Error Handling

Assignment-7 Solution

1. You are developing the error handling mechanism for an online shopping cart application. The application allows users to add products to their cart and proceed to checkout. Implement error handling to address different types of errors that might occur during the shopping process.

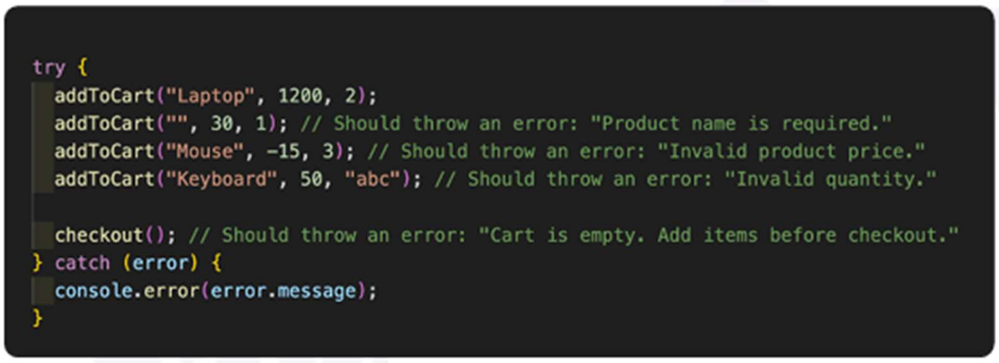
Task 1: Add Product to Cart Function

Create a function addToCart that simulates adding a product to the shopping cart. The function should take the product details (name, price, quantity) as parameters and throw errors under certain conditions:

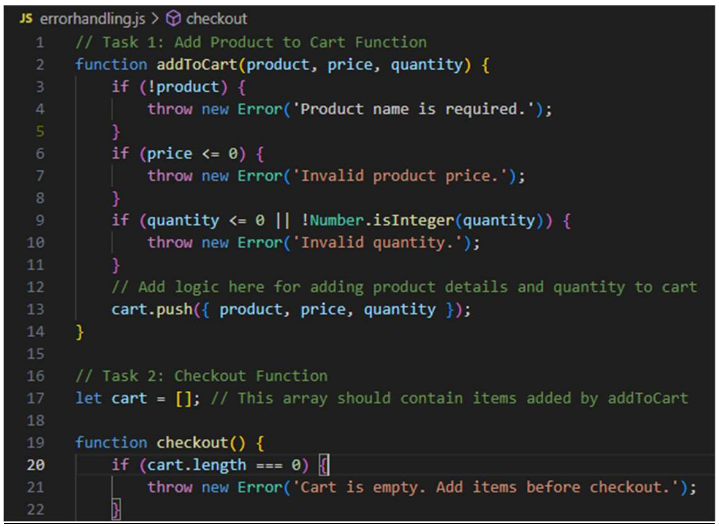
- If the product name is not provided, throw an error indicating "Product name is required."
- If the product price is not a positive number, throw an error indicating "Invalid product price."
- If the quantity is not a positive integer, throw an error indicating "Invalid quantity."

Task 2: Checkout Function

Create a function checkout that simulates the checkout process. This function should throw an error if the cart is empty, indicating "Cart is empty. Add items before checkout."

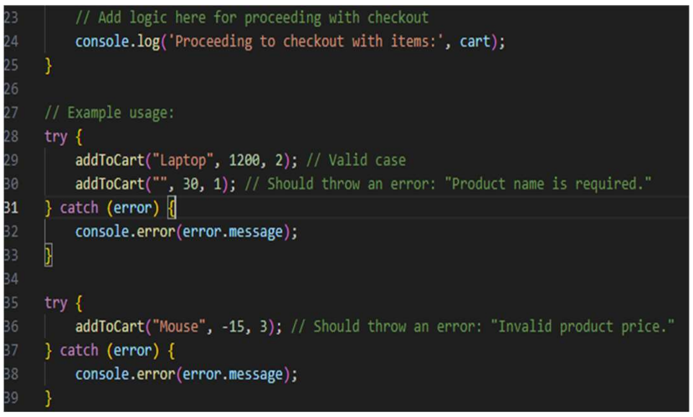


```javascript
try {
  addToCart("Laptop", 1200, 2);
  addToCart("", 30, 1); // Should throw an error: "Product name is required."
  addToCart("Mouse", -15, 3); // Should throw an error: "Invalid product price."
  addToCart("Keyboard", 50, "abc"); // Should throw an error: "Invalid quantity."

  checkout(); // Should throw an error: "Cart is empty. Add items before checkout."
} catch (error) {
  console.error(error.message);
}
```

