**CASE STUDY DOCUMENTATION**

EshoppingZone

Group Members

* YASIR AHMMED(GROUP LEAD)
* ABHINAV SASEENDRAN
* ANIKET KUMAR
* RAJ KISHOR SUNNY

Introduction

EShoppingZone is an online platform that enables buying and selling of products, which is commonly referred to as ecommerce. The website has two types of users: merchants, who offer products for sale, and customers, who purchase products. Customers can register an account on the website, which enables them to log in and browse various product categories. The website offers a wide range of products including electronics, books, personal-care items, and more. Customers can select products and add them to their cart. Once the customer has added all the desired products to their cart, they can proceed to the checkout page to see the total amount due. The website offers various payment options such as UPI, Cards and cash on delivery. Users can also update their profiles and view their previous orders and transactions

**Document Purpose**

This document describes the solution architecture for E-Shopping Zone microservice.

**Intented Audience**

This document is intended as a reference for the following roles and stakeholders who are interested in the Customer Management Microservice technical architecture

|  |  |
| --- | --- |
| ROLE | NATURE OF ENGAGEMENT |
| Customers | Primary users of the website who browse, add items to their cart and complete purchases. |
| Marketing And Sales Team | Manage product catalogs, promotions, discounts, and marketing campaigns to analyze sales data and improve the customer experience and revenue. |
| Operations Team | Responsible for inventory management, order processing, shipping, and communication with customers about their orders |
| Technical team | Responsible for designing, developing, testing, and maintaining the website, understanding the system's architecture and design. |

**Project Background And Objectives**

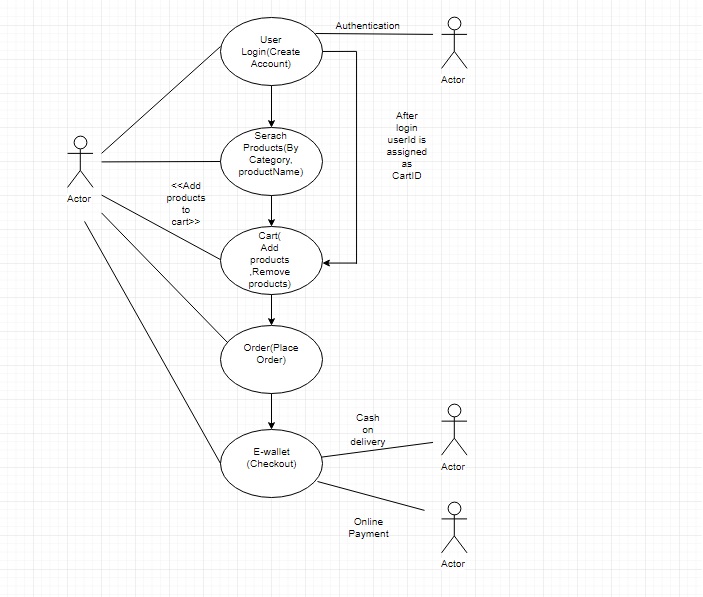
**Project Background:**

The project aims to develop an online shopping website to allow customers to purchase products easily from the comfort of their own homes. The website will provide a user-friendly interface to browse products, add items to the cart, and complete purchases.

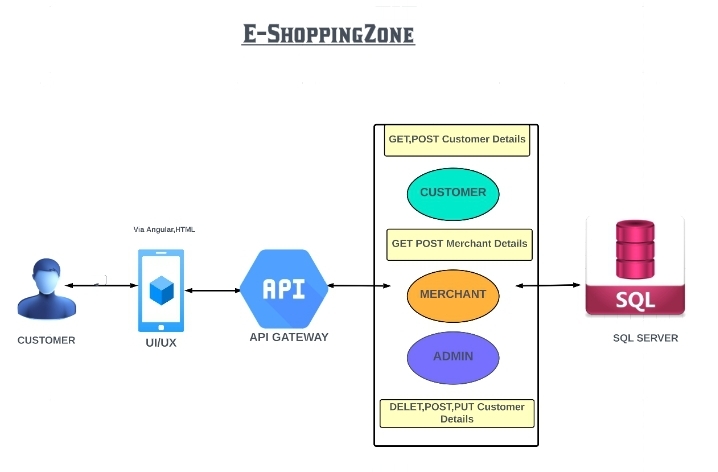
**Project Objective:**

* Develop a user-friendly interface that allows customers to browse products, add items to their cart, and complete purchases easily.
* Provide a personalized experience for customers through features such as wishlists, reviews and ratings
* Ensure website scalability, reliability, and security through proper error handling, testing, and deployment strategies.

