JAVA INSTITUTE
for Advanced Technology
INNOVATIONS FOR THE KNOWLEDGE SOCIETY

Web Application Development II

Project Documentation

Name: E. Kishan Chirantha

NIC: 200232702267

Batch: Narwhal Uni

1. Acknowledgment

I would like to express my sincere gratitude to everyone who contributed to the successful completion of this project. I am deeply thankful to my lecturer, Tharaka sir, for their valuable guidance, continuous support, and constructive feedback throughout the development of this gift box ordering web application. I also extend my appreciation to my classmates and friends for their encouragement and helpful suggestions during the project's progress. Special thanks go to my friends who participated in the calls and surveys, providing the essential information that helped shape the features of the system. Finally, I am grateful to my family for their constant motivation and understanding, which enabled me to focus on and complete this work successfully.

2. Table Of Content

3. Introduction

Gift-giving has long been a way for people to express appreciation, love, and celebration. In the modern world, however, finding the time to visit multiple stores, select products, and package them attractively can be challenging. This project presents a web-based gift box ordering platform designed to make gifting both convenient and highly customizable. The application ensures that customers always purchase complete gift boxes rather than individual items, maintaining the concept of a curated gift experience. Upon visiting the site, users are greeted on the index page, where they can view a variety of available gift box designs. To place an order, a customer first selects a gift box and then proceeds to choose from a range of items to include within it. After customizing their selection, they can review their order in the cart and proceed to secure online payment. The platform also features a role-based login system: clients are directed to the shopping interface, while administrators access a dedicated dashboard for adding or managing gift boxes and items. This combination of intuitive navigation, customization options, and administrative tools creates a seamless experience for both customers and business owners.

4. Problem Analysis

The traditional approach to buying gifts often requires customers to visit multiple stores or browse several online platforms to find the perfect combination of products. This process is not only time-consuming but also inconvenient, especially for individuals with busy schedules or those living far from physical stores. Many existing online gift shops focus on selling individual products or offer only fixed pre-packaged sets, leaving little room for personalization. As a result, customers are unable to create a meaningful and fully customized gift package that reflects the recipient's preferences.

From the business perspective, small and medium-sized gift retailers face challenges in reaching a wider audience, as physical locations limit their customer base. Without a dedicated online platform, these businesses struggle to keep inventory updated in real-time and efficiently manage orders. They also miss opportunities to increase sales value, as customers might buy only single products rather than complete packages. The absence of an integrated payment system further complicates order processing and delays fulfillment, reducing customer satisfaction.

Another key issue is the lack of guided purchasing flows in many existing online gift platforms. Even when customization options are available, the process is often unstructured, requiring customers to manually add items without any clear sequence. This can cause confusion, abandoned carts, and lost sales. For administrators, managing products and gift sets manually without centralized tools leads to inefficiencies, mistakes in stock records, and slower updates to product listings.

The proposed gift box ordering web application directly addresses these issues by enforcing a structured purchase flow that begins with selecting a gift box and then choosing items to fill it. This ensures that every order is complete, enhances the gifting experience, and increases the business's average order value. With role-based access, the system provides customers with a simple shopping experience and administrators with a powerful dashboard for product and inventory management. Together, these features bridge the gap between customer expectations and business capabilities, creating a solution that benefits both sides.

5. Project Plan

i. Features of the System

The gift box ordering web application provides a user-friendly, step-by-step shopping experience that ensures customers can easily create customized gift packages. The platform begins with an index page displaying all available gift box designs, allowing users to choose a box before selecting items. This structured flow ensures that every purchase is a complete gift package rather than individual products, maintaining a premium brand image. Once a box is selected, the customer is taken to the item selection page, where they can browse a catalog of products and add them to their box. The cart page provides a clear summary of the chosen box, selected items, and total cost before checkout. Integrated online payment processing allows for secure transactions. The system also features a role-based login mechanism, directing customers to the shopping interface and administrators to the admin dashboard. From the dashboard, admins can add or edit gift boxes, manage product listings, and maintain stock levels efficiently.

ii. Limitations

While the application offers a smooth and functional shopping experience, it has a few limitations. The system requires a stable internet connection for all operations, meaning it cannot be used offline. Payment processing depends on third-party gateways, which may impose transaction fees or have regional restrictions. Delivery tracking is not built directly into the system and instead relies on external courier services. Additionally, the platform currently supports only one language and currency, which may limit accessibility for international customers. Although the system allows customization of gift boxes, the available selection is limited to the inventory added by the administrator, and customers cannot upload or request items outside of the existing catalog.

iii. Future Enhancements

To further improve functionality and expand its market reach, several enhancements are planned for future versions of the system. Real-time delivery tracking integration would allow customers to follow their orders from dispatch to arrival. Multi-language and multi-currency support would make the platform accessible to a broader global audience. AI-powered product recommendation features could help customers discover items based on their previous purchases or preferences, improving the shopping experience and increasing sales. Additional customization options, such as selecting wrapping styles and adding personalized greeting cards, would make gift boxes even more unique. A loyalty points system could also be introduced to encourage repeat purchases and strengthen customer retention.

6. Research

Interviews: Conducted with potential customers and gift shop owners to gather qualitative insights on gifting preferences, customization needs, and challenges with existing options.

Selection rationale: In-depth discussions allowed us to clearly understand user expectations and the operational issues faced by gift businesses.

Questionnaires: Distributed to a broader group of potential customers to collect quantitative data about preferred gift types, customization trends, and willingness to purchase online.

Selection rationale: Quantifiable responses helped identify market patterns and confirmed the demand for a customizable online gift box platform.

Observation: Spent time observing customer behavior in physical gift stores, focusing on product selection habits, packaging choices, and seasonal demand trends.

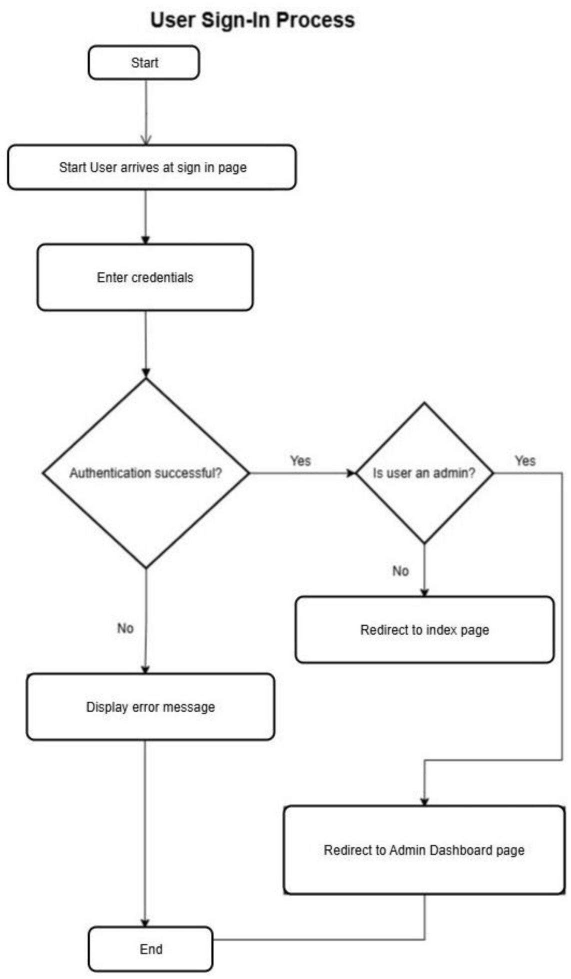
Reason for selection: Observation validated findings from interviews and questionnaires by revealing real-world customer decision-making processes.

Record Searching: Reviewed competitor websites and industry reports on the online gifting sector to identify strengths, weaknesses, and gaps in the market.

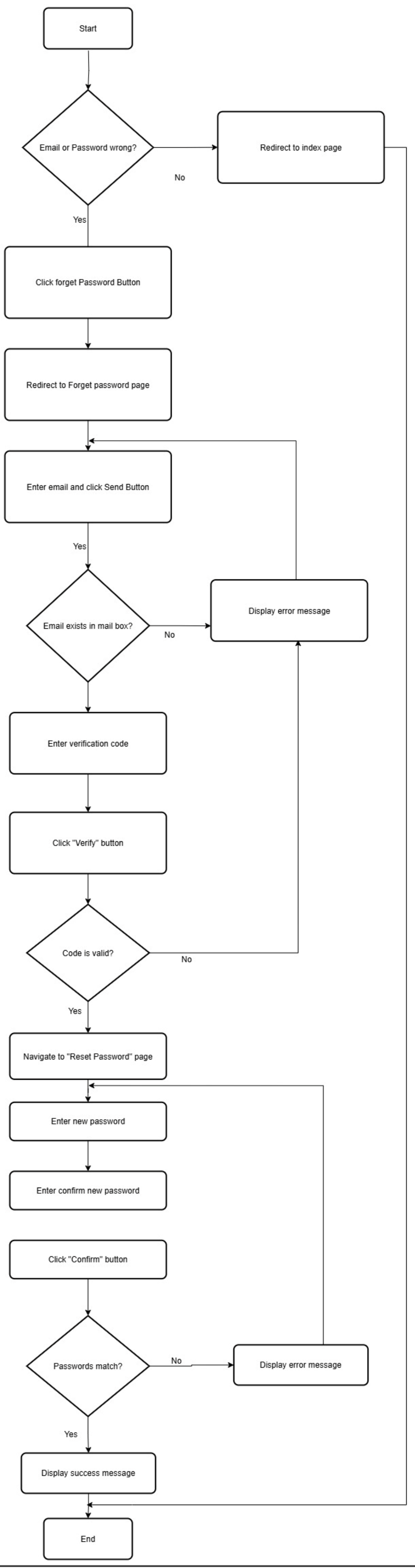
Selection rationale: This secondary research provided benchmarks and highlighted opportunities for differentiation in Boxella’s design and offerings.

