>
          {filteredProducts.length === 0 ? (
            <p>No products found</p>
          ) : (
            filteredProducts.map((product) => {
              const existingItem = cartItems.find((item) => item.id === product.id);
              const quantity = existingItem ? existingItem.quantity : 0;

              return (
                <Col key={product.id} md={4} className="mb-4" style={{padding:"20px",}}>
                  <Card className="product-card shadow-sm">
                    <Card.Img variant="top" src={product.image} className="product-image"
style={{width:"100%",}}/>
                    <Card.Body>
                      <Card.Title>{product.name}</Card.Title>
                      <Card.Text>{product.description}</Card.Text>
                      <Card.Text><strong>${product.price}</strong></Card.Text>

                      {quantity > 0 ? (
                        <div className="d-flex align-items-center justify-content-between">
                          <Button variant="danger" onClick={() =>
handleQuantityChange(product, -1)}>
                            -
                          </Button>
                          <span className="mx-2">{quantity}</span>
                          <Button variant="success" onClick={() =>
handleQuantityChange(product, 1)}>
                            +
                          </Button>
                        </div>
                      ) : (
                        <Button variant="primary" onClick={() => handleAddToCart(product, 1)}>
                          Add to Cart
                        </Button>
                      )}
                    </Card.Body>
                  </Card>
                </Col>
              );
            })
          )}
        </Row>
      </Container>
      <Footer />
    </>
  );
};
```

```
export default Products;
```

**Description:**

- The `ProductSchema` defines the structure for product information.
- The `getProducts` function fetches products from the database and allows filtering by category and price range

---

**3. Shopping Cart Functionality**

**Description:** The shopping cart allows users to add products, view the cart, and proceed to checkout.

**Source Code:**

```
import React from 'react';
```

```
import { useDispatch, useSelector } from 'react-redux';

import { Button, Row, Col, ListGroup, Container, Card } from 'react-bootstrap';

import { removeFromCart, updateCartQuantity, clearCart } from
'../redux/actions/cartActions';

import NavbarTop from './NavbarTop';

import axios from 'axios';

import { useNavigate } from 'react-router-dom';

import Footer from './Footer';



const Cart = () => {

  const dispatch = useDispatch();

  const navigate = useNavigate(); // Initialize useNavigate hook

  const cartItems = useSelector((state) => state.cart.cartItems);

  const handleRemoveFromCart = (id) => {

    dispatch(removeFromCart(id));

  };
  const handleQuantityChange = (id, quantity) => {

    if (quantity > 0) {

      dispatch(updateCartQuantity(id, quantity));

    }

  };



  const totalAmount = cartItems.reduce((acc, item) => acc + item.price *
item.quantity, 0).toFixed(2);



  const handlePlaceOrder = async () => {

    const userEmail = localStorage.getItem('email');



    if (!userEmail) {

      alert('Please log in to place an order.');

      return;

    }



    try {

      const orderData = {

        user: userEmail,

        orderItems: cartItems.map(item => ({

          name: item.name,

          quantity: item.quantity,

          price: item.price,

          total: (item.price * item.quantity),

        })),
        totalAmount: totalAmount,

      };


      await axios.post('/api/orders/create', orderData); // Ensure this URL is correct

      dispatch(clearCart()); // Clear the cart after order is placed

      alert('Order placed successfully!');

    } catch (error) {

      console.error('Error placing order:', error.response?.data || error.message);

      alert('Failed to place order. Please try again.');

