JASPER BOOK STORE

DESIGN DOCUMENT

Prepared for:

Hussein Al Osman, Associate Director and Assistant Professor School of Electrical Engineering and Computer Science (EECS) University of Ottawa

Prepared by:

Team Jasper (Varun Hanabe Muralidhara , Yan Zhang , Yash Mittal , Sheela Parthanarayana, Sehajpreet Taneja)

**ABSTRACT**

Project implements a web application which displays books of different genres. The SERVICE allows to view, select and purchase different categories of books. To facilitate this, cart and purchase order options are provided. The product is implemented using the 3 Tier MVC (Model-View-Controller) architecture approach.

For the implementation, several technologies had to be studied and worked upon. These include Multi-Tiered Architecture, Client (HTML, CSS & JavaScript) and Server-side web service implementation (Jersey implementation of JAX-RS), Relational Database Agent (Hibernate) and Relational Database (MySQL Server).

The document aims to present all the technologies used to create and implement the web application.

**TABLE OF CONTENTS**

Abstract….…………………………………………………………………………………………………………..…2

Acknowledgement………………………………………………………………………………………………….…3

**1.0** Introduction………………………………………………………………………………………………………..6

**2.0** Description of Source Code…………………………………………………………………………………….6

**2.1** Database Agent…………………………………………………………………………………………6

2.1.1 DAO Implementation

**2.2** Product Catalog Service……………………………………………………………………………….7

**2.2.1** API Layer

**2.2.2** Business Layer Logic

**2.2.3** DAO Layer

**2.3** Order Process Service………………………………………………………………………………….8

**2.3.1** API Layer

**2.3.2** Business Layer Logic

**2.3.3** DAO Layer

**2.4** Main Web Service……………………………………………………………………………………….8

**2.4.1** User Interface

**2.4.2** Controller

**2.4.3** Business Layer Logic

**3.0** Design Description…………………………………………………………………………………….…..……11

**3.1** Use Cases

**3.2** Use Case Diagram

**3.3** Sequence Diagram

**3.4** Class Diagram

**3.5** UML Diagrams

**4.0** Application Features Description……………………………………………………………………………..15

**5.0** Members’ Contributions…………………………………………………………………………………….…19

**1.0 Introduction**

The objective of this project is to develop an E-commerce Web Application based Book Store.

In this web application, the user can browse through the catalog and select the products of their interest. These items get collected in the shopping cart. Subsequently, the customer does the registration/signup and then the order is presented for confirmation with an option to modify the billing and shipping addresses. Finally, the payment information is entered and the order is placed.

**2.0 Description of Source Code**

**2.1 Database Agent**

The database agent acts as an interface between web service and the database.

**2.1.1 DAO Interface**

The DAO Interface consists of all the methods that are implemented in the DAO Implementation. All the methods being used in implementation must be declared in interface.

**๏ persist(entity)**

**๏ update(entity)**

**๏ findEntity(id, type)**

**๏ findAll(hql)**

**๏ findAllWithCondition(hql, map)**

**๏ openCurrentSession()**

**๏ openCurrentSessionwithTransaction()**

**๏ closeCurrentSessionwithTransaction()**

**๏ closeCurrentSession()**

**2.1.2 DAO Implementation**

We use hibernate implementation for DAO. Hibernate is an object-relational mapping library for Java, that provides a framework for mapping an object-oriented domain model to a traditional relational database. All the methods declared in the interface are implemented in the DAO Implementation.

**๏ persist(entity):** This method saves/persists the entity into the database.

**๏ update(entity):** This method updates the entity into the database.

**๏ findAll(hql):** This method finds/displays all the contents from the database.

**๏ findEntity(id, type):** This method finds the entity with the specific id type.

**๏ findAllWithCondition(hql, map):** This method finds the entities with the input conditions from the database.

**๏ openCurrentSession():** This method gets the session from the session factory.

**๏ openCurrentSessionwithTransaction():** This method gets the session from session factory and opens a new transaction.

