

Building an Artisanal E-Commerce Platform with IBM Cloud Foundry

Table of Contents

Introduction

Problem Statement

Project Contents

- 3.1. Login Page or Home Page
- 3.2. Product Showcase and Descriptions
- 3.3. Authentication and Registration
- 3.4. Modeling Shopping Cart and Checkout Process
- 3.5. Secure Payment
- 3.6. Advanced User Experience
- 3.7. IBM Cloud Foundry
- 3.8. Loading Data into Storage
- 3.9. Products Inserted into DB2

Future Enhancements

- 4.1. Internationalisation
- 4.2. Artisan Verification
- 4.3. Live Chat
- 4.4. Mobile App
- 4.5. Sustainability Initiatives
- 4.6. AI and Machine Learning
- 4.7. Inventory Management
- 4.8. Analytics and Reporting
- 4.9. Subscription Models
- 4.10. Social Integration

Conclusion

1. Introduction

The "Building an Artisanal E-Commerce Platform with IBM Cloud Foundry" project aims to create an e-commerce platform that connects skilled artisans with a global audience. This platform will provide a secure and intuitive shopping experience, showcasing handmade products, and facilitating secure payments.

2. Problem Statement

The problem statement for this project is to build an artisanal e-commerce platform with the following key features:

Platform Design: Create an appealing and user-friendly platform design.

Product Showcase and Descriptions: Allow artisans to list their products with detailed descriptions.

Authentication and Registration: Implement user authentication and registration.

Shopping Cart and Checkout Process: Model a seamless shopping cart and checkout process.

Secure Payment: Integrate secure payment gateways.

Advanced User Experience: Enhance user experience with personalized features.

3. Project Contents

3.1. Login Page or Home Page

Create a welcoming login or home page with an intuitive design.

Allow users to access their accounts or register as new users.

3.2. Product Showcase and Descriptions

Enable artisans to showcase their handmade products with high-quality images and descriptions.

Implement filtering and sorting for product discovery.

3.3. Authentication and Registration

Develop a user authentication system with secure login and registration features.

Consider social media integration for convenience.

3.4. Modeling Shopping Cart and Checkout Process

Allow users to add items to their shopping cart.

Streamline the checkout process with minimal steps for a convenient experience.

3.5. Secure Payment

Integrate multiple payment gateways to ensure secure transactions.
Implement data encryption and secure payment processing.

3.6. Advanced User Experience

Offer personalized product recommendations based on user behavior.
Provide responsive customer support through chat, email, or phone.
Collect user feedback to continuously improve the platform.

3.7. IBM Cloud Foundry

Host the e-commerce platform on IBM Cloud Foundry for scalability and reliability.

3.8. Loading Data into Storage

Implement a data storage solution to efficiently manage product information and user data.

3.9. Products Inserted into DB2

Utilize IBM Db2 for managing the database, ensuring data reliability and security.

4. Future Enhancements

4.1. Internationalization

Support multiple languages and currencies to attract a diverse global audience.

4.2. Artisan Verification

Develop a system to verify artisans' skills and product authenticity.

4.3. Live Chat

Add real-time chat support for immediate customer assistance.

4.4. Mobile App

Create a dedicated mobile app for a seamless mobile shopping experience.

4.5. Sustainability Initiatives

Highlight eco-friendly and sustainable products and practices on the platform.

4.6. AI and Machine Learning

Utilize AI for better product recommendations and personalized experiences.

4.7. Inventory Management

Assist artisans in managing their inventory effectively.

4.8. Analytics and Reporting

Provide detailed insights and reports to artisans to optimize their sales strategies.

4.9. Subscription Models

Introduce subscription-based services for artisans and customers, offering exclusive benefits.

4.10. Social Integration

Enable users to share products on social media and implement social commerce features.

5. Conclusion

This detailed documentation outlines the plan for building an artisanal e-commerce platform using IBM Cloud Foundry. The project focuses on addressing the initial problem statement and suggests potential future enhancements to enhance the platform's functionality and user experience. Regularly gather feedback and iterate on the platform to ensure it continues to meet the needs of skilled artisans and a global audience.