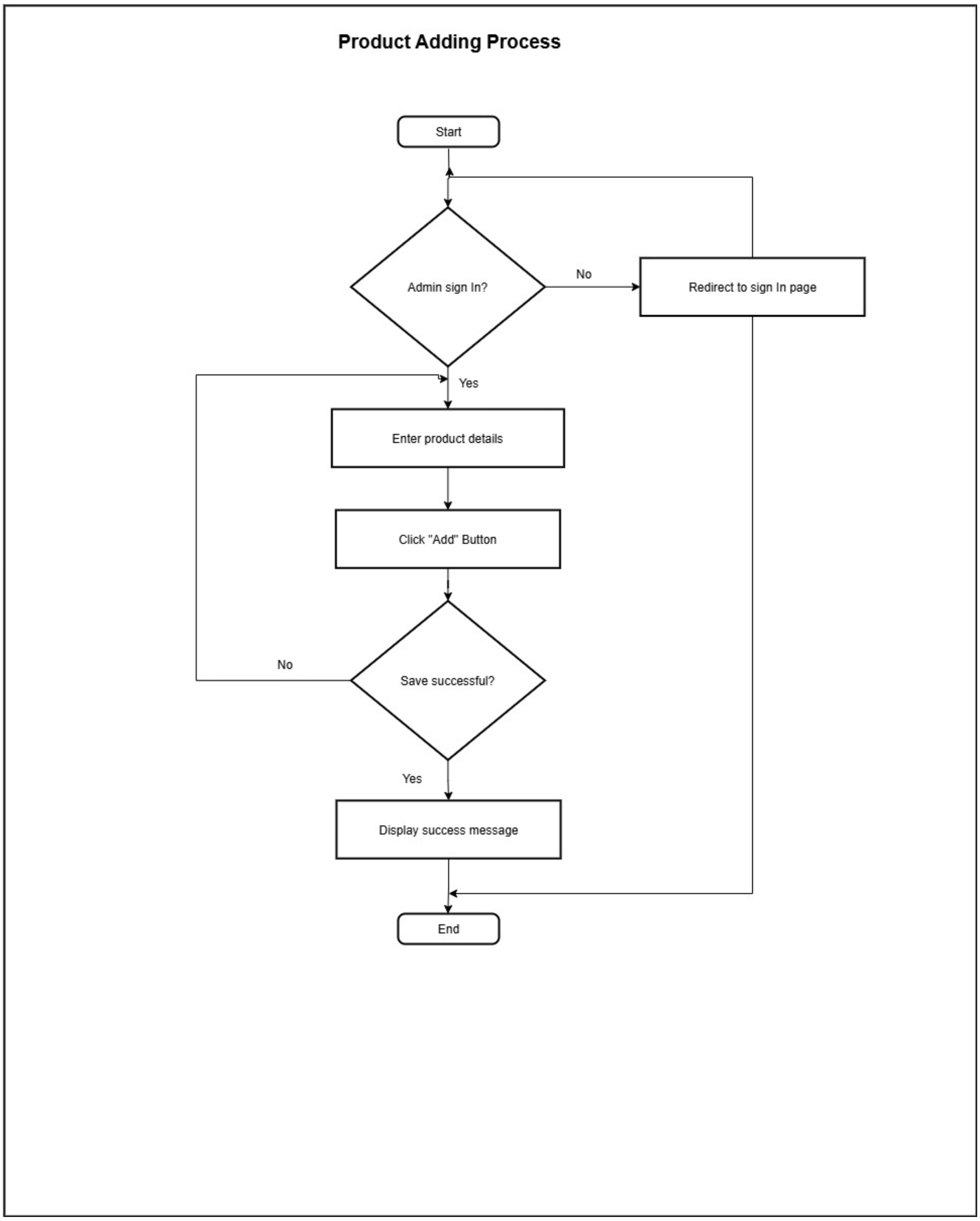
7. Analysis

I. Business Process Diagrams

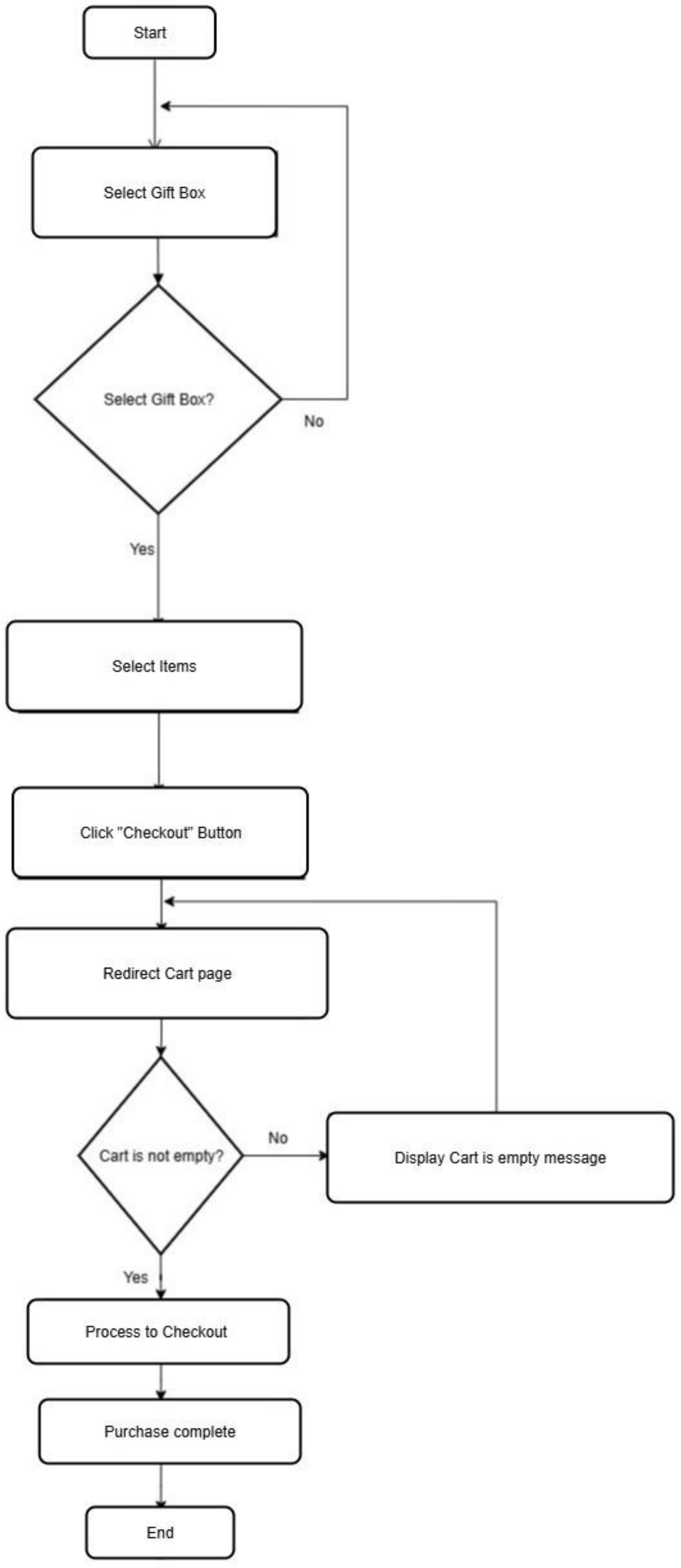


Forget Password Process

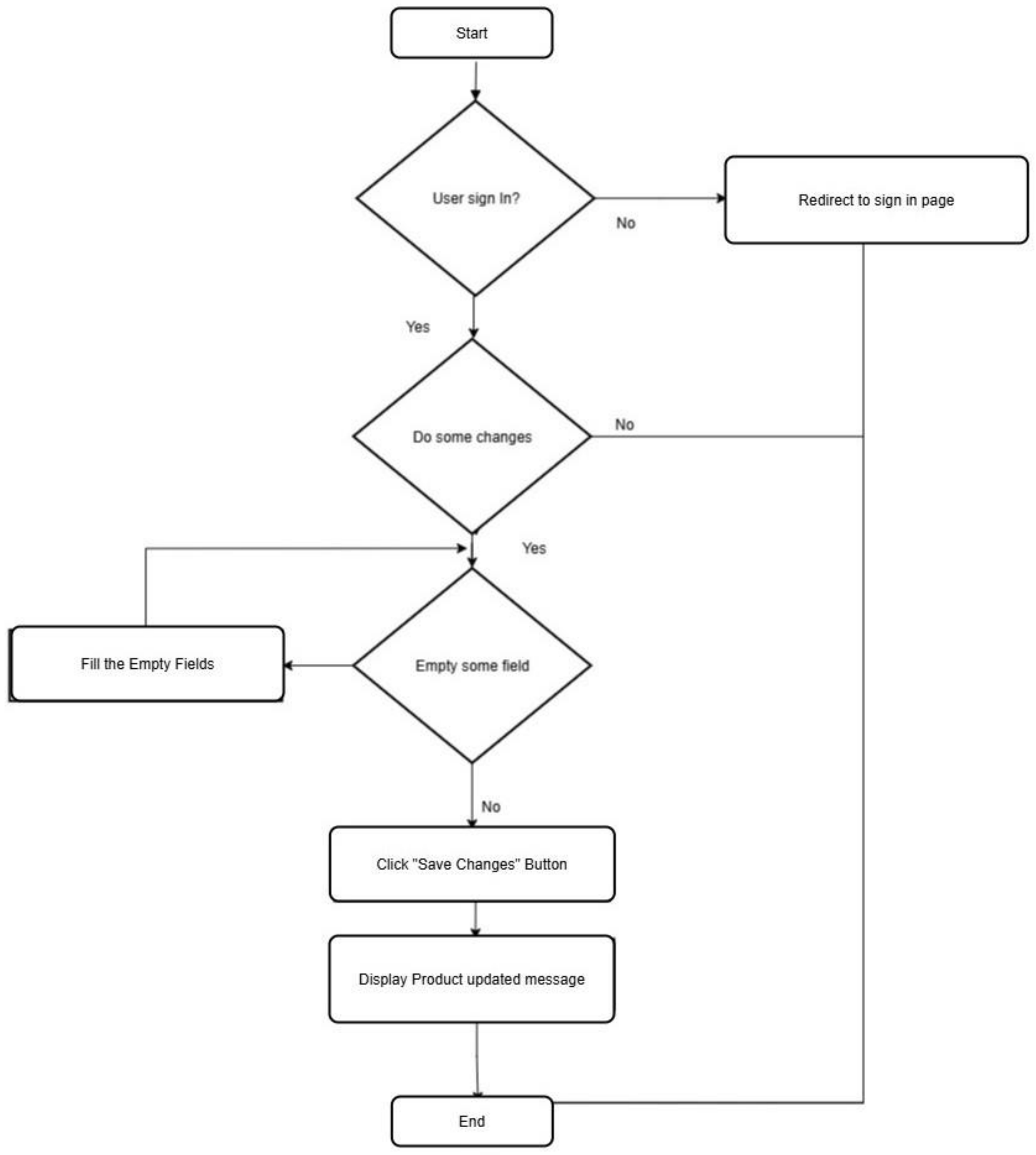