**๏ closeCurrentSessionwithTransaction():** This method closes the session after committing the transaction.

**๏ closeCurrentSession():** This method closes the session.

**2.2 Product Catalog Service**

This web service handles all the product related functionalities. The user is able to access all the categories, fetch the product through product ID and categories through the category ID.

Following are the descriptions of all the classes created to manage the stated functions:

**2.2.1 API Layer**

This class is the initiator of product catalog service, exposed via 3 path end points. The transfer then progresses to business layer.

**๏ getCategory()**: This method fetches all the categories by invoking the service layer which in turn invokes the DAO layer and then builds the response.

**๏ getProductInfo(productId)**: This method fetches the product information through the productId parameter by invoking the service layer which then invokes DAO and builds response.

**๏ getProduct(categoryId)**: This method is used to fetch all the products of a particular category Id, by invoking the service layer which then invokes DAO to build the response.

**2.2.2 Business Layer Logic**

This class handles the product related processes to get the products and categories from the Database. This class interacts with service layer classes that interact with the Database.

**๏ getAllCategories()**: This method is created to get all the categories from the

database.

**๏ getAllProducts(categoryId)**: This method returns a list of products based on the category Id provided in the parameter.

**๏ getProductInfo(productId)**: This method takes product Id as parameter and returns the corresponding product.

**2.3 Order Process Service**

This service handles all the account and order related functionalities. The user is able to register, login, create order and confirm the order.

Following are the descriptions of all the classes created to manage the stated functions:

**2.3.1 API Layer**

This class is the initiator of order process service, exposed via 4 path end points. The transfer then progresses to business layer.

**๏ createAccount(accountInfo)**: This method receives a request to create a new user account. The business class then responds by creating a new user in the database.

**๏ checkAccount(accountInfo)**: This method receives the login request with an email and password. The response is then presented in accordance with the existence of the given input.

**๏ createOrder(orderInfo)**: This method receives an order creation request with the input order details. The business layer then responds with appropriate response by .

**๏ confirmOrder(orderInfo)**: This method receives a request to confirm the order. Then the business class responds with appropriate response.

**2.3.2 Business Layer Logic**

This class handles the order and account related processes. This class interacts with service layer classes, that interact with the Database.

**๏ isNewAccount(accountInfo)**: This method creates new account for user in

database, on a condition that the email does not already exists in database. This is done by invoking the service layer, which invokes DAO to build response.

**๏ checkIfAccountExists(userInfo)**: This method acts as a login credentials validator, if correct, it logs in the user else sends failure response. This is done by invoking the service layer, which invokes DAO to build response.

**๏ placeOrder(orderInfo)**: This method creates the order and saves it in the

database via the service class. This is done by invoking the service layer, which invokes DAO to build response.

**๏ authorizeOrder(orderInfo)**: This method acts as a confirmation to a previously existing order. This is done by invoking the service layer, which invokes DAO to build response.

**๏ checkTransactionsCount()**: This method checks the transaction count, if it’s a

5th request it’s denied with authorization error else it proceeds with successful confirmation. This is done by invoking the service layer, which invokes DAO to build response.

**2.4 Main Web Service**

The main service handles the Graphical User Interface and has controller logic to interact with services and databases.

Below are the descriptions of all the elements created to handle the application features:

**2.4.1 UI Layer ( View )**

**๏ index.jsp**: This jsp page displays all the products present in the database to the user. By default its set to Category as Computers. The product catalog web service plays its role here.

**๏ detail.jsp**: This jsp displays all the information for a particular product when

selected, with an Add to Cart option as well.

**๏ checkout.jsp**: This jsp displays the shopping cart of user along with the quantity and total price.

**๏ login.jsp**: This jsp acts as both LogIn and SignUp page. It is mandatory to LogIn to place the order. JavaScript validations are performed here.

**๏ order.jsp**: This jsp displays the order details along with an option to update and review the shipping and billing information.