BUILDING AN ARTISANAL E-COMMERCE PLATFORM WITH IBM CLOUD FOUNDRY

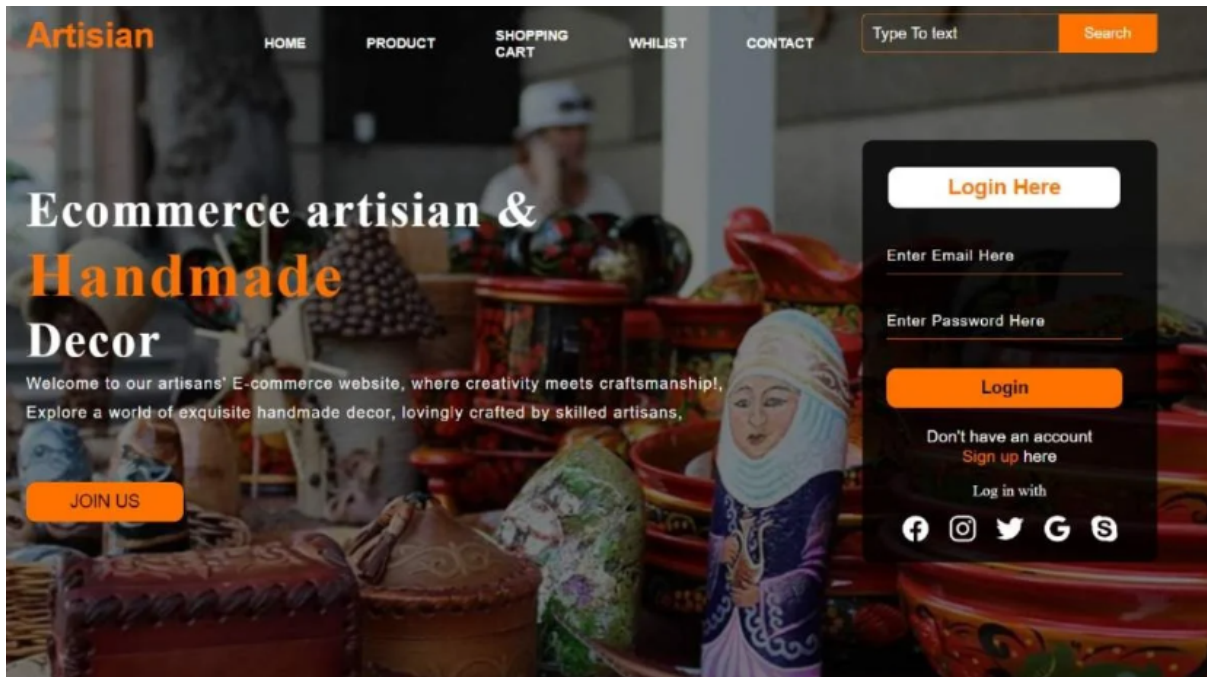
BY TEAM CLOUD CHASERS

1. ***Yuvashri S***
(211521244062)
2. ***Swathikasri S***
(211521244052)
3. ***Swetha V***
(211521244054)
4. ***Swetha G S***
(211521244053)
5. ***Udhaya E***
(211521244056)

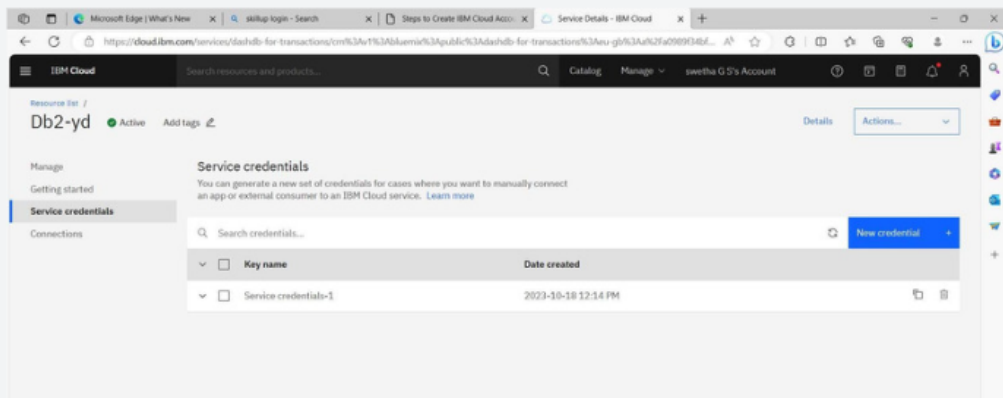
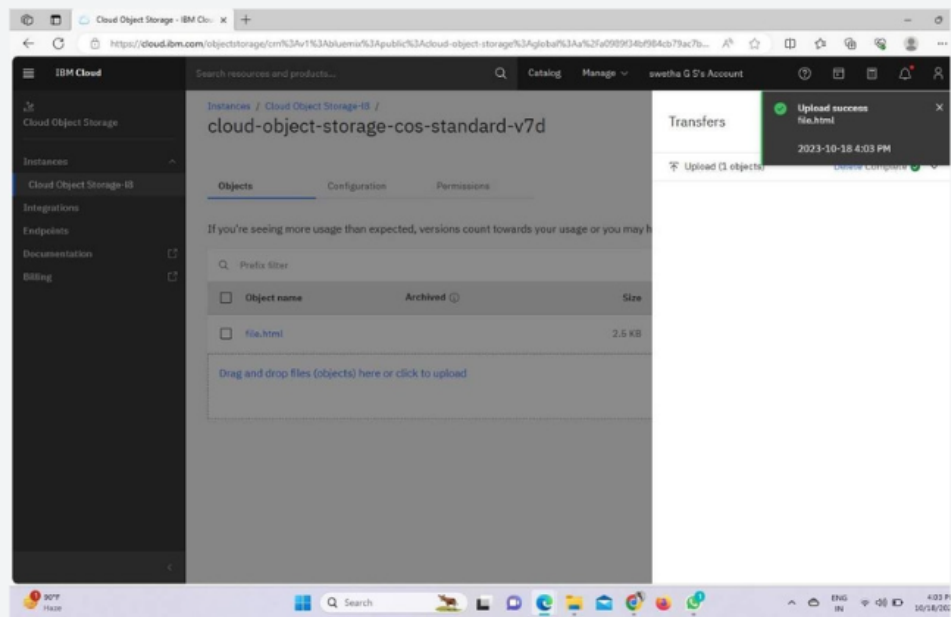
PROBLEM STATEMENT:

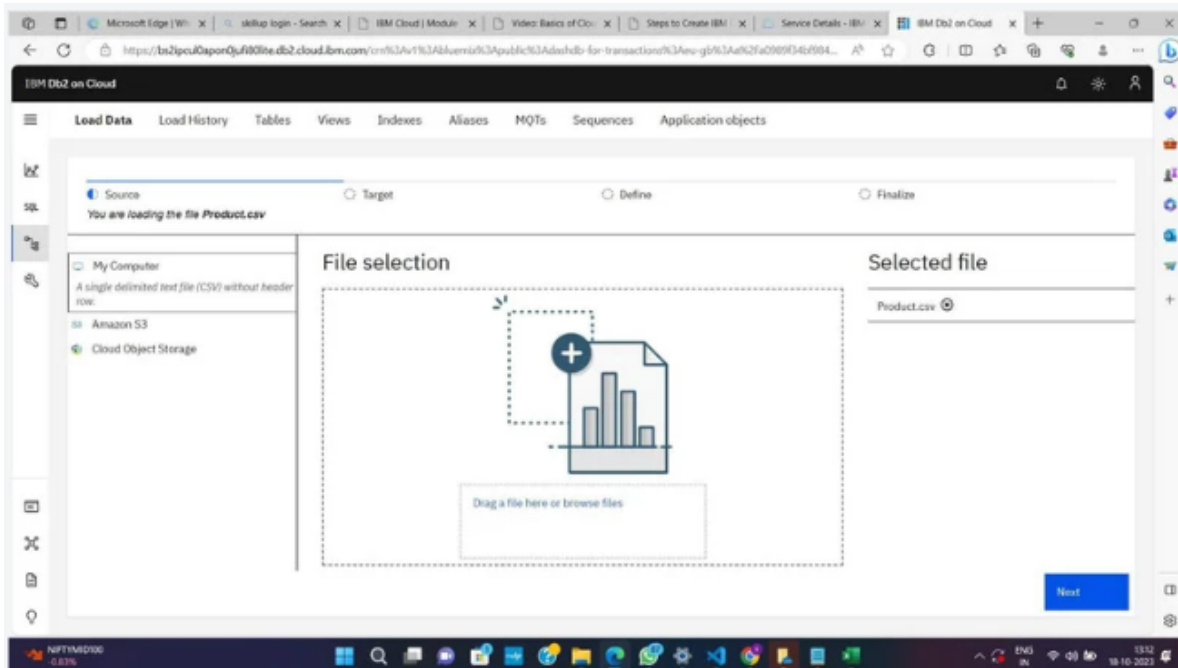
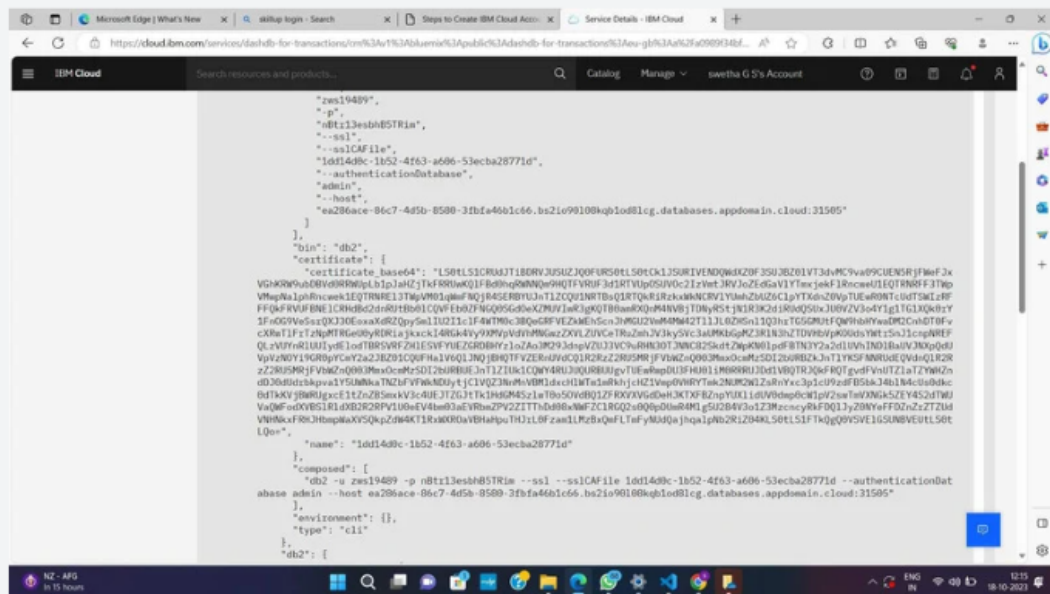
- Building an artisanal e-commerce platform to connect skilled artisans with a global audience
- This application showcases their handmade products and provides features like secure shopping carts, payment gateways, and an intuitive checkout process
- Adding features which enhances Platform Design, Product Showcase, User Authentication, Shopping Cart and Checkout, Payment Integration, and User Experience