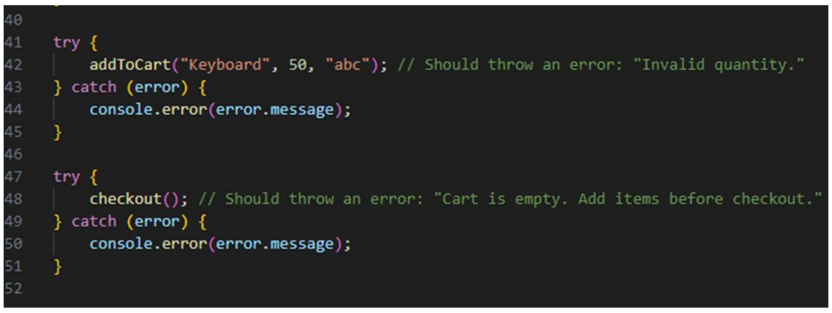


errorhandling.js > checkout

```javascript
// Task 1: Add Product to Cart Function
function addToCart(product, price, quantity) {
    if (!product) {
        throw new Error('Product name is required.');
    }
    if (price <= 0) {
        throw new Error('Invalid product price.');
    }
    if (quantity <= 0 || !Number.isInteger(quantity)) {
        throw new Error('Invalid quantity.');
    }
    // Add logic here for adding product details and quantity to cart
    cart.push({ product, price, quantity });
}

// Task 2: Checkout Function
let cart = []; // This array should contain items added by addToCart

function checkout() {
    if (cart.length === 0) {
        throw new Error('Cart is empty. Add items before checkout.');
    }
```

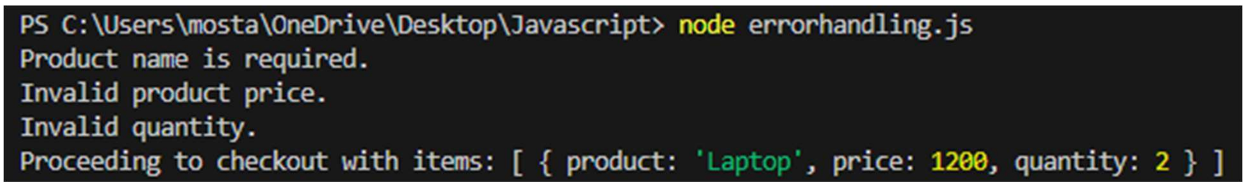


```javascript
    // Add logic here for proceeding with checkout
    console.log('Proceeding to checkout with items:', cart);
}

// Example usage:
try {
    addToCart("Laptop", 1200, 2); // Valid case
    addToCart("", 30, 1); // Should throw an error: "Product name is required."
} catch (error) {
    console.error(error.message);
}

try {
    addToCart("Mouse", -15, 3); // Should throw an error: "Invalid product price."
} catch (error) {
    console.error(error.message);
}
```



```javascript

try {
    addToCart("Keyboard", 50, "abc"); // Should throw an error: "Invalid quantity."
} catch (error) {
    console.error(error.message);
}

try {
    checkout(); // Should throw an error: "Cart is empty. Add items before checkout."
} catch (error) {
    console.error(error.message);
}

```

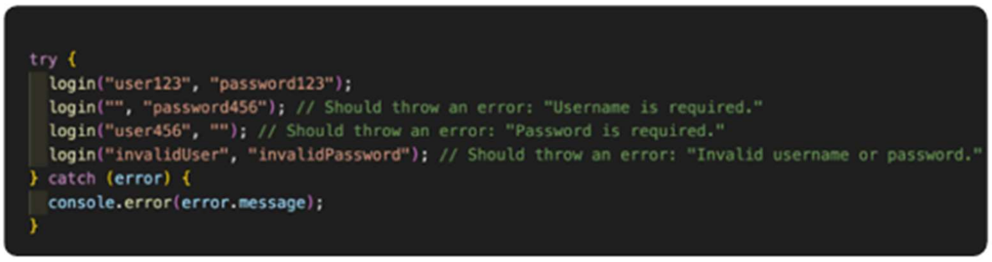
Output:



```
PS C:\Users\mosta\OneDrive\Desktop\Javascript> node errorhandling.js
Product name is required.
Invalid product price.
Invalid quantity.
Proceeding to checkout with items: [ { product: 'Laptop', price: 1200, quantity: 2 } ]
```