**๏ payment.jsp**: This jsp allows to provide credit card and other payment details for the order confirmation. Javascript validations are done here.

**๏ success.jsp**: This jsp presents successful confirmation of order.

**๏ welcome.jsp**: This jsp is presented when there is an unexpected error or the page you are looking is not found.

**2.4.2 Controller Logic**

This controller controls the complete flow of the application. Whenever there is a request, controller decides on the method to be invoked. Controller is implemented via the servlets presented below:

**๏ Home Servlet**: The requests to index page or the home page are moved to this

controller. Following are the methods in this controller:

**๏ doPost(request, response):** HTTP POST request is made

**๏ doGet(request, response):** HTTP GET request is made

**๏ constructPath(request, response):** URL Path is created

**๏ Detail Servlet**: The requests to detail page are moved to this controller. Following are the methods in this controller:

**๏ doPost(request, response):** HTTP POST request is made

**๏ doGet(request, response):** HTTP GET request is made

**๏ Checkout Servlet**: The requests to index page or the home page are moved to this controller. Following are the methods in this controller:

**๏ doPost(request, response):** HTTP POST request is made

**๏ doGet(request, response):** HTTP GET request is made

**๏ Login Servlet**: The requests to index page or the home page are moved to this controller. Following are the methods in this controller:

**๏ doPost(request, response):** HTTP POST request is made.

**๏ doGet(request, response):** HTTP GET request is made.

**๏ checkIfValid(session):** If Cart is empty, exception is made.

**๏ SignUp Servlet**: The requests to index page or the home page are moved to this controller. Following are the methods in this controller:

**๏ doPost(request, response):** HTTP POST request is made.

**๏ doGet(request, response):** HTTP GET request is made.

**๏ Logout Servlet**: The requests to index page or the home page are moved to this controller. Following are the methods in this controller:

**๏ doPost(request, response):** HTTP POST request is made

**๏ doGet(request, response):** HTTP GET request is made

**๏ logOut(request, response):** Invalidates the session and clears DB entry.

**๏ Order Servlet**: The requests to index page or the home page are moved to this controller. Following are the methods in this controller:

**๏ doPost(request, response):** HTTP GET request is made

**๏ doGet(request, response):** HTTP POST request is made

**๏ cleanSession(session):** Invalidates the session

**2.4.3 Business Layer Logic**

This logic is used to manage the order and account related functions, along with interaction with the database.

**๏ Home Servlet**: The requests to index page or the home page are moved to this controller. Following are the methods in this controller:

**๏ getCategory():** This Method is used to fetch entire category list

**๏ getBooks(category): This** Method is used to fetch entire product list based on category Id

**๏ Detail Servlet**: The requests to index page or the home page are moved to this controller. Following are the methods in this controller:

**๏ getBook(bookId):** This Method is used to fetch entire detail for a product

**๏ Checkout Servlet**: The requests to index page or the home page are moved to this controller. Following are the methods in this controller:

**๏ constructPurchaseOrder(session, bookIds):** This Method is used to construct the purchase order in database

**๏ getBookids(Map, item):** This Method is used to fetch all the book ids in the cart

**๏ LogIn Servlet**: The requests to index page or the home page are moved to this controller. Following are the methods in this controller:

**๏ getUserDetails(request):** This Method is used to fetch user’s login id and password

**๏ SignUp Servlet**: The requests to index page or the home page are moved to this controller. Following are the methods in this controller:

**๏ constructAccount(request):** This Method is used to construct a user account

**๏ LogOut Servlet**: The requests to index page or the home page are moved to this controller. Following are the methods in this controller:

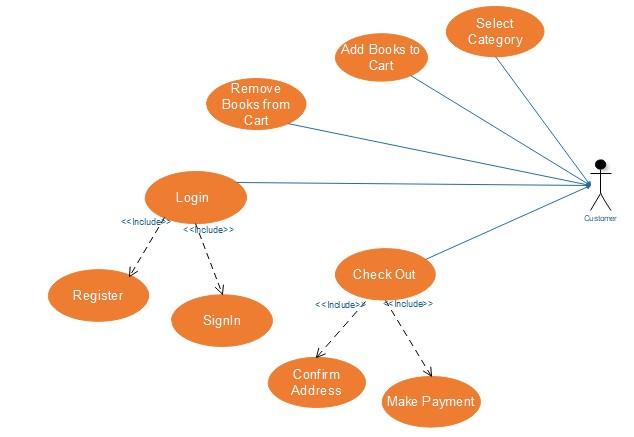
**๏ logOut(request, response):** Logouts the user and invalidates the session

**๏ Order Servlet**: The requests to index page or the home page are moved to this controller. Following are the methods in this controller

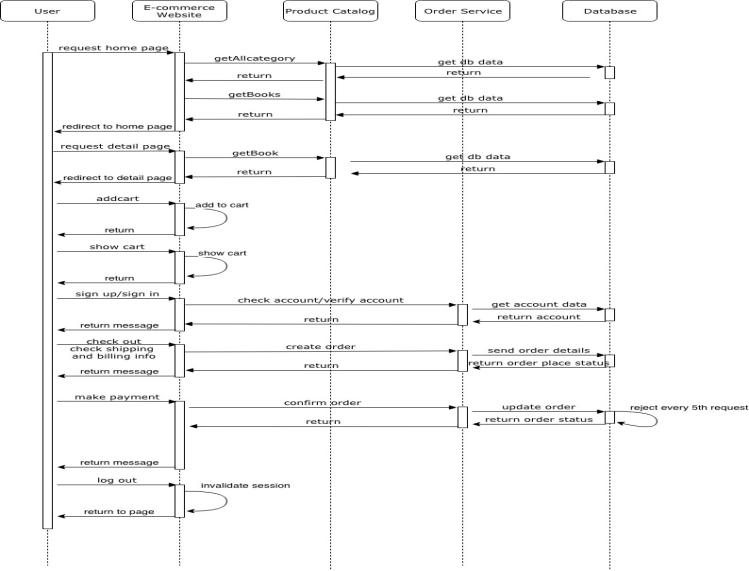
**๏ createConfirmOrder(purchaseId):** Confirms the order based on the purchase id after the payment is done