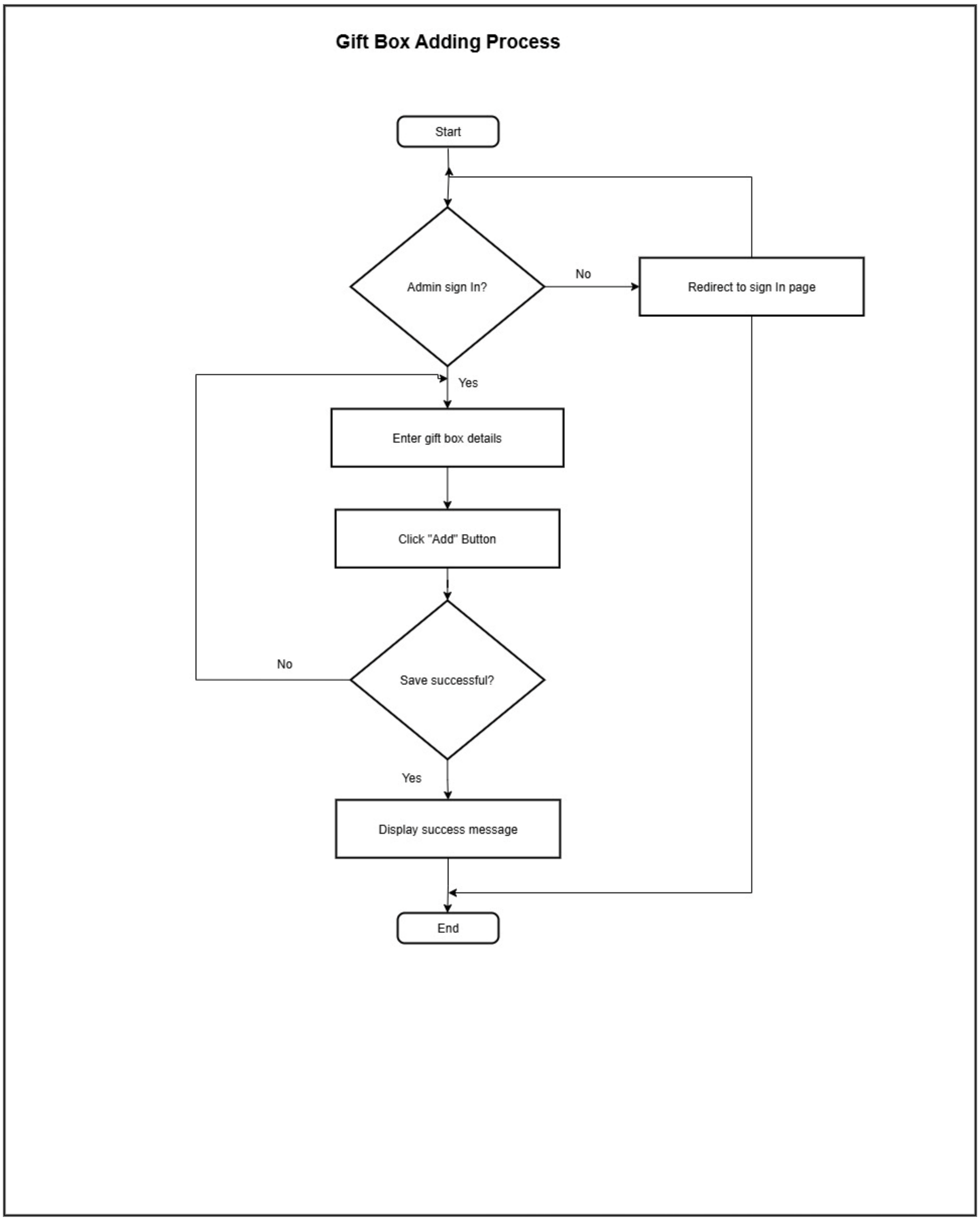

Product Adding Process
Start
Admin sign In?
No
Redirect to sign In page
Yes
Enter product details
Click "Add" Button
Save successful?
No
Yes
Display success message
End