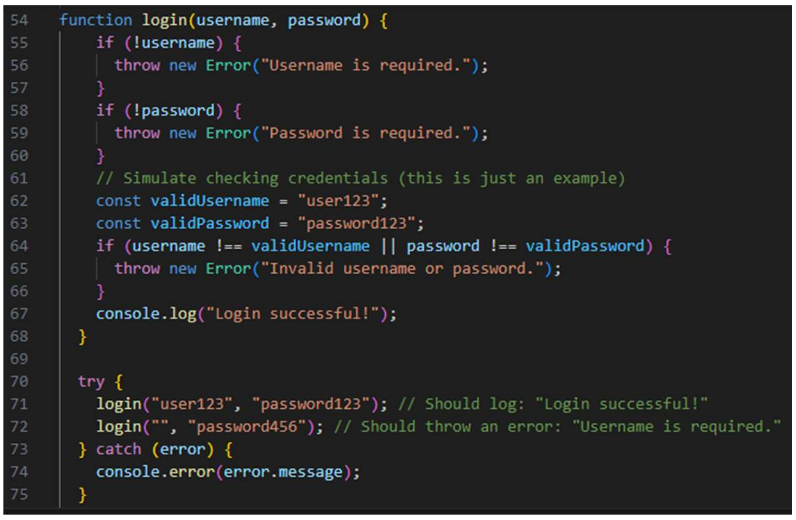
2. **You are working on a user authentication module for a web application. Implement error handling for the login process.Create a function login that simulates the user login process. The function should take the username and password as parameters and throw errors under certain conditions:**

- **If the username is not provided, throw an error indicating "Username is required."**
- **If the password is not provided, throw an error indicating "Password is required."**
- **If the username and password do not match any valid credentials, throw an error indicating "Invalid username or password."**

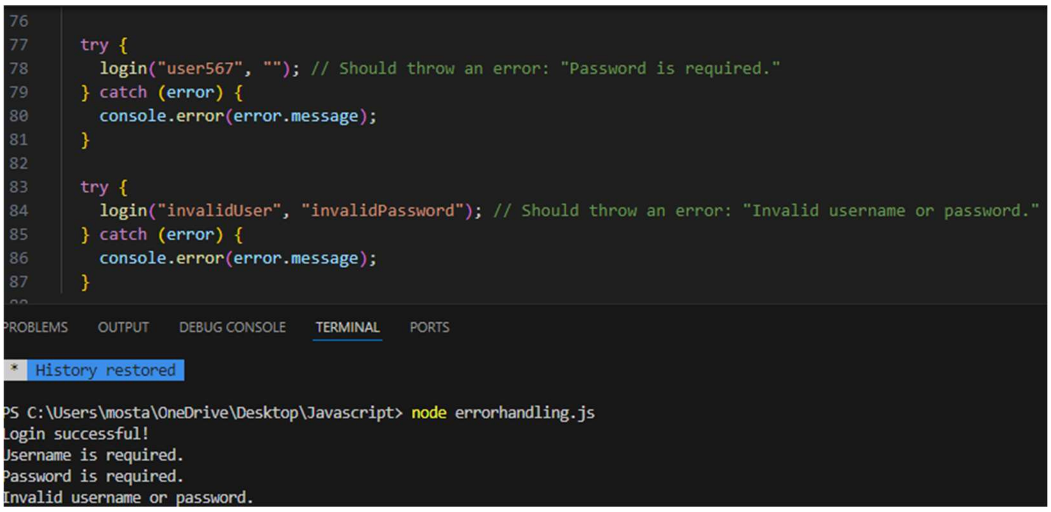


```javascript
try {
  login("user123", "password123");
  login("", "password456"); // Should throw an error: "Username is required."
  login("user456", ""); // Should throw an error: "Password is required."
  login("invalidUser", "invalidPassword"); // Should throw an error: "Invalid username or password."
} catch (error) {
  console.error(error.message);
}
```