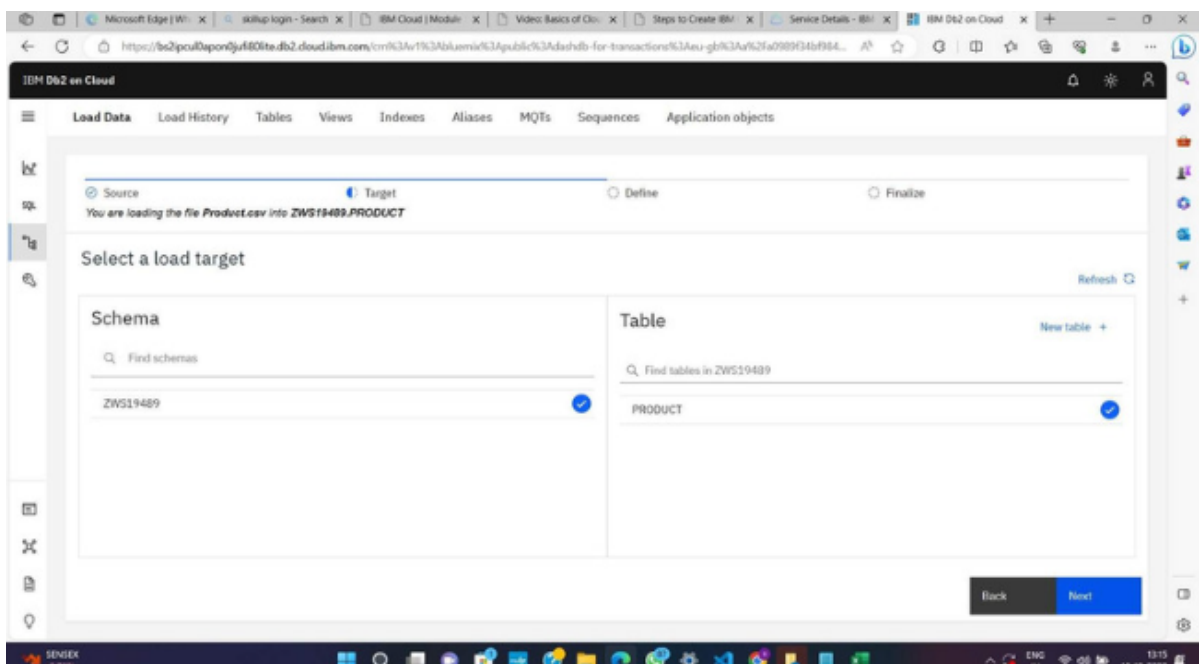
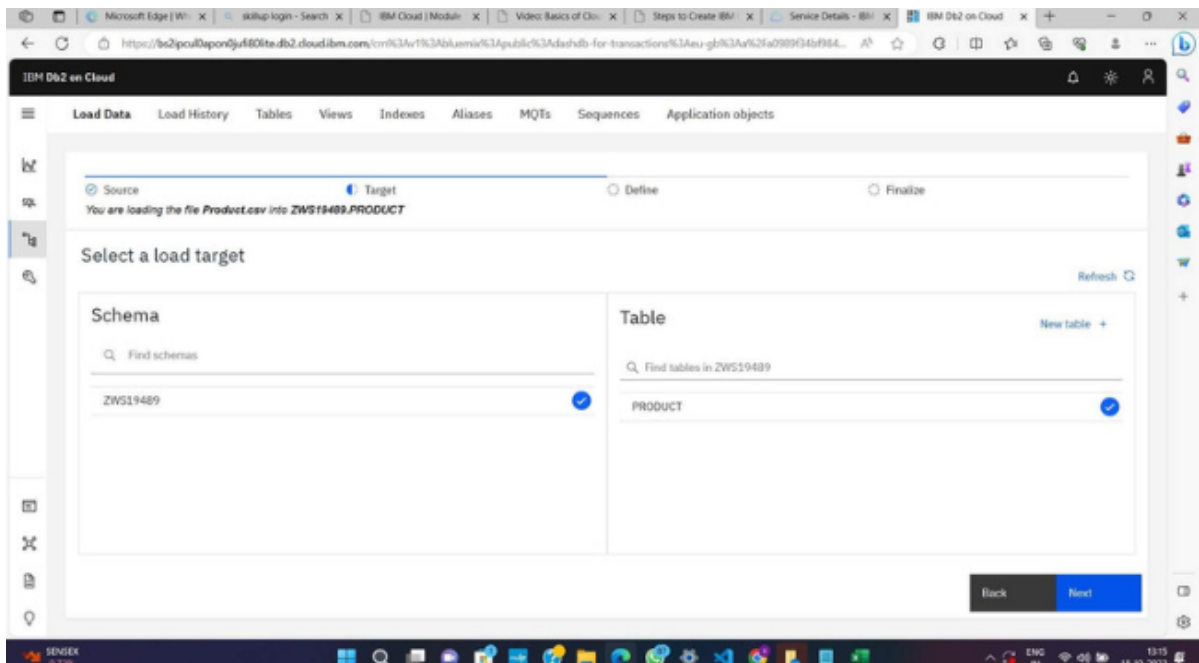
- 01 Creating a login page or home page
- 02 Product showcase and discriptions
- 03 Authentication and registration
- 04 Modeling Shopping Cart and Checkout process
- 05 Secure Payment
- 06 Advanced user Experience
- 07 IBM cloud foundry
- 08 loading data into storage
- 09 products inserted into db2



IMPLEMENTED IN IBM CLOUD USING DB2 AND CLOUD STORAGE







IBM Db2 on Cloud

Load Data Load History Tables Views Indexes Aliases MQTs Sequences Application objects

Load details

My computer Target
Product.csv ZWS19489.PRODUCT

10 Rows read 10 Rows loaded 0 Rows rejected

Start time: 10/18/2023 1:18:58 PM
End time: 10/18/2023 1:19:02 PM

The data load job succeeded.
You can now work with your data.

