KGiSL Institute Of Technology

NAAN MUDHALVAN

Cloud Application Development

Project name: **E-Commerce Application on IBM cloud Foundry**

Team Members:

P.Manthra

B.Kaviya

M.Samraj

M.Soundarapandi

Problem Definition:

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.

To develop a complex artisanal e-commerce platform, you need to:

1. Implement product management, order processing, and payment integration.

2. Add security features, such as input validation and protection against common web vulnerabilities.
3. Develop the frontend using a modern framework (React, Angular, etc.).
4. Integrate a payment gateway (e.g., Stripe) for handling transactions securely.
5. Implement advanced features such as user reviews, internationalization, and more.

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;
```