Checkout Process



Profile Updating Process

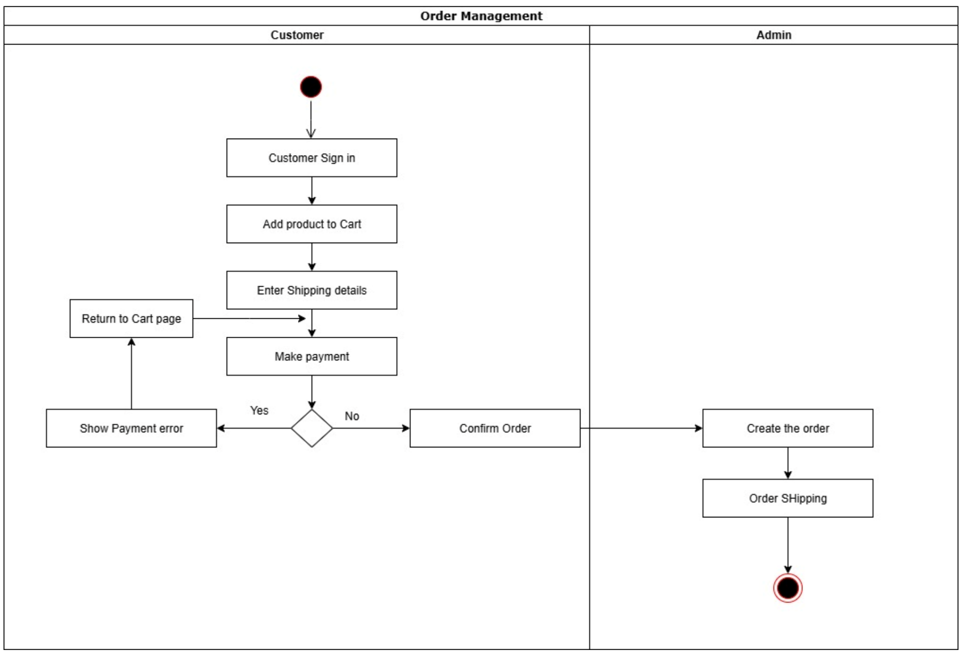


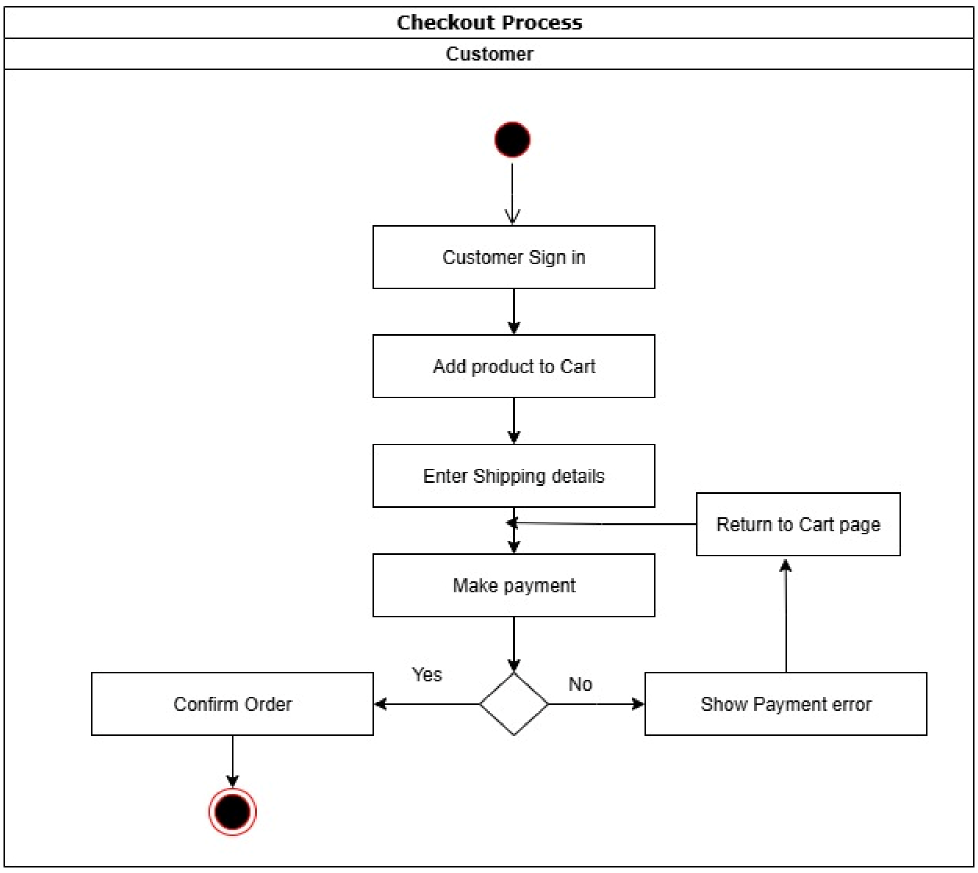

Gift Box Adding Process
Start
Admin sign In?
No
Redirect to sign In page
Yes
Enter gift box details
Click "Add" Button
Save successful?
No
Yes
Display success message
End

8. Activity Diagram

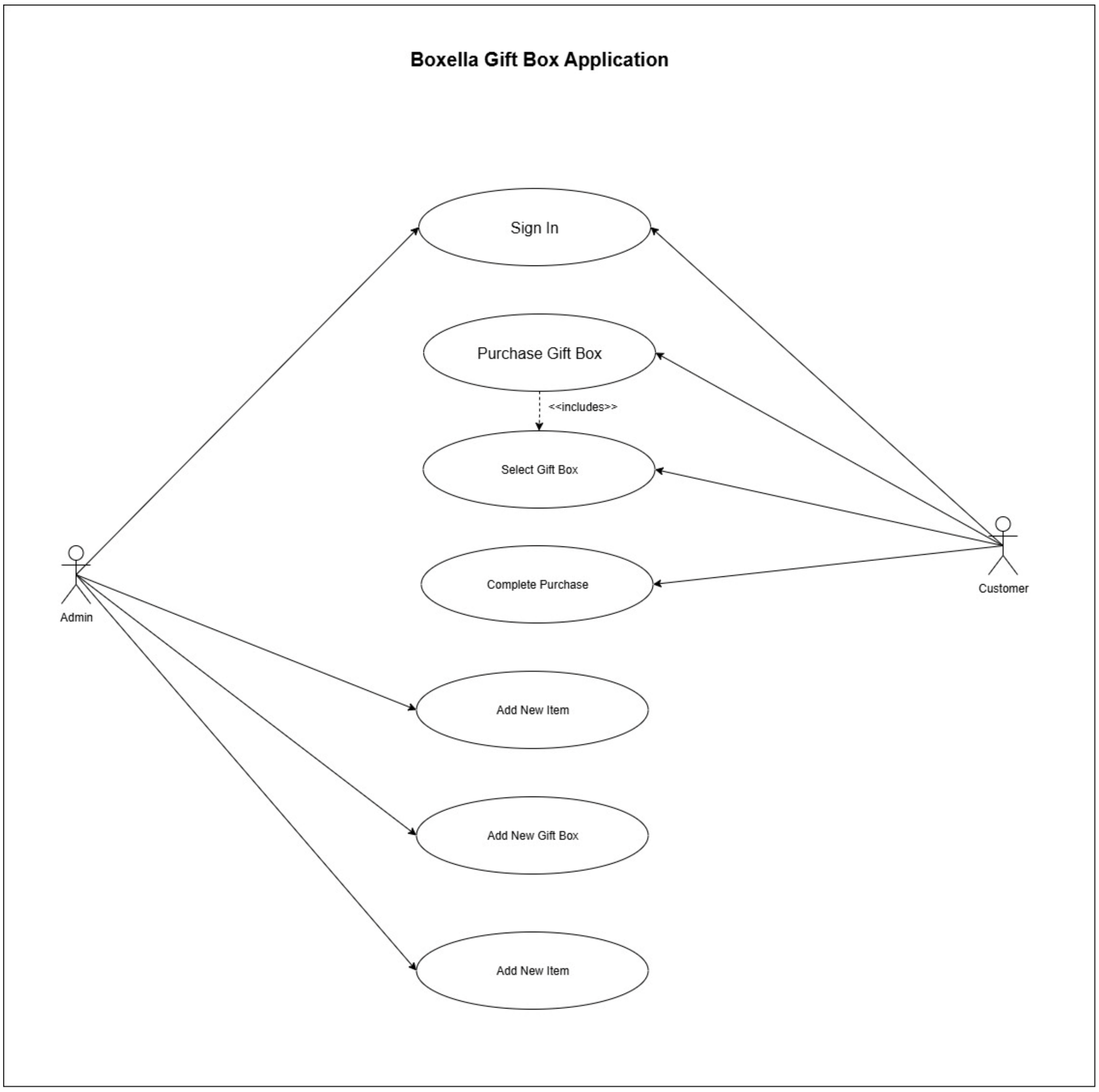
I. System Design

i. Activity Diagram



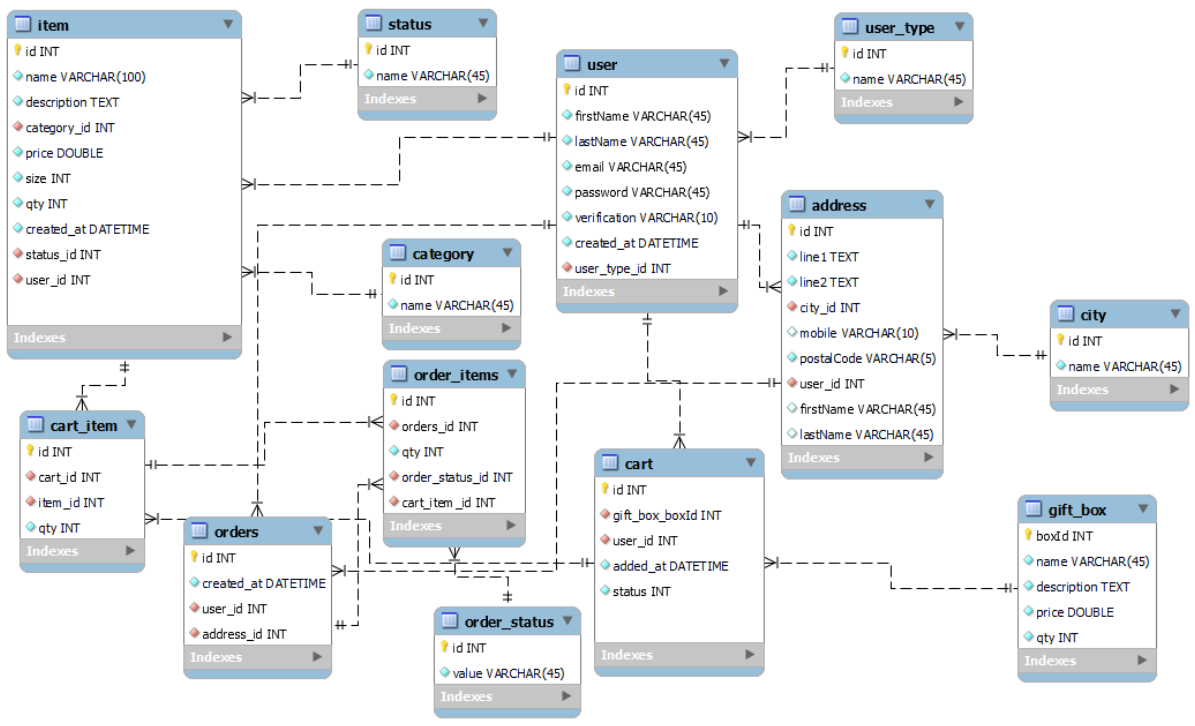


ii. **Use Case Diagram**



II. Database Design

i. ER Diagram



ii. Database Schema (generated via Hibernate mapping)