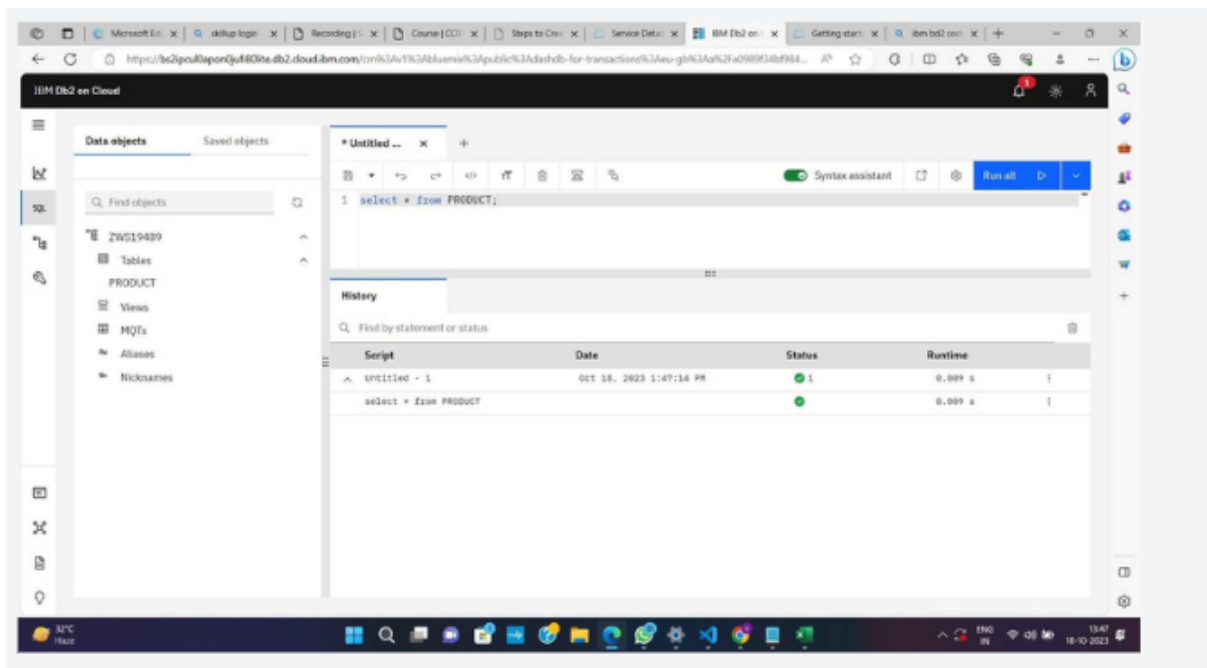
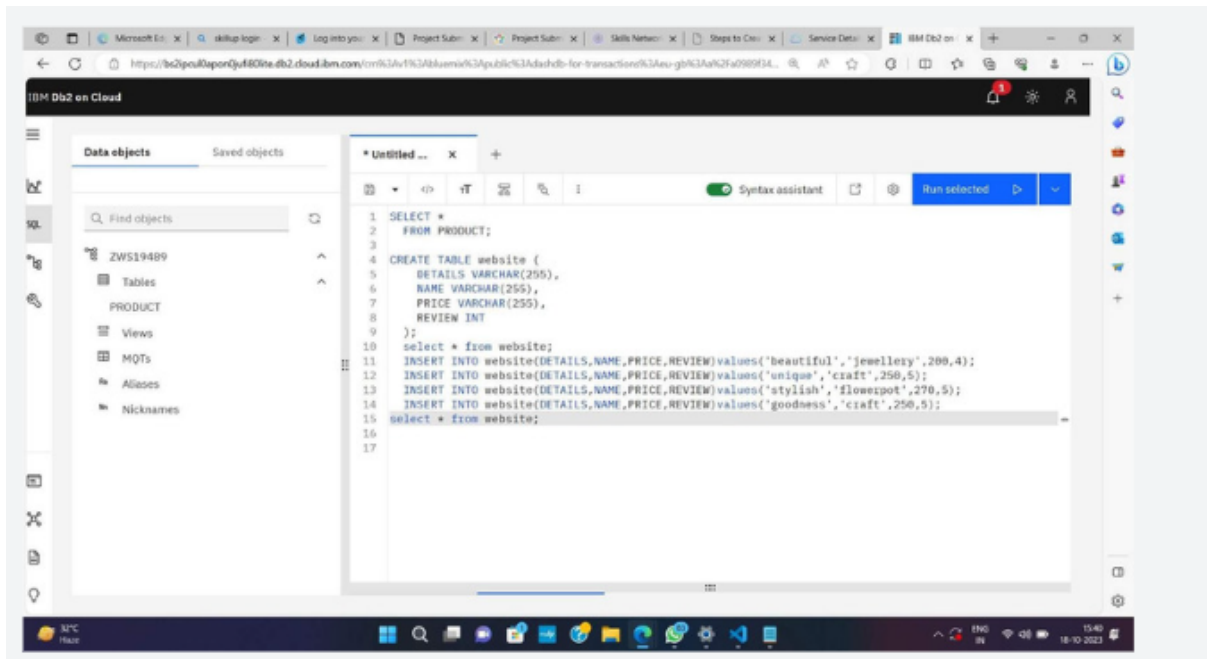
No errors