    }

  };


  return (

    <>

      <NavbarTop />

      <Container className="my-4">

        <Row>

          <Col md={8}>

            <h2>Your Cart</h2>

            <ListGroup variant="flush">

              {cartItems.length === 0 ? (

                <ListGroup.Item>Your cart is empty</ListGroup.Item>

              ) : (

                cartItems.map((item) => {

                  const itemTotal = (item.price * item.quantity).toFixed(2);
                  return (
                    <ListGroup.Item key={item.id}>
                      <Row className="align-items-center">
                        <Col md={4} className="d-flex align-items-center">
                          <img src={item.image} alt={item.name} className="cart-image"
/>
                          <span className="ms-2">{item.name}</span>
                        </Col>
                        <Col md={4} className="d-flex align-items-center">
                          <Button
                            variant="danger"
                            onClick={() => handleQuantityChange(item.id, item.quantity
- 1)}
                            disabled={item.quantity <= 1}
                          >
                            -
                          </Button>
                          <span className="mx-2">{item.quantity}</span>
                          <Button
                            variant="success"
                            onClick={() => handleQuantityChange(item.id, item.quantity
+ 1)}
                          >
                            +
                          </Button>
                        </Col>
                        <Col md={2}>${item.price.toFixed(2)}</Col>
                        <Col md={2}>${itemTotal}</Col>

                        <Col md={2}>

                          <Button

                            variant="danger"

                            onClick={() => handleRemoveFromCart(item.id)}

                          >

                            Remove

                          </Button>

                        </Col>

                      </Row>

                    </ListGroup.Item>

                  );

                })

              )}

            </ListGroup>

          </Col>

          <Col md={4}>

            <Card className="mt-4">

              <Card.Header as="h5">Cart Summary</Card.Header>

              <Card.Body>

                <Card.Text>

                  <strong>Total Amount:</strong> ${totalAmount}

                </Card.Text>

                <Button variant="primary" className="w-100"
onClick={handlePlaceOrder}>

                  Place order
                </Button>

              </Card.Body>

            </Card>

          </Col>

        </Row>

      </Container>

      <Footer />

    </>

  );

};

export default Cart
```

**Description:**

- The `CartSchema` tracks user-selected products.
- The `addToCart` function allows users to add items to their cart or update the quantity of existing items.

---

**4. Order Management and Checkout**

**Description:** This module allows users to place orders and tracks their status.

**Source Code:**

```
const mongoose = require('mongoose');

const orderSchema = new mongoose.Schema({
  user: {
    type: String,
    required: true, // Assuming user will be stored as email
  },
  orderItems: [
    {
      name: { type: String, required: true },
      quantity: { type: Number, required: true },
      price: { type: Number, required: true },
      total: { type: Number, required: true },
    }
  ],
  totalAmount: {
    type: Number,
    required: true,
  },
  createdAt: {
    type: Date,
    default: Date.now,
  }
});

const Order = mongoose.model('Order', orderSchema);
module.exports = Order;
```

**Description:**

- The `OrderSchema` stores details of the user's orders.
- The `createOrder` function handles the creation of a new order when the user proceeds to checkout.

---

**5. Backend and Frontend Integration**

**Description:** The backend and frontend communicate using RESTful APIs. The backend handles all user requests and interactions with the database, while the frontend displays data to users.

**Source Code (Backend - Example API Endpoint):**

**Source Code (Frontend - Fetch User Data):**

```
const User = require('../models/User');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');

// Register a new user
exports.register = async (req, res) => {
  const { username, email, password } = req.body;

  try {
    // Check if the user already exists
    let user = await User.findOne({ email });
    if (user) {
      return res.status(400).json({ msg: 'User already exists' });
    }

    // Create a new user
    user = new User({
      username,
      email,
      password: await bcrypt.hash(password, 10),
    });

    await user.save();

    res.status(201).json({ msg: 'User registered successfully' });
  } catch (error) {
    console.error(error);
    res.status(500).json({ msg: 'Server error' });
  }
};

// Login user
exports.login = async (req, res) => {
  const { email, password } = req.body;

  try {
    const user = await User.findOne({ email });
    if (!user) {
      return res.status(400).json({ msg: 'Invalid credentials' });
    }

    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) {
      return res.status(400).json({ msg: 'Invalid credentials' });
    }

    // Create and assign a token
    const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET, {
      expiresIn: '1h',
    });

    res.json({ email: user.email, token });
  } catch (error) {
    console.error(error);
    res.status(500).json({ msg: 'Server error' });
  }
};
```

**Description:**

- The backend API serves user data, which is fetched and displayed on the frontend using React components.