Project Title:

E-COMMERCE APPLICATION

Phase 4: Development Part 2

Activity 1: User Authentication Implementation

Description: In this phase, we focused on implementing user registration and authentication features using a backend server. The goal is to ensure a secure and seamless experience for users of our e-commerce platform.

Steps Taken:

1. Set up a backend server using Node.js.
2. Utilized the Passport.js library for user authentication.
3. Created routes for user registration and login.
4. Implemented password hashing for enhanced security.

Challenges Faced:

- Challenge: Ensuring proper handling of user sessions and authentication.
- Challenge: Integrating with our existing database schema.

Solutions:

- Solution: Leveraged Passport.js for session management and authentication.
- Solution: Made necessary adjustments to our database schema for user-related data.

Code Snippets:

1. **Setting up Passport.js for Authentication:**

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

const LocalStrategy = require('passport-local').Strategy;


passport.use(new LocalStrategy(
```

Project Title:

E-COMMERCE APPLICATION

Phase 4: Development Part 2

```
 (username, password, done) => {
   // Check username and password against the database
   // Handle authentication success or failure
 }
));
```

User Registration Route:

```
app.post('/register', (req, res) => {
 // Handle user registration and validation
 // Store user information in the database
});
```

User Login Route:

```
app.post('/login', passport.authenticate('local'), (req, res) => {
 // Handle successful login
});
```

Activity 2: Shopping Cart Implementation

Description: In this activity, we implemented shopping cart functionality, allowing users to add products, manage their cart, and calculate the total cost of their selections.

Steps Taken:

1. Designed and developed the front-end shopping cart interface.
2. Created a data structure or database schema for shopping cart items.
3. Implemented routes to add, remove, and update items in the shopping cart.
4. Calculated the total cost of items in the cart.

Challenges Faced:

- Challenge: Managing real-time updates to the shopping cart.
- Challenge: Ensuring data consistency during concurrent updates.

Solutions:

Project Title:

E-COMMERCE APPLICATION

Phase 4: Development Part 2

- Solution: Utilized a front-end framework to handle real-time cart updates.
- Solution: Implemented server-side locking mechanisms to maintain data consistency.

Code Snippets:

1. **Adding Items to the Shopping Cart:**

```
app.post('/add-to-cart', (req, res) => {
  // Handle adding products to the user's shopping cart
  // Update the cart and calculate the new total cost
});
```

Calculating the Total Cost:

```
function calculateTotalCost(cartItems) {
  // Calculate the total cost based on items in the cart
  return cartItems.reduce((total, item) => total + item.price, 0);
}
```

Activity 3: Checkout Process Implementation

Description: During this phase, we focused on creating a smooth checkout process for our users. This includes a review of the cart, entering shipping details, selecting payment methods, and completing the transaction.

Steps Taken:

1. Developed a dedicated checkout page for users.
2. Integrated with payment gateways to facilitate secure transactions.
3. Ensured the secure handling of payment information.
4. Created confirmation and order summary pages for successful transactions.

Challenges Faced:

- Challenge: Integrating with external payment gateways.
- Challenge: Handling and securely storing payment information.

Project Title:

E-COMMERCE APPLICATION

Phase 4: Development Part 2

Solutions:

- Solution: Utilized well-documented APIs provided by payment gateways (e.g., Stripe).
- Solution: Implemented encryption and tokenization for secure payment data storage.

Code Snippets:

1. **Checkout Route:**

```
app.post('/checkout', (req, res) => {
  // Process payment using a payment gateway
  // Create an order and update the user's order history
  // Provide order confirmation and summary
});
```

Payment Gateway Integration:

```
// Sample code for integrating with Stripe for payment processing
const stripe = require('stripe')('your_secret_api_key');

stripe.charges.create({
  amount: totalCostInCents, // Amount in cents
  currency: 'usd',
  source: 'tok_visa', // Token representing the credit card
  description: 'Payment for Order #123',
}, (err, charge) => {
  if (err) {
    // Handle payment failure
  } else {
    // Payment successful, complete the order
  }
```

Project Title:

E-COMMERCE APPLICATION

Phase 4: Development Part 2

```
});
```

Conclusion

Phase 4 has seen significant progress with the successful implementation of user authentication, shopping cart functionality, and a streamlined checkout process. The project is well-positioned for the next phase, which will focus on additional enhancements and testing.