**3.0 Design Description**

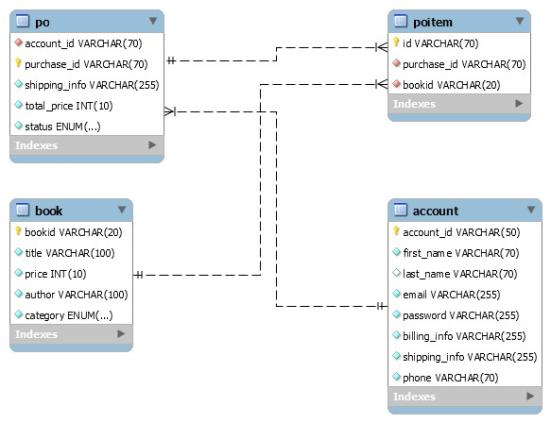
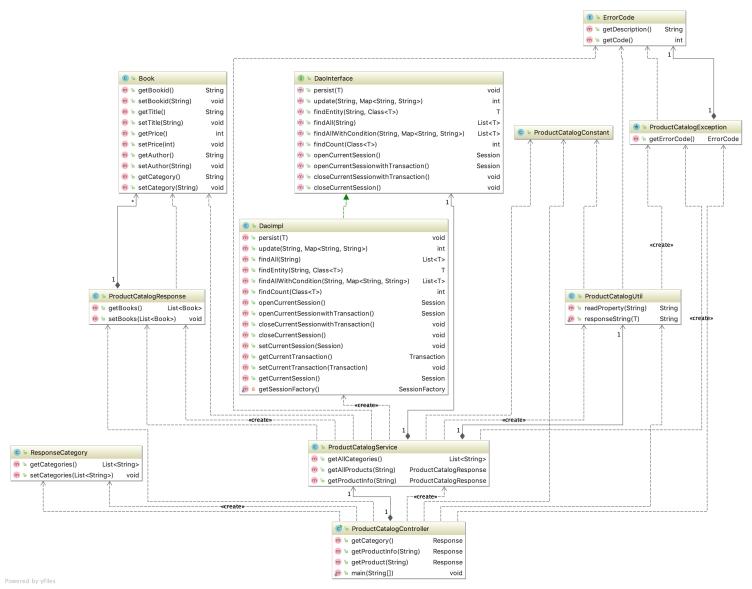
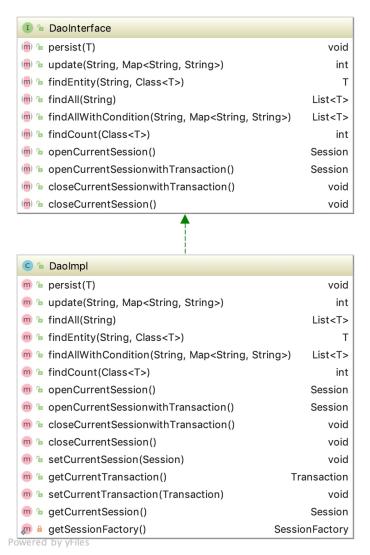
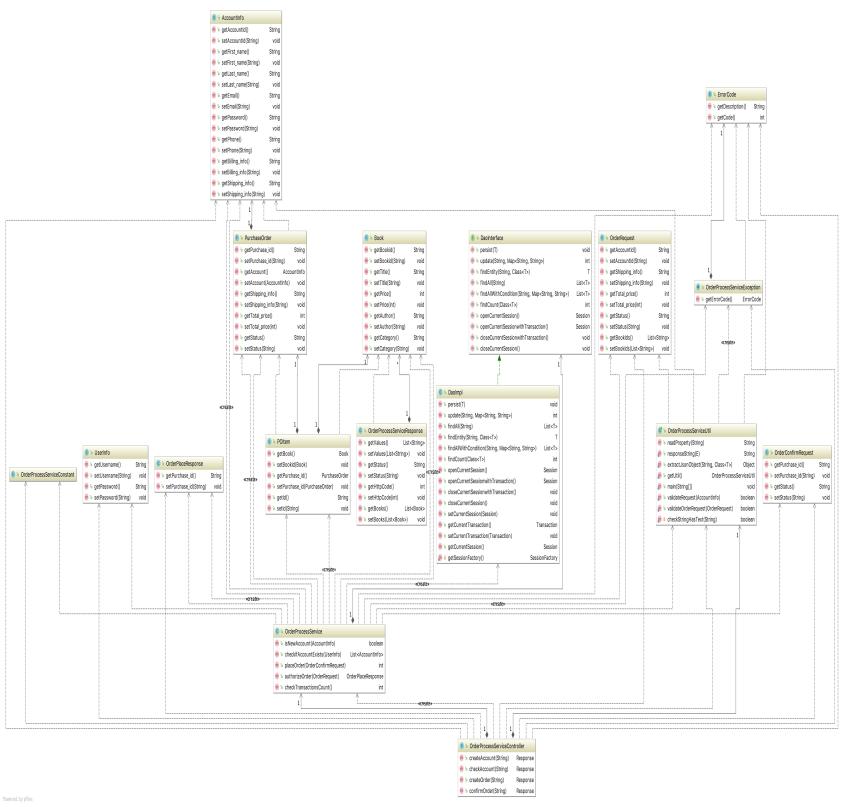
**3.1 Use Case Diagram**