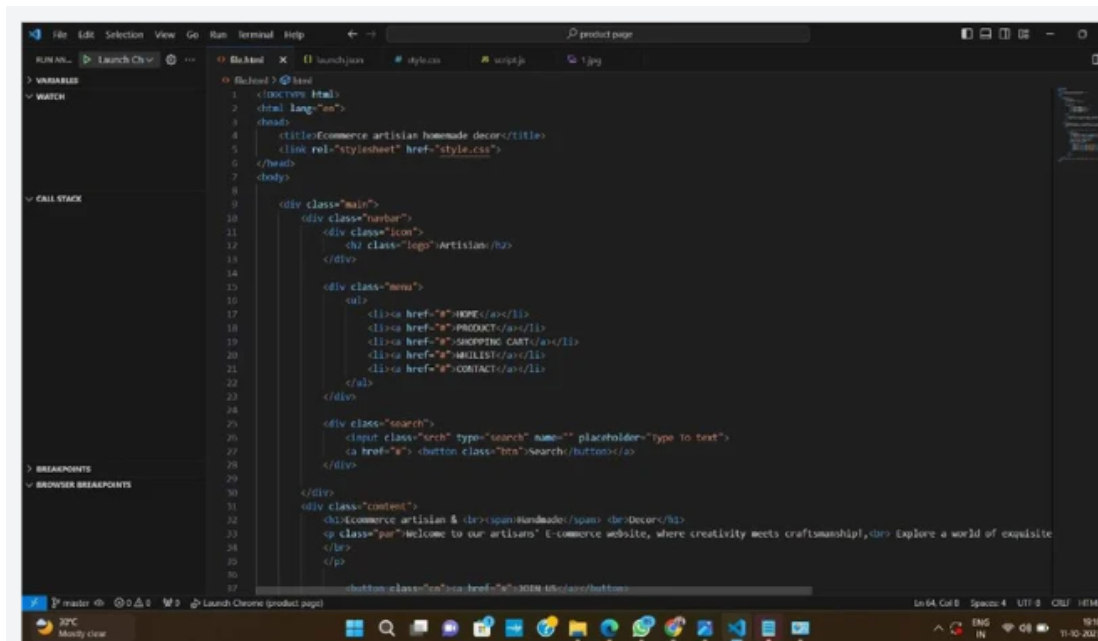
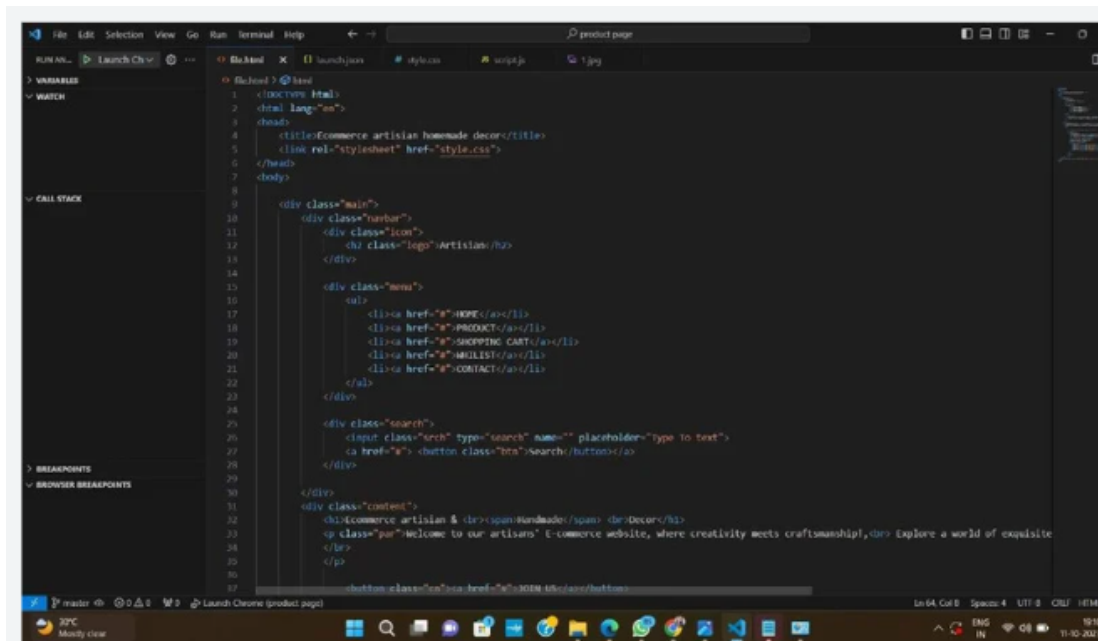
2023/10/18, 01:19 PM