**Use Case Diagram**



**Solution Diagram**



**Flow Diagram**

USER

ADMIN

Login

Add Category

Add Product

Manage Payment

Manage Order

Check Feedback

Login

SignUp

Log Out

Search Product

View Product

Buy Product

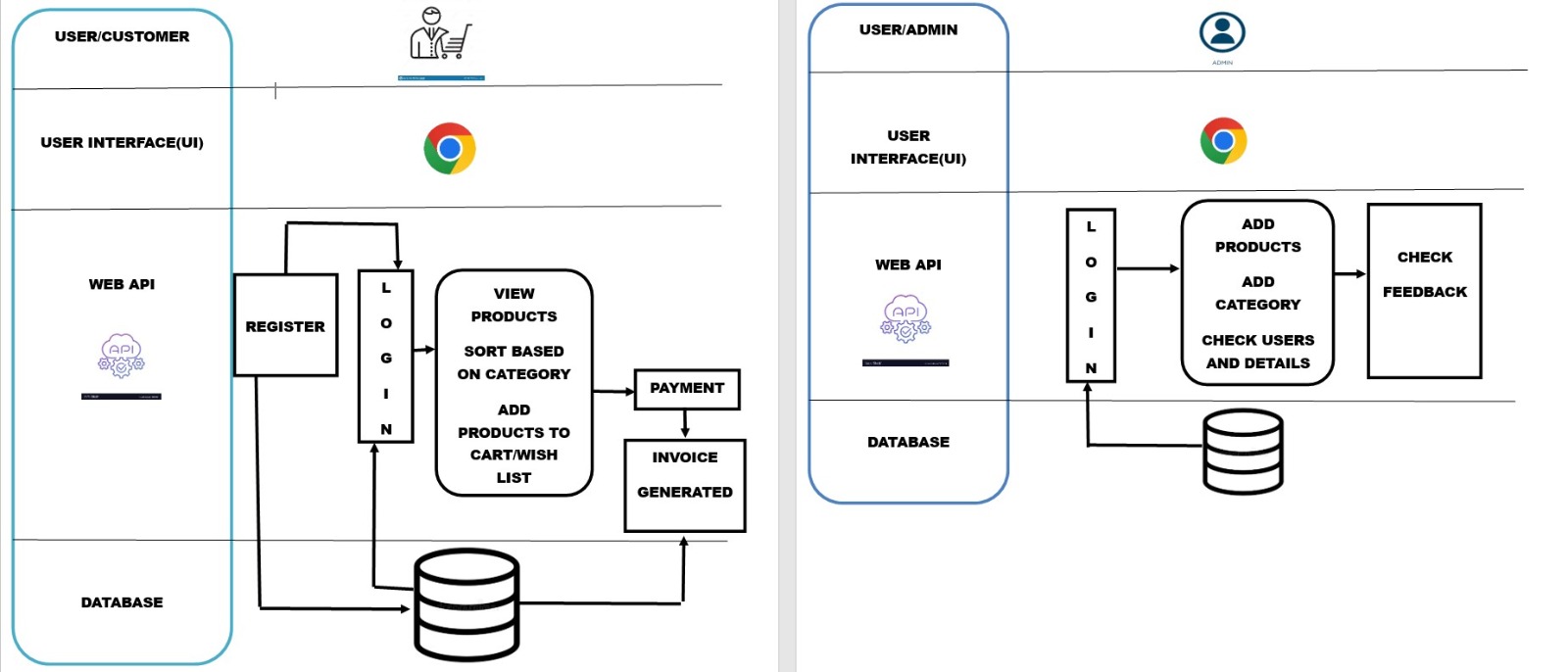
Add to Cart

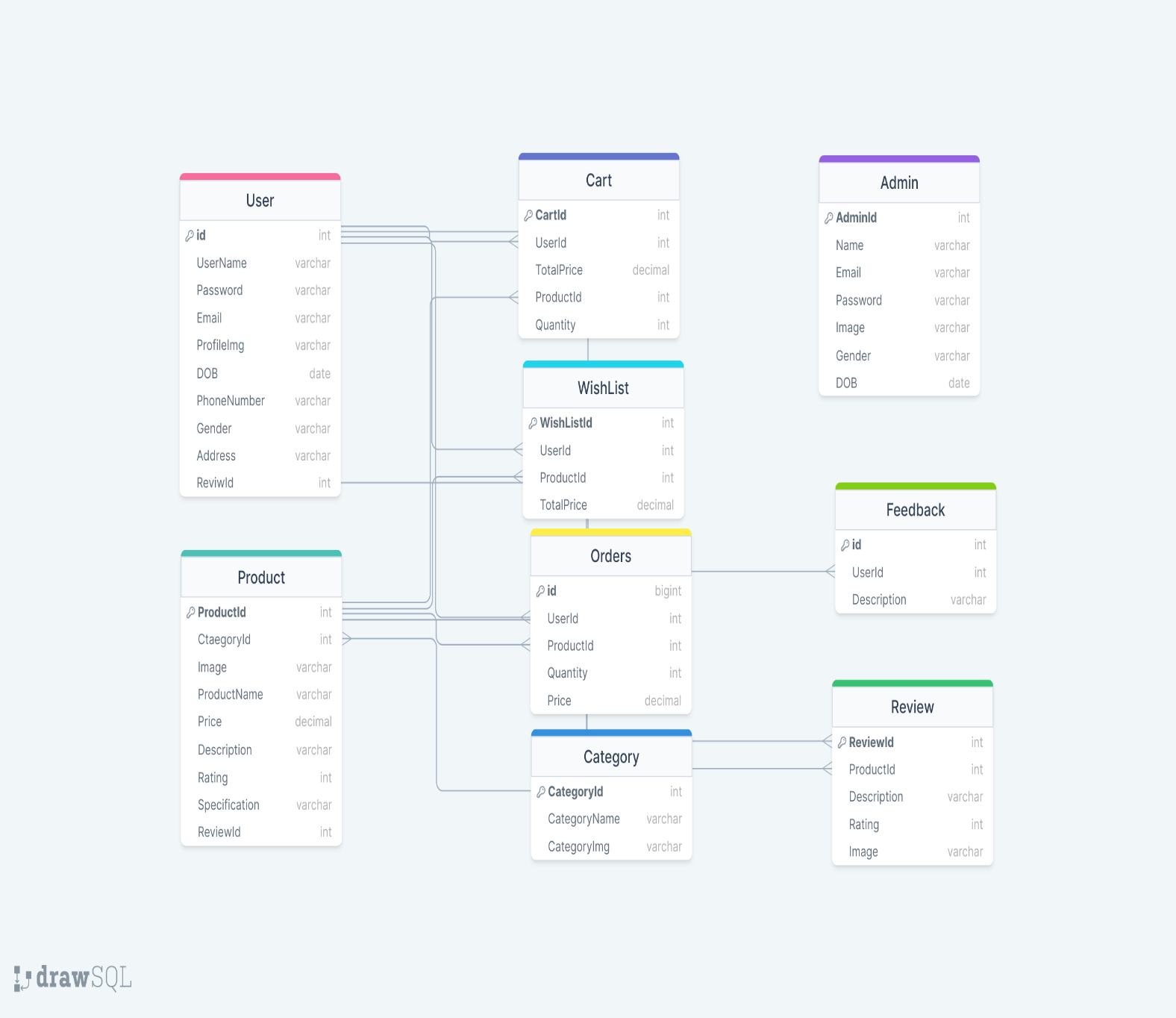
Pay

Online Pay

Order Placed

**Architecture Diagram**



****

**Database Schema**

**Modules**

* USERS

1. **Unlogged-In Users**

* Login and Sign-Up Page
* Home-Page
* Forgot Password
* Reviews
* About
* Category Page
* Product category Page
* Product Page

1. **Logged-In Users**

* Home-Page
* Product Page
* Cart
* WishList
* Category Page
* Product category Page
* Product Page
* Previous Orders Page
* User Profile Page
* Reviews
* About
* ADMIN

* Dashboard Page
* Profile Page
* Add category Page
* Check User
* Check Feedback