**3.2 Sequence Diagram**



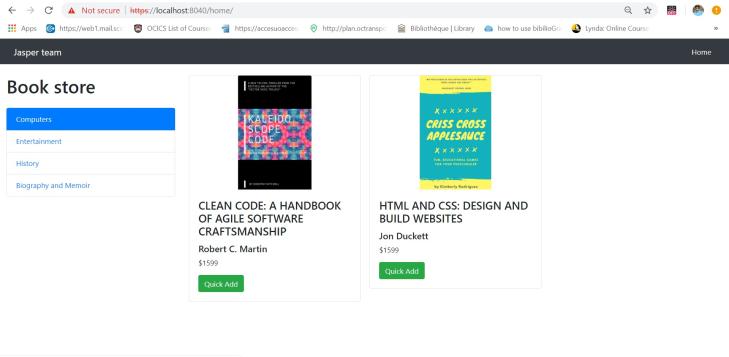
**3.3 UML Diagrams**



**4.0 Application Feature Description**

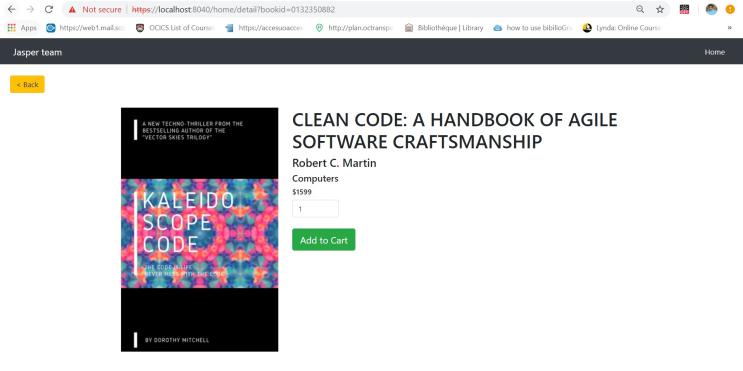
**4.1 Home Page**

The home page allows to browse the books through list of categories. Categories put up on left side with product on right side



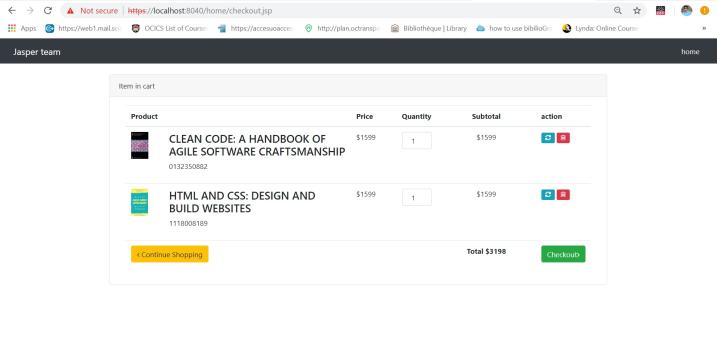
**4.2 Detail Page**

The detail page displays information regarding a particular product based on the product id.



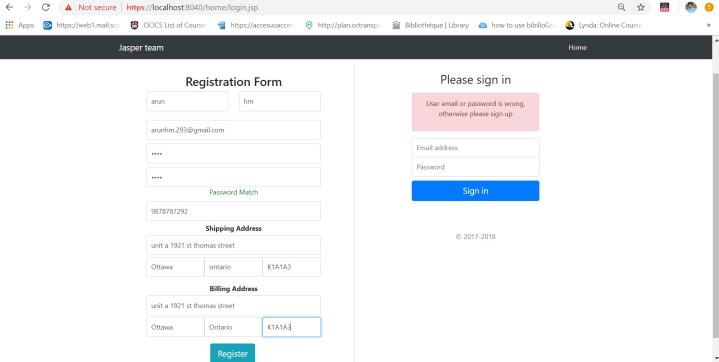
**4.3 Checkout Page**

The detail page displays the books selected along with their quantity and total price. Checkout button is displayed, which takes on to next pages to confirm the order.