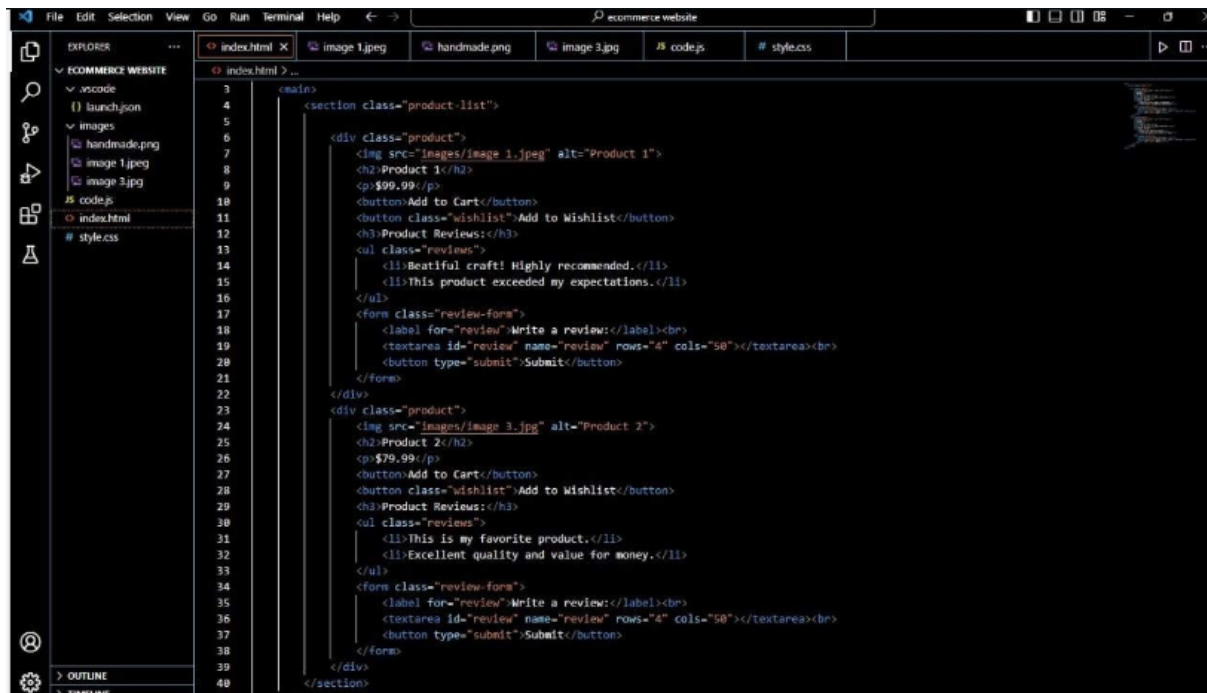
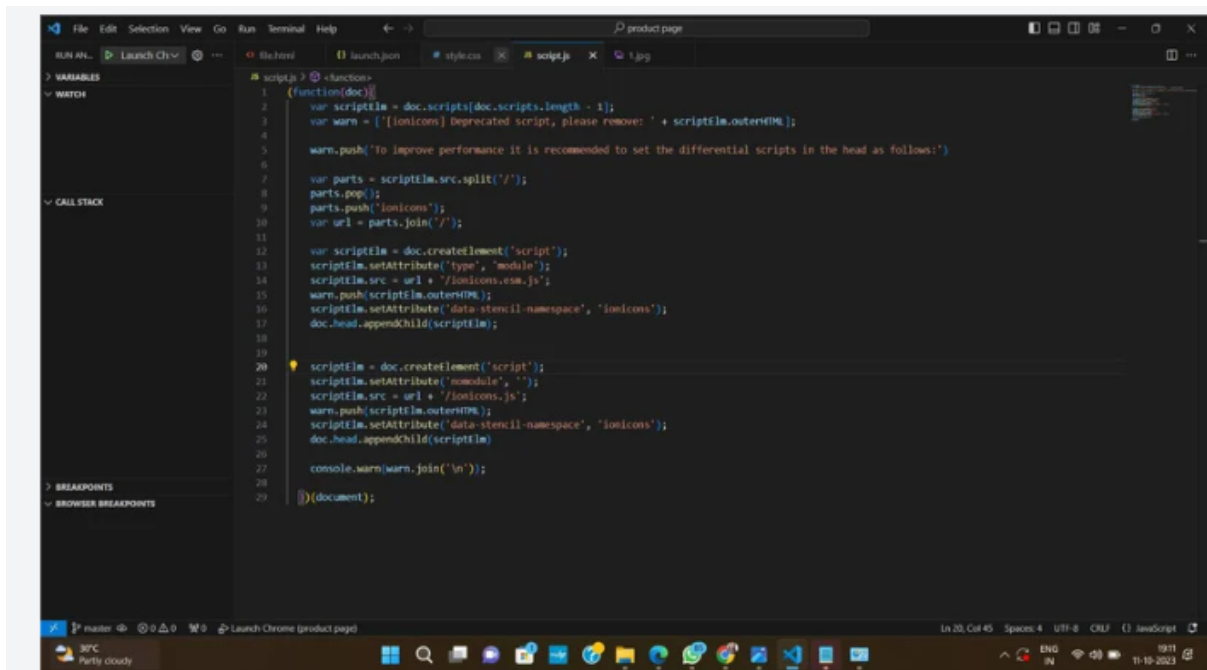
IBM Db2 on Cloud

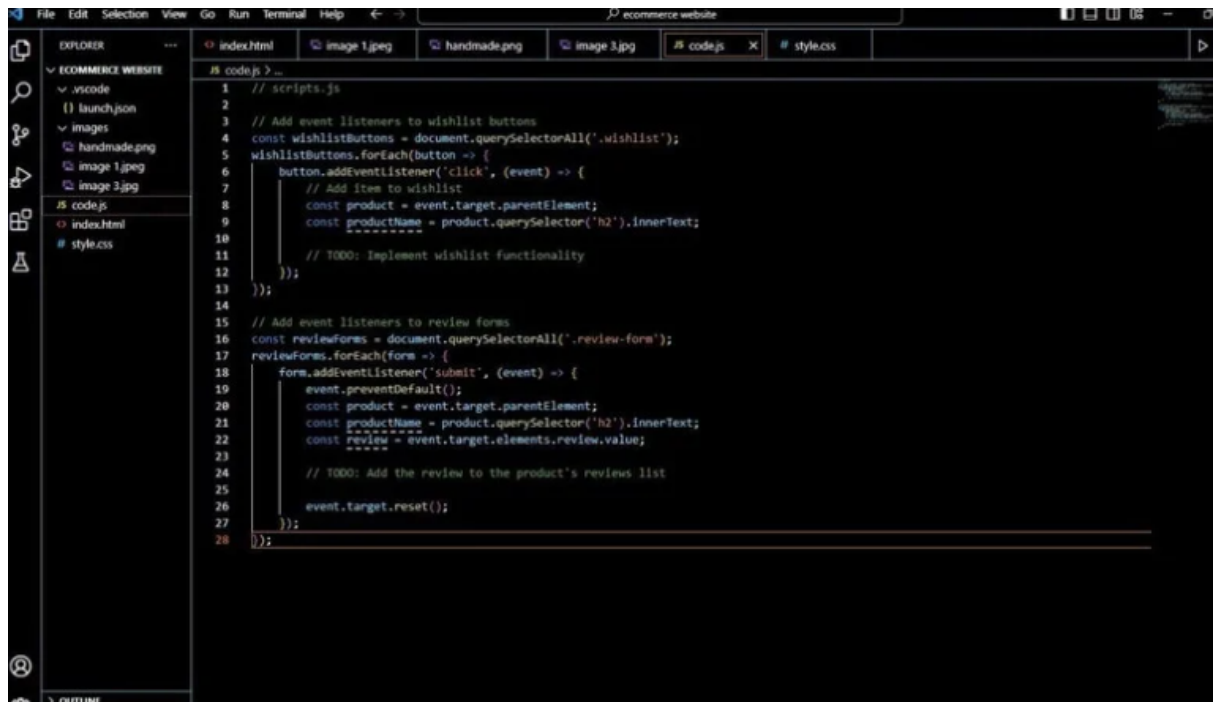
Load Data Load History Tables Views Indexes Aliases MQTs Sequences Application objects

| STATUS | SOURCE | FILENAME | TARGET | REQUESTED BY | ROWS LOADED | ROWS REJECTED |
|---------|-------------|-------------|------------------|--------------|-------------|---------------|
| Success | My computer | Product.csv | ZWS19489.PRODUCT | zws19489 | 10 | 0 |