➢ Address Entity Class

```
@Entity

@Table(name = "address")

public class Address implements Serializable {


  @Id
```

```
@GeneratedValue(strategy = GenerationType.IDENTITY)
@Column(name = "id")
private int id;

@Column(name = "line1", nullable = false)
private String line1;

@Column(name = "line2", nullable = false)
private String line2;

@ManyToOne
@JoinColumn(name = "city_id")
private City city;

@Column(name = "mobile", length = 10, nullable = true)
private String mobile;

@Column(name = "postalCode", nullable = false)
private String postalCode;

@ManyToOne
@JoinColumn(name = "user_id")
private User user;

@Column(name = "firstName", length = 45, nullable = true)
private String firstName;

@Column(name = "lastName", length = 45, nullable = true)
private String lastName;
```

```
public Address() {
}

public int getId() {
   return id;
}

public void setId(int id) {
   this.id = id;
}

public String getLine1() {
   return line1;
}

public void setLine1(String line1) {
   this.line1 = line1;
}

public String getLine2() {
   return line2;
}

public void setLine2(String line2) {
   this.line2 = line2;
}

public City getCity() {
    return city;
}

public void setCity(City city) {
    this.city = city;
}

public String getPostalCode() {
    return postalCode;
}

public void setPostalCode(String postalCode) {
    this.postalCode = postalCode;
}

public User getUser() {
    return user;
}

public void setUser(User user) {
    this.user = user;
}

public String getMobile() {
    return mobile;
}

public void setMobile(String mobile) {
    this.mobile = mobile;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

}
```

- Cart Entity class

```
@Entity
@Table(name = "cart")
public class Cart implements Serializable{

   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   @Column(name = "id")
   private int id;

   @ManyToOne
   @JoinColumn(name = "gift_box_boxId")
   private GiftBox giftBox;

   @ManyToOne
   @JoinColumn(name = "user_id")
   private User user;

   @Column(name = "added_at")
   private Date added_at;

   @Column(name = "status",nullable = false)
   private int status;

   public Cart() {
   }
```

```
public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public Date getAdded_at() {
    return added_at;
}

public void setAdded_at(Date added_at) {
    this.added_at = added_at;
}

public User getUser() {
    return user;
}

public void setUser(User user) {
    this.user = user;
}

public GiftBox getGiftBox() {
    return giftBox;
}
public void setGiftBox(GiftBox giftBox) {
        this.giftBox = giftBox;
    }

    public int getStatus() {
        return status;
    }

    public void setStatus(int status) {
        this.status = status;
    }

}
```

➢ CartItem Entity Class

```
@Entity
@Table(name = "cart_item")
public class CartItem {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;

    @ManyToOne
    @JoinColumn(name = "cart_id")
    private Cart cart;
```

```
@ManyToOne
@JoinColumn(name = "item_id")
private Item item;

@Column(name = "qty")
private int qty;

public CartItem() {
}

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public Cart getCart() {
    return cart;
}

public void setCart(Cart cart) {
    this.cart = cart;
}

public Item getItem() {
    return item;
}
```

```
    public void setItem(Item item) {
        this.item = item;
    }

    public int getQty() {
        return qty;
    }

    public void setQty(int qty) {
        this.qty = qty;
    }

}
```

➢ Category Entity Class

```
@Entity
@Table(name = "category")
public class Category implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;

    @Column(name = "name", length = 45, nullable = false)
    private String name;


    public Category() {
    }


    public int getId() {
        return id;
    }


    public void setId(int id) {
        this.id = id;
    }


    public String getName() {
        return name;
    }


    public void setName(String name) {
        this.name = name;
    }

}
```

- City Entity Class

```
@Entity
@Table(name = "city")
public class City  implements Serializable{

   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   @Column(name = "id")
   private int id;

   @Column(name = "name", length = 45, nullable = false)
   private String name;

   public City() {
   }



   public int getId() {
      return id;
   }

   public void setId(int id) {
      this.id = id;
   }
```

```
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

}
```

- Gift Box Entity Class

```
@Entity
@Table(name = "gift_box")
public class GiftBox implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "boxId")
    private int boxId;

    @Column(name = "name", length = 100, nullable = false)
```

```
private String name;

@Column(name = "description", nullable = false)
private String description;

@Column(name = "price", nullable = false)
private double price;

@Column(name = "qty", nullable = false)
private int qty;

public GiftBox() {
}

public int getBoxId() {
   return boxId;
}

public void setBoxId(int boxId) {
   this.boxId = boxId;
}

public String getName() {
   return name;
}
```

```
public void setName(String name) {
    this.name = name;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public double getPrice() {
    return price;
}

public void setPrice(double price) {
    this.price = price;
}

public int getQty() {
    return qty;
}

public void setQty(int qty) {
    this.qty = qty;
    }

}
```

- Item Entity Class

```
@Entity
@Table(name = "item")
public class Item implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;

    @Column(name = "name", length = 45, nullable = false)
    private String name;

    @Column(name = "description", nullable = false)
    private String description;

    @ManyToOne
    @JoinColumn(name = "category_id")
    private Category category;
```

```
@Column(name = "price", nullable = false)
private double price;

@Column(name = "size", length = 45, nullable = false)
private int size;

@Column(name = "qty")
private int qty;

@Column(name = "created_at")
private Date created_at;

@ManyToOne
@JoinColumn(name = "status_id")
private Status status;

@ManyToOne
@JoinColumn(name = "user_id")
private User user;

public Item() {
}

public int getId() {
   return id;
}
```

```
public void setId(int id) {
   this.id = id;
}

public String getName() {
   return name;
}

public void setName(String name) {
   this.name = name;
}

public String getDescription() {
   return description;
}

public void setDescription(String description) {
   this.description = description;
}

public Category getCategory() {
   return category;
}

public void setCategory(Category category) {
    this.category = category;
}

public double getPrice() {
    return price;
}

public void setPrice(double price) {
    this.price = price;
}

public int getSize() {
    return size;
}

public void setSize(int size) {
    this.size = size;
}

public int getQty() {
    return qty;
}

public void setQty(int qty) {
    this.qty = qty;
}
```

```
    public Date getCreated_at() {
        return created_at;
    }

    public void setCreated_at(Date created_at) {
        this.created_at = created_at;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public Status getStatus() {
        return status;
    }

    public void setStatus(Status status) {
        this.status = status;
    }

}
```

➢ Orders Entity Class

```
@Entity
@Table(name = "orders")
public class Orders implements Serializable{

   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   @Column(name="id")
   private int id;

    @Column(name="created_at")
   private Date created_at;

   @ManyToOne
   @JoinColumn(name="user_id")
   private User user;

   @ManyToOne
   @JoinColumn(name="address_id")
   private Address address;

   public Orders() {
   }
```

```
public int getId() {
   return id;
}

public void setId(int id) {
   this.id = id;
}

public Date getCreated_at() {
   return created_at;
}

public void setCreated_at(Date created_at) {
   this.created_at = created_at;
}

public User getUser() {
   return user;
}

public void setUser(User user) {
   this.user = user;
}

public Address getAddress() {
        return address;
    }

    public void setAddress(Address address) {
        this.address = address;
    }

}
```

➢ OrderItems Entity Class

```
@Entity
@Table(name = "order_items")
public class OrderItems implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;

    @Column(name = "qty", nullable = false)
    private int qty;

    @ManyToOne
```

```
@JoinColumn(name = "cart_item_id")
private CartItem cartItem;

@ManyToOne
@JoinColumn(name = "order_status_id")
private OrderStatus orderStatus;

@ManyToOne
@JoinColumn(name = "orders_id")
private Orders orders;

public OrderItems() {
}

public int getId() {
   return id;
}

public void setId(int id) {
   this.id = id;
}

public int getQty() {
   return qty;
}
```

```
public void setQty(int qty) {
   this.qty = qty;
}

public OrderStatus getOrderStatus() {
   return orderStatus;
}

public void setOrderStatus(OrderStatus orderStatus) {
   this.orderStatus = orderStatus;
}

public Orders getOrders() {
   return orders;
}

public void setOrders(Orders orders) {
   this.orders = orders;
}

public CartItem getCartItem() {
   return cartItem;
}

public void setCartItem(CartItem cartItem) {
   this.cartItem = cartItem;
```

```
    }

}
```