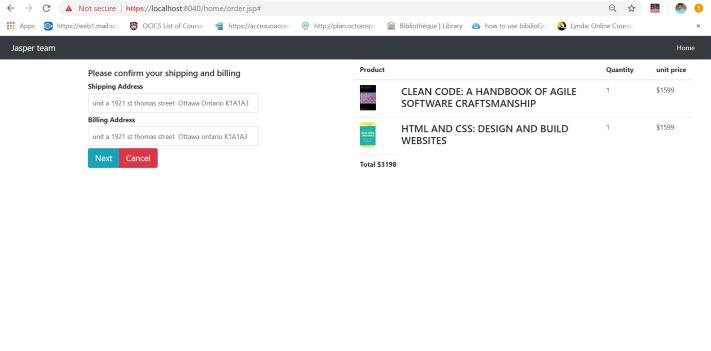
**4.4 Login/SignUp Page**

The LogIn/SignUp page asks user to either login or register before confirming the order for final payment. All the user details are submitted in the form and stored in Database.



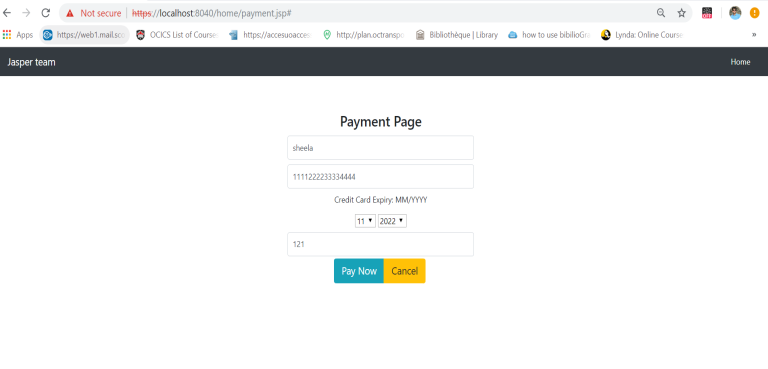
**4.5 Order Page**

This page displays the order on one side and address confirmation on the other side. User has the option to modify the shipping and billing information.



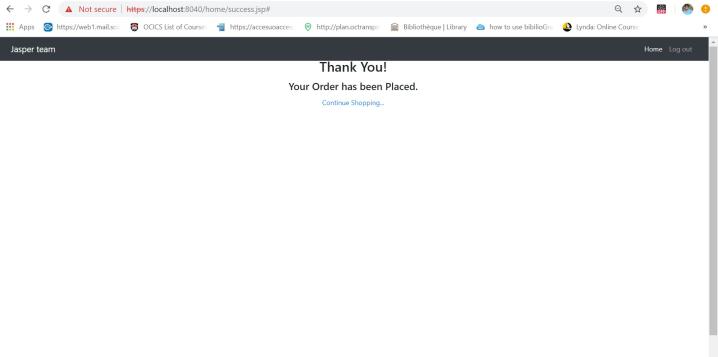
**4.6 Payment Page**

The payment page displays the option to input the credit card details and then submit to confirm the order.



**4.7 Success Page**

This page displays the thank you message after confirmation of order.



**5.0 Member Contributions**

**5.1 Varun Hanabe Muralidhara (SID: 300055628)\***

Product Catalog and Order Catalog Web Service

DB agent

Checkout Product ( Logic ) Serenity BDD( Contribution)

Code Refactoring ( Contribution )

**5.2 Yan Zhang (SID: 300052103)**

Web Page Servlets & Filters ( All Except Login & SignUp ) Web Pages ( All Except Login/SignUp, Payment & Success ) Shopping Cart ( Logic )

Code Refactoring ( Contribution )

**5.3 Yash Mittal (SID: 300051651)**

Web Page Servlets ( Login & SignUp )

Web Pages ( Login/SignUp, Payment & Success ) JavaScript Validations on All JSP’s

Front End Automation Utility

**5.4 Sheela P Acharya (SID: 300068677)**

Hibernate and

MVC (Contribution)

JUnit for Backend Web Services Design Document

Test Plan

**5.5 Sehajpreet Taneja (SID: 300066562)**

Order JSP Page ( Contribution ) Serenity BDD ( Contribution ) Test Plan Document Documentation