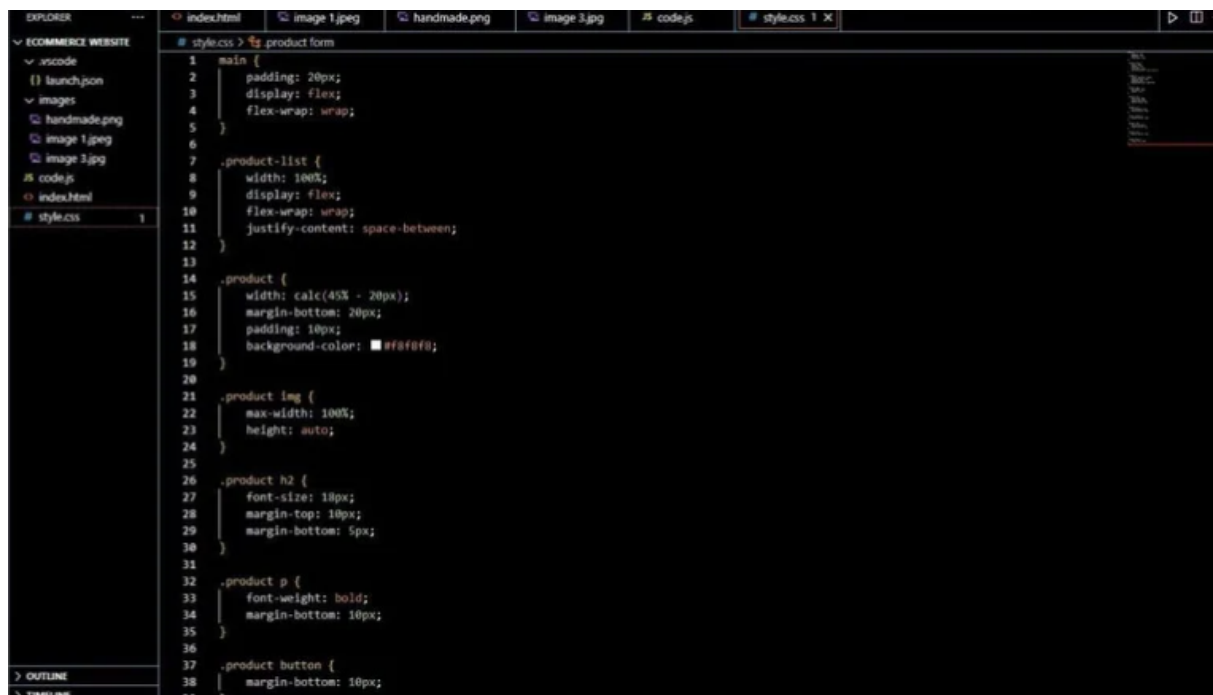






The screenshot shows the VS Code editor interface with a project named "ecommerce website". The Explorer sidebar on the left shows the file structure: `ecommerce website` (root), `.vscode`, `launch.json`, `images` (containing `handmade.png`, `image 1.jpeg`, and `image 3.jpg`), `code.js`, `index.html`, and `style.css`. The main editor area displays the `scripts.js` file with the following JavaScript code:

```
1 // scripts.js
2
3 // Add event listeners to wishlist buttons
4 const wishlistButtons = document.querySelectorAll('.wishlist');
5 wishlistButtons.forEach(button => {
6   button.addEventListener('click', (event) => {
7     // Add item to wishlist
8     const product = event.target.parentElement;
9     const productName = product.querySelector('h2').innerText;
10
11     // TODO: Implement wishlist functionality
12   });
13 });
14
15 // Add event listeners to review forms
16 const reviewForms = document.querySelectorAll('.review-form');
17 reviewForms.forEach(form => {
18   form.addEventListener('submit', (event) => {
19     event.preventDefault();
20     const product = event.target.parentElement;
21     const productName = product.querySelector('h2').innerText;
22     const review = event.target.elements.review.value;
23
24     // TODO: Add the review to the product's reviews list
25
26     event.target.reset();
27   });
28 });
```



The screenshot shows the VS Code editor interface with the same project. The Explorer sidebar is identical. The main editor area displays the `style.css` file with the following CSS code:

```
1 main {
2   padding: 20px;
3   display: flex;
4   flex-wrap: wrap;
5 }
6
7 .product-list {
8   width: 100%;
9   display: flex;
10  flex-wrap: wrap;
11  justify-content: space-between;
12 }
13
14 .product {
15   width: calc(45% - 20px);
16   margin-bottom: 20px;
17   padding: 10px;
18   background-color: #f8f8f8;
19 }
20
21 .product img {
22   max-width: 100%;
23   height: auto;
24 }
25
26 .product h2 {
27   font-size: 18px;
28   margin-top: 10px;
29   margin-bottom: 5px;
30 }
31
32 .product p {
33   font-weight: bold;
34   margin-bottom: 10px;
35 }
36
37 .product button {
38   margin-bottom: 10px;
```