ARTISANAL E COMMERCE PLATFORM ON IBM CLOUD FOUNDRY

Introduction

This platform is meticulously crafted to provide a unique shopping experience, offering handcrafted products from artisans around the world. Leveraging the robust capabilities of IBM Cloud Foundry, we've built a seamless and secure environment for both buyers and sellers

Step 1: Setting Up Your Project

- Create a new Node.js project:

->Code: (node.js)

```
mkdir e-commerce-platform
cd e-commerce-platform
npm init -y
```

- Install necessary packages:

```
npm install express mongoose bcrypt jsonwebtoken body-parser
```

Step 2: Implement User Authentication

->Code :(java script)

->Set up a basic Express server:

```
const express = require('express');
const app = express();
const PORT = process.env.PORT || 3000;

app.use(express.json());

// Add routes and middleware for user authentication
// ...

app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

Step 2: Implement User Authentication

- Set up a basic Express server:

->Code:(javascript)

```
const express = require('express');
const app = express();
const PORT = process.env.PORT || 3000;

app.use(express.json());

// Add routes and middleware for user authentication
// ...

app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

- Create a models/User.js file to define the user schema and model using Mongoose.

➢ Implement routes for user registration and authentication:

- /api/register (POST): Handles user registration.
- /api/login (POST): Handles user login and issues a JWT token upon successful login.

Step 3: Implement Shopping Cart

➢ Create a models/Product.js file to define the product schema and model using Mongoose.

➢ Create routes for managing the shopping cart:

- /api/cart/add (POST): Adds a product to the user's shopping cart.
- /api/cart/remove (POST): Removes a product from the cart.
- /api/cart (GET): Retrieves the user's current shopping cart.

Step 4: Implement Checkout Functionality

➢ Create a route for handling the checkout process:

- /api/checkout (POST): Processes the user's order, deducts the items from the inventory, and generates an order confirmation.

➢ Update the product model to include an inventory count.

Step 5: Frontend Integration

➢ Integrate your backend with a frontend framework (e.g., React, Vue, Angular) to provide a user interface for your e-commerce platform. Use APIs you've created to handle user authentication, shopping cart operations, and checkout.

Step 6: User Interface

➢ Design and implement the user interface for registration, login, product listings, shopping cart, and checkout pages.

Step 7: Testing and Deployment

➢ Write unit tests to ensure the functionality works as expected.

- Deploy your application using platforms like Heroku, AWS, or any other hosting service of your choice.

Conclusion :

Remember to handle edge cases, validation, error handling, and security measures (e.g., input validation, password hashing, secure JWT implementation) to ensure your platform is robust and secure.

Please note that this is a high-level overview, and each step involves detailed coding, database setup, and testing. It's recommended to break down each step further and refer to specific documentation as you go along.