Code:



```javascript
function login(username, password) {
    if (!username) {
      throw new Error("Username is required.");
    }
    if (!password) {
      throw new Error("Password is required.");
    }
    // Simulate checking credentials (this is just an example)
    const validUsername = "user123";
    const validPassword = "password123";
    if (username !== validUsername || password !== validPassword) {
      throw new Error("Invalid username or password.");
    }
    console.log("Login successful!");
  }

  try {
    login("user123", "password123"); // Should log: "Login successful!"
    login("", "password456"); // Should throw an error: "Username is required."
  } catch (error) {
    console.error(error.message);
  }
```



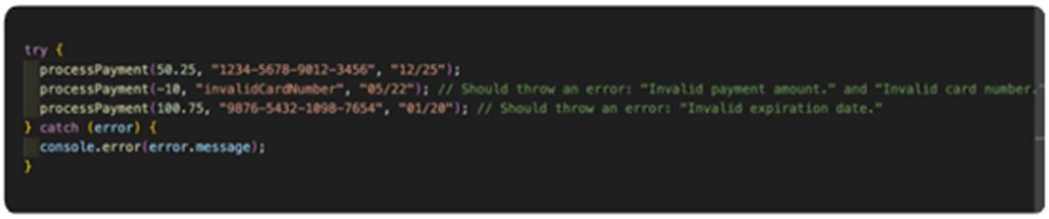
```javascript

  try {
    login("user567", ""); // Should throw an error: "Password is required."
  } catch (error) {
    console.error(error.message);
  }

  try {
    login("invalidUser", "invalidPassword"); // Should throw an error: "Invalid username or password."
  } catch (error) {
    console.error(error.message);
  }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
History restored

PS C:\Users\mosta\OneDrive\Desktop\Javascript> node errorhandling.js
Login successful!
Username is required.
Password is required.
Invalid username or password.
```

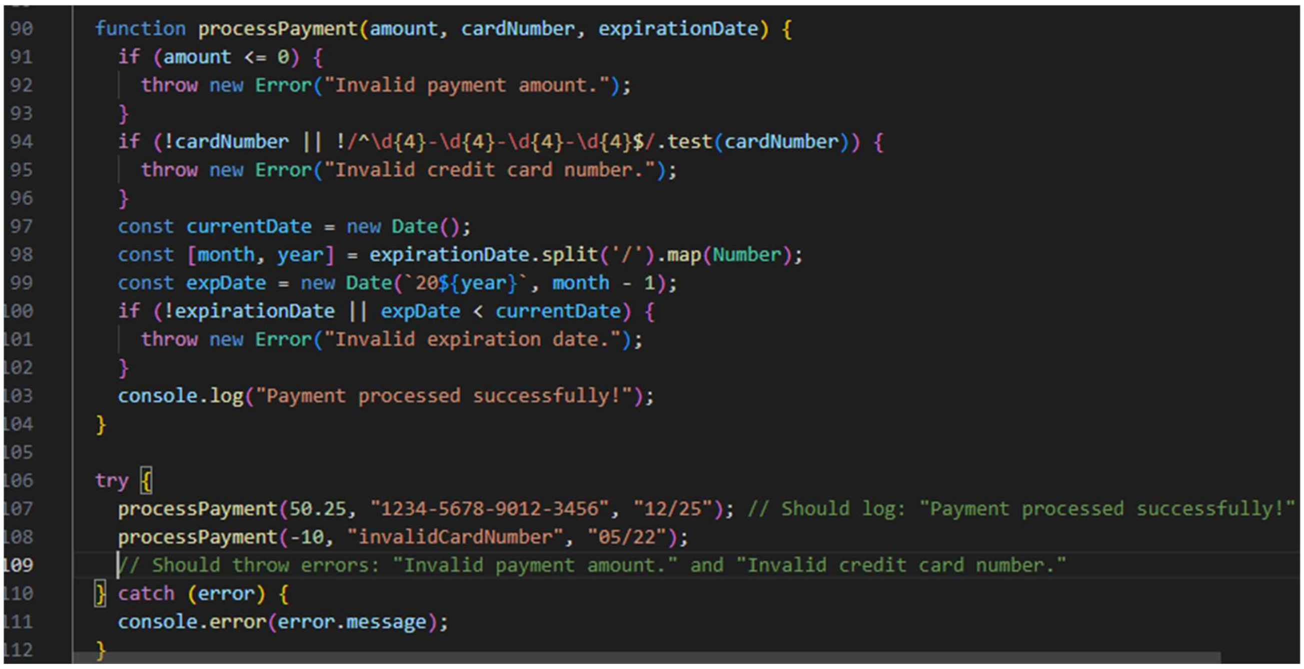
3. **You are developing a payment processing module for an e-commerce platform. Implement error handling for the payment transaction process.Create a function processPayment that simulates processing a payment transaction. The function should take payment details (amount, card number, expiration date) as parameters and throw errors under certain conditions:**