**SOLUTION STEPS FOR VARIOUS PAGES:-**

**CUSTOMER:-**

1. **Customer\_LogIn\_Page:-**
2. User navigates to the login page of the ecommerce web application and enters their username and password and clicks the login button.
3. The browser directs the request to the customer login API endpoint.
4. The API gateway does the routing and forwards the request to the customerLoginHandler.handle function.
5. The handle function will call the doProcess() function, which will call the customerAuthenticationValidator.validateCustomerAuthentication() function to validate the input.
   1. If validation fails, then it will return an error code and error description with a corresponding HTTP error response code.

b. If validation is successful, then the handler will call the customerAuthenticationService.authenticateCustomer() function, which will call the customerAuthenticationRepository.authenticateCustomer() function to check the username and password against the database.

i. If the authentication is successful, the service will generate an access token and return it to the handler.

ii. If the authentication fails, the service will return an error code and error description to the handler with a corresponding HTTP error response code.

1. The handler sends the access token to the user in the response body with a corresponding HTTP success response code.
2. The user can use the access token to access protected resources in the ecommerce web application.

**2) Customer\_Registration/Sign\_Up\_Page:-**

1) User navigates to the registration page of the ecommerce web application and enters the required details such as first Name, last Name, email, password, address, phone number, gender, age, and clicks the submit button.

1. The browser directs the request to the customer registration API endpoint.
2. The API gateway does the routing and forwards the request to the customerRegistrationHandler.handle function.
3. The handle function will call the do Process() function, which will call the customerRegistrationValidator.validateCustomerRegistration() function to validate the input. a. If validation fails, then it will return an error code and error description with a corresponding HTTP error response code. b. If validation is successful, then the handler will call the customerRegistrationService.registerCustomer() function, which will call the customerRegistrationRepository.saveCustomer() function to store the data in the database.

i. If registration is successful, the service will generate an access token and return it to the handler.

ii. If registration fails, the service will return an error code and error description to the handler with a corresponding HTTP error response code.

1. The handler sends the access token to the user in the response body with a corresponding HTTP success response code.
2. The user can use the access token to access protected resources in the ecommerce web application.

**3) User\_Forgot\_Password:-**

1. User navigates to the "Forgot Password" page of the ecommerce web application and enters their email address.
2. The browser directs the request to the "Forgot Password" API endpoint.
3. The API gateway does the routing and forwards the request to the forgotPasswordHandler.handle function.
4. The handle function will call the do Process() function, which will call the forgotPasswordValidator.validateEmail() function to validate the input email. a. If validation fails, then it will return an error code and error description with a corresponding HTTP error response code. b. If validation is successful, then the handler will call the forgotPasswordService.sendPasswordResetEmail() function, which will generate a password reset token and send an email to the user with a link to reset their password.
   1. If sending the email is successful, the service will return an HTTP success response code to the handler.
   2. If sending the email fails, the service will return an error code and error description to the handler with a corresponding HTTP error response code.
5. The handler sends an HTTP success response code to the user with a message informing them that an email has been sent to their email address with instructions to reset their password.
6. The user receives the email with the password reset token and follows the instructions to reset their password.
7. The user navigates to the "Reset Password" page of the ecommerce web application and enters their new password along with the password reset token received in the email.
8. The browser directs the request to the "Reset Password" API endpoint.
9. The API gateway does the routing and forwards the request to the resetPasswordHandler.handle function.
10. The handle function will call the doProcess() function, which will call the resetPasswordValidator.validateResetPassword() function to validate the input password and reset token. a. If validation fails, then it will return an error code and error description with a corresponding HTTP error response code. b. If validation is successful, then the handler will call the resetPasswordService.resetPassword() function, which will update the user's password in the database with the new password. i. If resetting the password is successful, the service will return an HTTP success response code to the handler. ii. If resetting the password fails, the service will return an error code and error description to the handler with a corresponding HTTP error response code.
11. The handler sends an HTTP success response code to the user with a message informing them that their password has been reset successfully.

**ADMIN:-**

1. **Admin\_Add\_Product\_Category:-**
2. The admin logs into the admin panel of the ecommerce web application using their credentials.
3. The admin navigates to the "Product Category" section within the admin panel.
4. The admin clicks on the "Add New Category" button to add a new product category.
5. The admin is presented with a form where they can enter the details of the new product category, such as the name, description, and any other relevant information.
6. Once the admin has entered all the required information, they click on the "Save" button to save the new product category.
7. The application validates the input data to ensure that it meets the required format and that the required fields are not left empty. If any validation errors occur, the application displays an error message to the admin, prompting them to correct the errors and resubmit the form.
8. If the data validation is successful, the application stores the new product category in the database.
9. The admin is presented with a confirmation message indicating that the product category has been successfully added.
10. The admin can view the newly added product category in the list of product categories and can also edit or delete it if needed.