➢ OrderStatus Entity Class

```
@Entity
@Table(name = "order_status")
public class OrderStatus  implements Serializable{

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id")
    private int id;

    @Column(name="value",length = 45,nullable = false)
    private String value;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getValue() {
        return value;
    }

    public void setValue(String value) {
        this.value = value;
    }

}
```

- Status Entity Class

```
@Entity
@Table(name = "status")
public class Status implements Serializable{

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;

    @Column(name = "name", length = 45, nullable = false)
    private String name;


    public Status() {
    }


    public int getId() {
        return id;
    }



    public void setId(int id) {
        this.id = id;
    }



    public String getName() {
        return name;
    }


    public void setName(String name) {
        this.name = name;
    }

}
```

- User Entity Class

```
@Entity
@Table(name = "user")
public class User implements Serializable {

   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   @Column(name = "id")
   private int id;

   @Column(name = "firstName", length = 45, nullable = false)
   private String firstName;

   @Column(name = "lastName", length = 45, nullable = false)
   private String lastName;

   @Column(name = "email", length = 45, nullable = false)
   private String email;

   @Column(name = "password", length = 45, nullable = false)
   private String password;



   @Column(name = "verification", length = 10, nullable = false)
```

```
private String verification;

@Column(name = "created_at", nullable = false)
private Date created_at;

@ManyToOne
@JoinColumn(name = "user_type_id")
private UserType userType;

public User() {
}

public int getId() {
   return id;
}

public void setId(int id) {
   this.id = id;
}

public String getFirstName() {
   return firstName;
}

public void setFirstName(String firstName) {
   this.firstName = firstName;
}

public String getLastName() {
   return lastName;
}

public void setLastName(String lastName) {
   this.lastName = lastName;
}

public String getEmail() {
   return email;
}

public void setEmail(String email) {
   this.email = email;
}

public String getPassword() {
   return password;
}

public void setPassword(String password) {
   this.password = password;
}
```

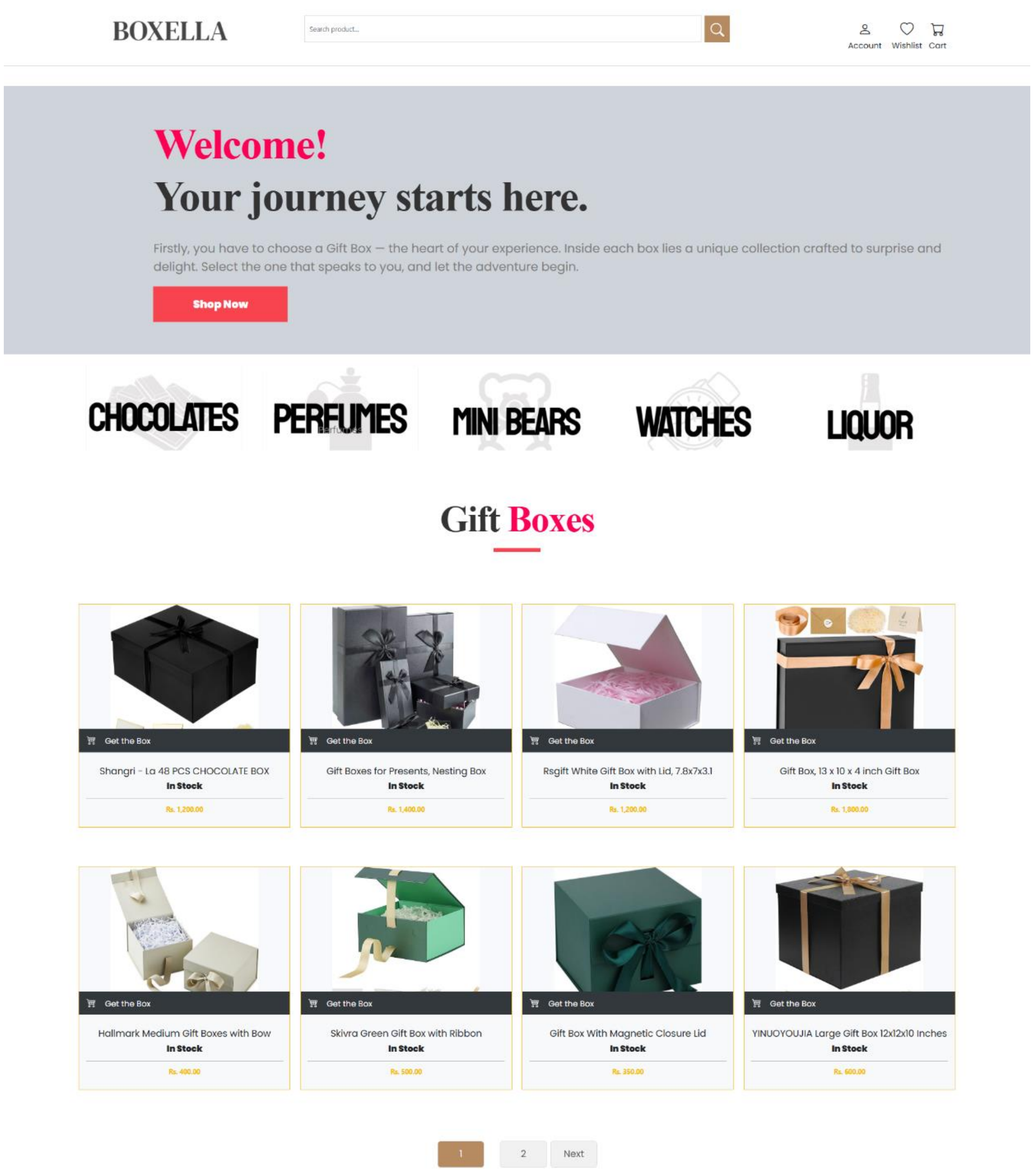
```
    }

    public String getVerification() {
        return verification;
    }

    public void setVerification(String verification) {
        this.verification = verification;
    }

    public Date getCreated_at() {
        return created_at;
    }

    public void setCreated_at(Date created_at) {
        this.created_at = created_at;
    }

    public UserType getUserType() {
        return userType;
    }

    public void setUserType(UserType userType) {
        this.userType = userType;
    }

}
```