- **If the payment amount is not a positive number, throw an error indicating "Invalid payment amount."**
- **If the card number is not provided or is not a valid credit card number, throw an error indicating "Invalid card number."**
- **If the expiration date is not provided or is in the past, throw an error indicating "Invalid expiration date."**

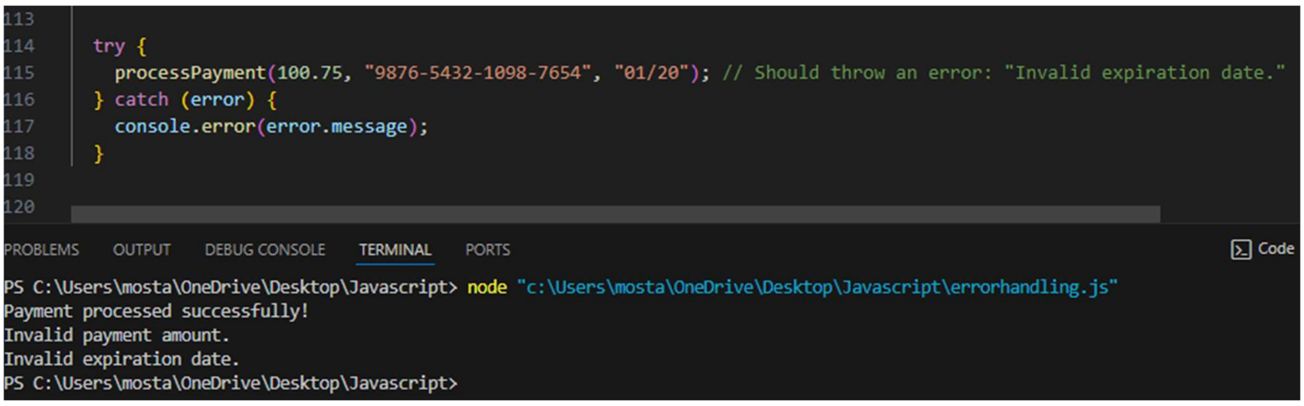


```
try {
  processPayment(50.25, "1234-5678-9012-3456", "12/25");
  processPayment(-10, "invalidCardNumber", "05/22"); // Should throw an error: "Invalid payment amount." and "Invalid card number."
  processPayment(100.75, "9876-5432-1098-7654", "01/20"); // Should throw an error: "Invalid expiration date."
} catch (error) {
  console.error(error.message);
}
```

Code:



```
function processPayment(amount, cardNumber, expirationDate) {
  if (amount <= 0) {
    throw new Error("Invalid payment amount.");
  }
  if (!cardNumber || !/^\d{4}-\d{4}-\d{4}-\d{4}$/.test(cardNumber)) {
    throw new Error("Invalid credit card number.");
  }
  const currentDate = new Date();
  const [month, year] = expirationDate.split('/').map(Number);
  const expDate = new Date(`20${year}`, month - 1);
  if (!expirationDate || expDate < currentDate) {
    throw new Error("Invalid expiration date.");
  }
  console.log("Payment processed successfully!");
}

try {
  processPayment(50.25, "1234-5678-9012-3456", "12/25"); // Should log: "Payment processed successfully!"
  processPayment(-10, "invalidCardNumber", "05/22");
  // Should throw errors: "Invalid payment amount." and "Invalid credit card number."
} catch (error) {
  console.error(error.message);
}
```



```

try {
  processPayment(100.75, "9876-5432-1098-7654", "01/20"); // Should throw an error: "Invalid expiration date."
} catch (error) {
  console.error(error.message);
}

```

```
PS C:\Users\mosta\OneDrive\Desktop\Javascript> node "c:\Users\mosta\OneDrive\Desktop\Javascript\errorhandling.js"
Payment processed successfully!
Invalid payment amount.
Invalid expiration date.
PS C:\Users\mosta\OneDrive\Desktop\Javascript>
```