**2) Admin\_Add\_Product:-**

1. Admin navigates to the "Add Product" page on the admin panel of the ecommerce website.
2. Admin fills in the required details for the product, such as product name, description, price, category, and image.
3. Once the details are filled, Admin clicks on the "Add Product" button to submit the product details to the server.
4. The request is sent to the API gateway of the ecommerce website.
5. The API gateway routes the request to the appropriate handler function, "addProductHandler".
6. The "addProductHandler" function calls the "doProcess" function to process the request.
7. The "doProcess" function calls the "productSchemaValidator.doValidate" function to validate the product details. The "productSchemaValidator" is responsible for validating the product schema to ensure all required fields are present and have the correct data types.
8. If the validation is successful, the "doProcess" function calls the "addProductService.addProduct" function. This service function adds the product to the database by calling the "addProductRepository.addProduct" function.
9. The "addProductRepository" function adds the product details to the database.
10. If the product is successfully added to the database, a success message is sent back to the "addProductHandler" function.
11. The "addProductHandler" function sends a success message to the admin panel, indicating that the product has been successfully added to the website.
12. If there is an error, the error message is sent back to the "addProductHandler" function, which then displays the error message on the admin panel.

**3) Admin\_Check\_Feedback:-**

1. Admin logs into the admin panel of the ecommerce web application.
2. The admin navigates to the feedback management section.
3. The feedback management section displays a list of all the feedback received from the users.
4. The admin can filter the feedback based on various criteria such as date, rating, product, etc.
5. The admin can view the details of each feedback item, including the user who submitted it, the date of submission, the rating, and the feedback text.
6. The admin can also take action on the feedback, such as replying to the user, deleting the feedback, or marking it as resolved.
7. If the feedback requires further investigation, the admin can assign it to a customer support representative for follow-up.
8. Once the feedback has been resolved, the admin can mark it as resolved and add any notes for future reference.
9. The admin can also generate reports on the feedback data, such as the number of feedback items received in a particular period, the average rating, and the most commonly reported issues.
10. Finally, the admin can log out of the admin panel.

**Data Model/Table**

**1.User**

|  |  |
| --- | --- |
| Table Id | Data Type |
| UserId | int |
| UserName | varchar(20) |
| Password | varchar(20) |
| Email | varchar(50) |
| ProfileImg | image |
| DateOfBirth | date |
| PhoneNumber | varchar(12) |
| Gender | varchar(17) |
| Address | varchar(100) |

**2.Admin**

|  |  |
| --- | --- |
| Table Id | Data Type |
| AdminId | int |
| Name | varchar(20) |
| Email | varchar(20) |
| Password | varchar(20) |
| Image | image |
| Gender | varchar(7) |
| DateOfBirth | date |

**3.Cart**

|  |  |
| --- | --- |
| Table Id | Data Type |
| CartId | int |
| UserId | int |
| TotalPrice | decimal(7,2) |
| ProductId | int |
| Quantity | int |

**4.Category**

|  |  |
| --- | --- |
| Table Id | Data Type |
| CategoryId | int |
| CategoryName | varchar(20) |
| CategoryImage | image |

**5.Product**

|  |  |
| --- | --- |
| Table Id | Data Type |
| ProductId | int |
| CategoryId | int |
| Image | image |
| ProductName | decimal(8,2) |
| Description | varchar(500) |
| Rating | int |
| Specification | varchar(2000) |
| ReviewId | int |

**6.WishList**

|  |  |
| --- | --- |
| Table Id | Data Type |
| WishListId | int |
| UserId | int |
| ProductId | int |
| TotalPrice | Decimal(7,2) |

**7.Feedback**

|  |  |
| --- | --- |
| Table Id | Data Type |
| FeedbackId | int |
| UserId | int |
| Description | varchar(2000) |

**8.Review**

|  |  |
| --- | --- |
| Table Id | Data Type |
| ReviewId | int |
| ProductId | int |
| Description | varchar(2000) |
| Rating | int |
| Image | image |