➢ UserType Entity Class

```
@Entity
@Table(name = "user_type")
public class UserType implements Serializable{

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;

    @Column(name = "name", length = 45, nullable = false)
    private String name;

    public UserType() {
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

III. Storyboards (Wireframes)

01. Index.html



BOXELLA

Search product...

Account Wishlist Cart

Welcome!

Your journey starts here.

Firstly, you have to choose a Gift Box — the heart of your experience. Inside each box lies a unique collection crafted to surprise and delight. Select the one that speaks to you, and let the adventure begin.

Shop Now

CHOCOLATES PERFUMES MINI BEARS WATCHES LIQUOR

Gift Boxes

Get the Box
Shangri - La 48 PCS CHOCOLATE BOX
In Stock
Rs. 1,200.00

Get the Box
Gift Boxes for Presents, Nesting Box
In Stock
Rs. 1,400.00

Get the Box
Rsgift White Gift Box with Lid, 7.8x7x3.1
In Stock
Rs. 1,200.00

Get the Box
Gift Box, 13 x 10 x 4 inch Gift Box
In Stock
Rs. 1,800.00

Get the Box
Hallmark Medium Gift Boxes with Bow
In Stock
Rs. 400.00

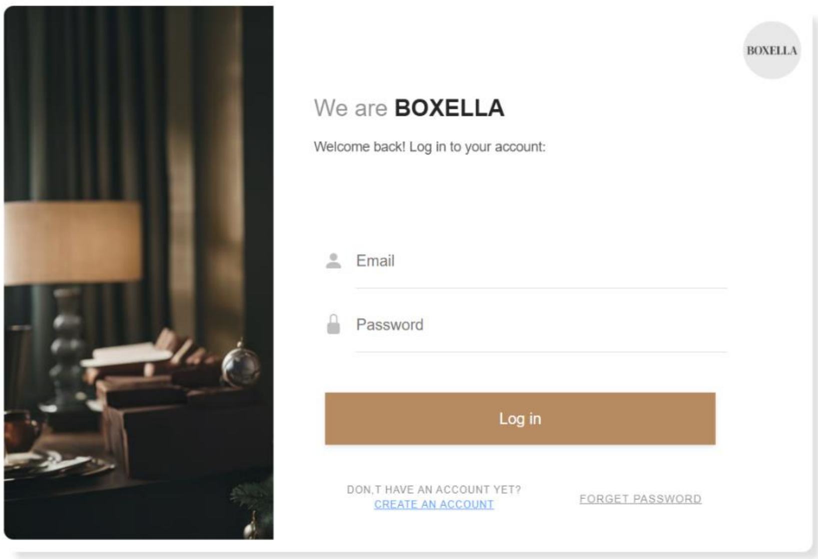
Get the Box
Skivra Green Gift Box with Ribbon
In Stock
Rs. 500.00

Get the Box
Gift Box With Magnetic Closure Lid
In Stock
Rs. 350.00

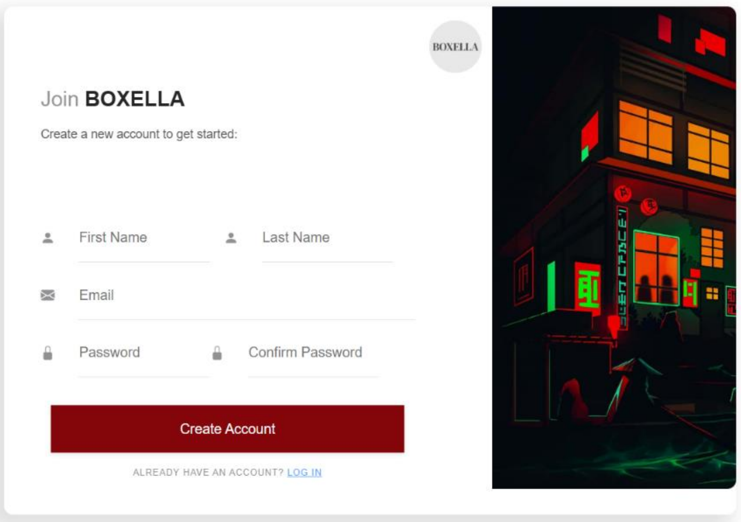
Get the Box
YINUOYOUJIA Large Gift Box 12x12x10 Inches
In Stock
Rs. 600.00

1 2 Next

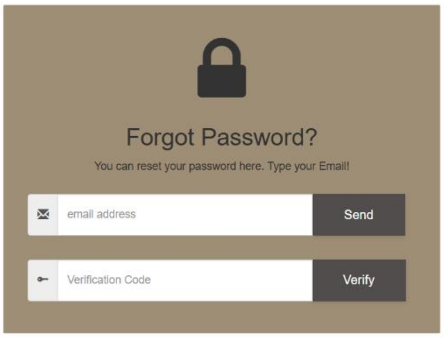
02. signIn.html



03. signUp.html

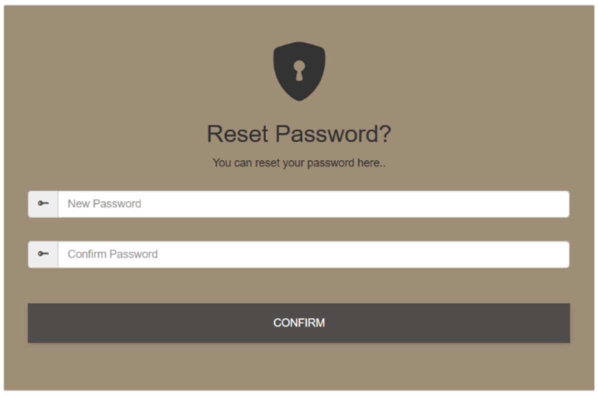
<


04. forgetPassword.html


Forgot Password?

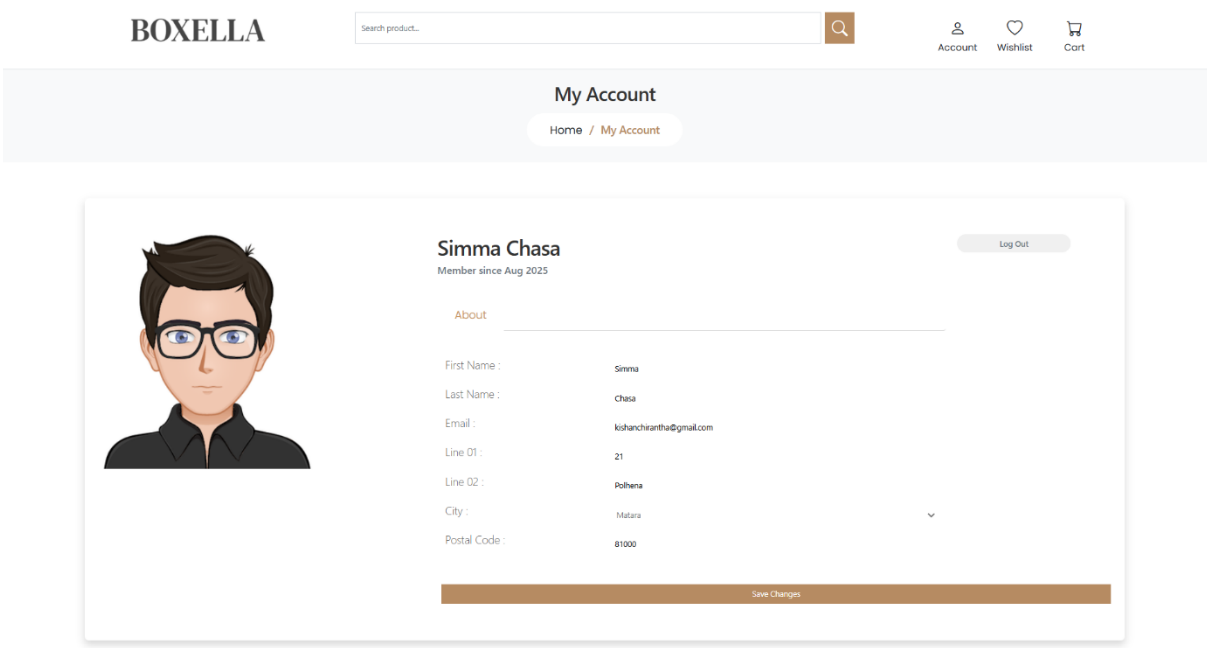
You can reset your password here. Type your Email!

05. resetPassword.html


Reset Password?

You can reset your password here..

06. account.html


BOXELLA

Search product...

Account Wishlist Cart

My Account

Home / My Account

Simma Chasa

Member since Aug 2025

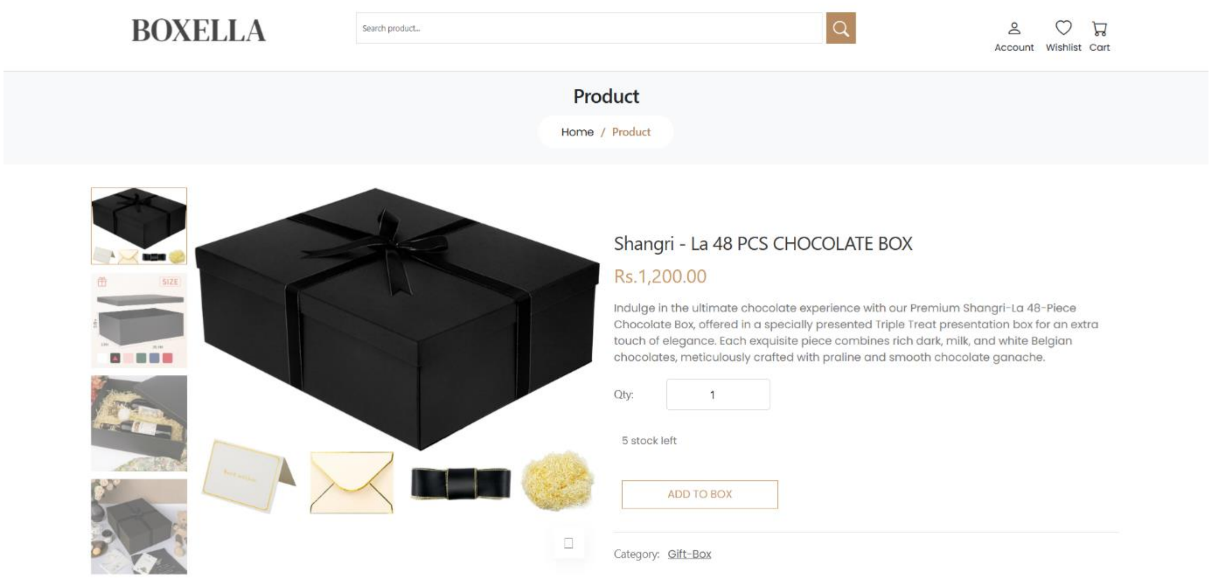
Log Out

About

First Name :	Simma
Last Name :	Chasa
