KGiSL Institute Of Technology

NAAN MUDHALVAN

Problem Definition:

The project is to build an artisanal e-commerce platform using IBM Cloud Foundry. The goal is to connect skilled artisans with a global audience, showcasing their handmade products and providing features like secure shopping carts, payment gateways, and an intuitive checkout process. This involves designing the e-commerce platform, implementing necessary features, and ensuring a seamless user experience.

Project Name:

E-commerce application on IBM Cloud Foundry

Team Members:

1. Manthra.P
2. Kaviya.B
3. Samraj.M
4. Soundarapandi.M

Problem Statement: Build an artisanal e-commerce platform using IBM Cloud Foundry. Connect skilled artisans with a global audience. Showcase handmade products, from exquisite jewelry to artistic home decor. Implement secure shopping carts, smooth payment gateways, and an intuitive checkout process. Nurture creativity and support small businesses through an artisan's dream marketplace.

1. Define Your Artisan E-commerce Platform's Goals and Features:

Clearly define the objectives of your platform.Identify key features such as product listings, secure shopping carts, payment gateways, user profiles, reviews, and more.Understand your target audience and their needs.

2. Set Up IBM Cloud Foundry:

Create an IBM Cloud account if you don't already have one.Install and configure the IBM Cloud CLI.Create an IBM Cloud Foundry application where your e-commerce platform will be hosted.

3. Choose and Set Up a Database:

Decide on a database solution like IBM Db2, PostgreSQL, or a NoSQL database like Cloudant.Configure and set up the chosen database to store product information, user data, and transaction records.

4. Develop the Frontend:

Choose a technology stack for your frontend (e.g., React, Angular, or Vue.js).Create a user-friendly and visually appealing interface for your platform.Implement responsive design for mobile devices.Integrate the frontend with your IBM Cloud Foundry application.

5. Build the Backend:

Choose a backend technology stack (e.g., Node.js, Ruby on Rails, Python with Flask/Django).Develop APIs for user registration, product listing, shopping carts, and payment processing.Implement user authentication and authorization.Connect the backend to the database.

DEVELOPMENT:

Creating a full-fledged e-commerce platform involves a significant amount of code and configuration, which is beyond the scope of a single response. However, I can provide you with a basic outline and code snippets to get you started on building your artisanal e-commerce platform using IBM Cloud Foundry.

Steps:

1. Set Up Your Project Structure:

```
artisanal-ecommerce/
|-- public/
| |-- index.html
| |-- style.css
|-- server/
| |-- server.js
|-- manifest. yml
|-- . cfignore
```

2. Implement Backend (server/server.js):

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
const port = process.env.PORT || 3000;
app.use(bodyParser.json());
// Implement your routes and logic for handling products,
payments, etc.
app.listen(port, () => {
```

3.Implement Frontend (public/index.html):

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<link rel="stylesheet" href="style.css">
<title>Artisanal E-Commerce Platform</title>
</head>
<body>
<!-- Your HTML content for displaying products, shopping cart,
and checkout form -->
</body>
</html>
```

4. Configure Manifest File (manifest.yml):

```
applications:
- name: artisanal-ecommerce
memory: 256M
instances: 1
buildpacks:
- nodejs_buildpack
```

A node.js backend Structure:

```
 // server.js
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const passport = require('passport');
const app = express();
const port = process.env.PORT || 3000;
// Middleware
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
// Database connection (using MongoDB as an example)
mongoose.connect('mongodb://localhost/artisanal_ecommerce', {
useNewUrlParser: true });
mongoose.connection.on('error', console.error.bind(console,
'MongoDB connection error:'));
mongoose.set('useFindAndModify', false);
// Passport middleware for user authentication
app.use(passport.initialize());
require('./config/passport')(passport);
// Define routes for user authentication, product management, and
orders
app.use('/api/auth', require('./routes/auth'));
app.use('/api/products', require('./routes/products'));
app.use('/api/orders', require('./routes/orders'));
// Start the server
app.listen(port, () => {
console.log(`Server is running on port ${port}`);
});
```

The authentication route (`routes/auth.js`):

```
const express = require('express');
const router = express.Router();
const passport = require('passport');
// Load User model
const User = require('../models/User');
// Route to register a new user
router.post('/register', (req, res) => {
// Implement user registration logic here
});
// Route to log in
router.post('/login', (req, res) => {
// Implement user login logic here
});
// Route to log out
router.get('/logout', (req, res) => {
// Implement user logout logic here
});
// Protected route that requires authentication
router.get('/profile', passport.authenticate('jwt', { session: false }),
(req, res) => {
res.json({ user: req.user });
});
module.exports = router;
```

A product management route (`routes/products. js`):

```
const express = require('express');
const router = express.Router();
// Load Product model
const Product = require('../models/Product');
// Route to create a new product
router.post('/', (req, res) => {
// Implement product creation logic here
});
// Route to get a list of products
router.get('/', (req, res) => {
// Implement product retrieval logic here
});
// Route to edit a product
router.put('/:id', (req, res) => {
// Implement product update logic here
});
// Route to delete a product
router.delete('/:id', (req, res) => {
// Implement product deletion logic here
});
module.exports = router;
```