Email :	kishanchirantha@gmail.com
Line 01 :	21
Line 02 :	Polhena
City :	Matara
Postal Code :	81000

Save Changes

07. singleProduct.html


BOXELLA

Search product...

Account Wishlist Cart

Product

Home / Product

Shangri - La 48 PCS CHOCOLATE BOX

Rs.1,200.00

Indulge in the ultimate chocolate experience with our Premium Shangri-La 48-Piece Chocolate Box, offered in a specially presented Triple Treat presentation box for an extra touch of elegance. Each exquisite piece combines rich dark, milk, and white Belgian chocolates, meticulously crafted with praline and smooth chocolate ganache.

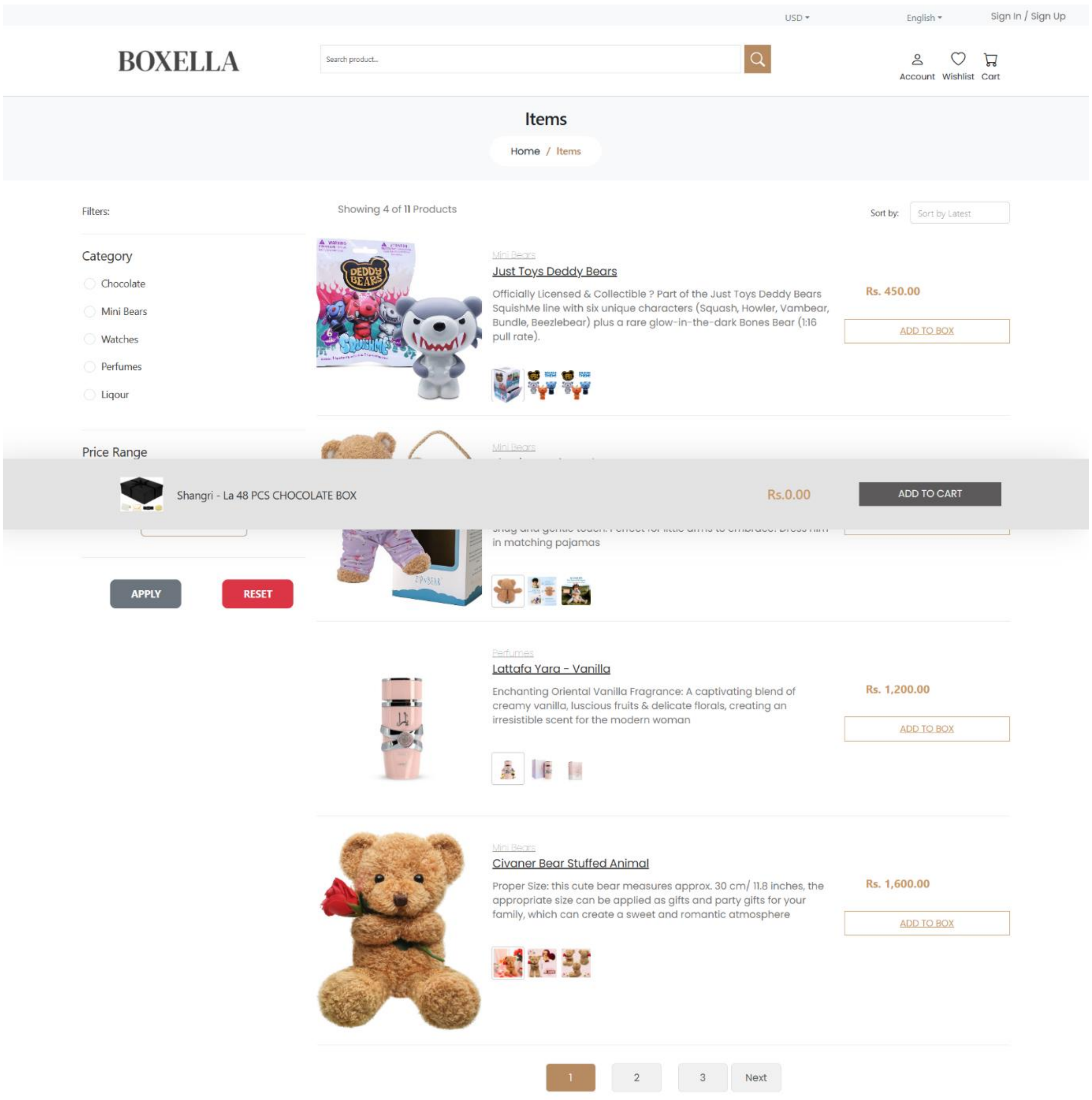
Qty: 1

5 stock left

ADD TO BOX

Category: Gift-Box

08. Items.html



USD ▾ English ▾ Sign In / Sign Up

BOXELLA

Search product...

Account Wishlist Cart

Items

Home / Items

Filters:

Showing 4 of 11 Products

Sort by: Sort by Latest

Category

- Chocolate
- Mini Bears
- Watches
- Perfumes
- Liqour

Price Range

APPLY RESET

Mini Bears

Just Toys Deddy Bears

Officially Licensed & Collectible ? Part of the Just Toys Deddy Bears SquishMe line with six unique characters (Squash, Howler, Vambear, Bundle, Beezlebear) plus a rare glow-in-the-dark Bones Bear (1:16 pull rate).

Rs. 450.00

ADD TO BOX

Mini Bears

in matching pajamas

Shangri - La 48 PCS CHOCOLATE BOX Rs.0.00 ADD TO CART

Perfumes

Lattafa Yara - Vanilla

Enchanting Oriental Vanilla Fragrance: A captivating blend of creamy vanilla, luscious fruits & delicate florals, creating an irresistible scent for the modern woman

Rs. 1,200.00

ADD TO BOX

Mini Bears

Civaner Bear Stuffed Animal

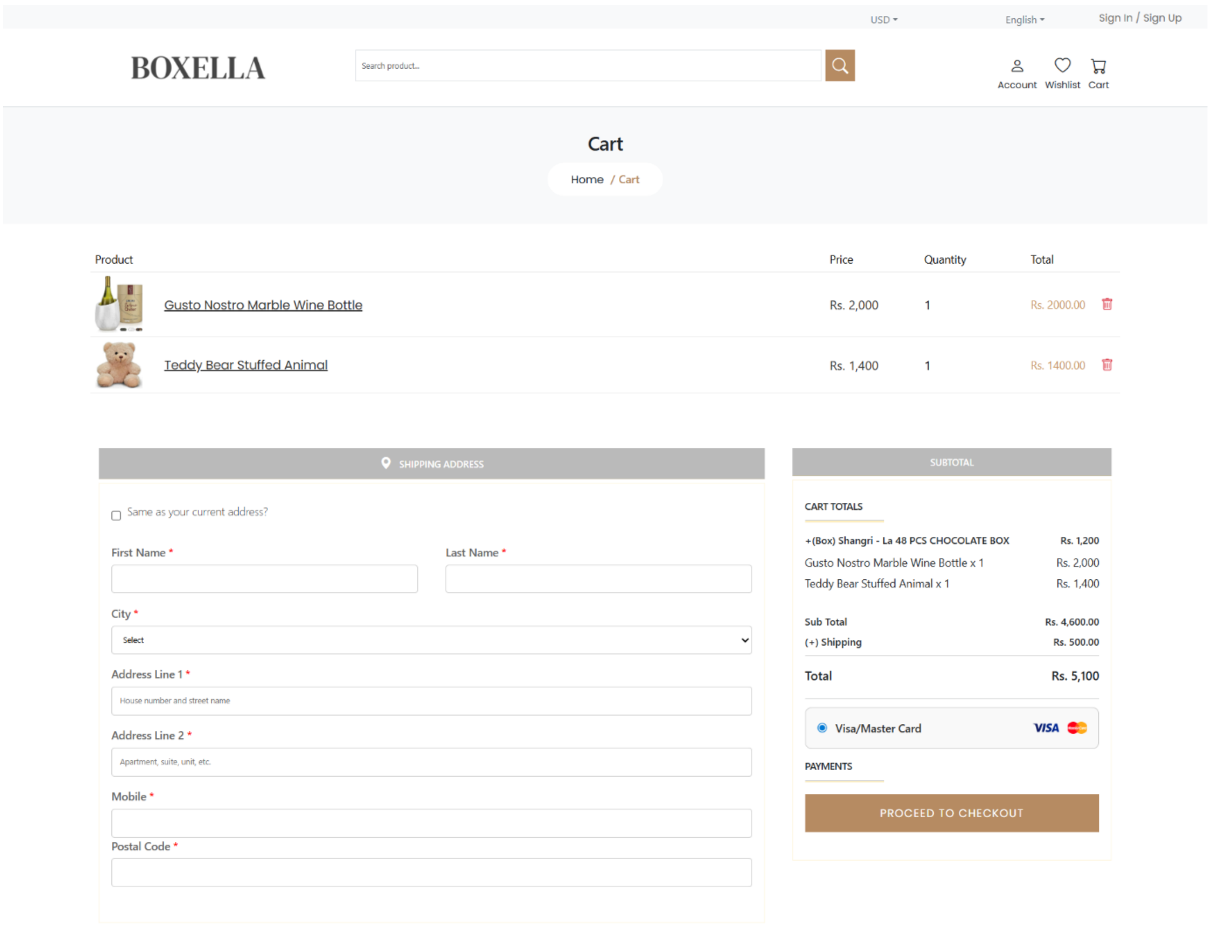
Proper Size: this cute bear measures approx. 30 cm/ 11.8 inches, the appropriate size can be applied as gifts and party gifts for your family, which can create a sweet and romantic atmosphere

Rs. 1,600.00

ADD TO BOX

1 2 3 Next

09. cart.html


USD ▾ English ▾ Sign In / Sign Up

BOXELLA

Search product...

Account Wishlist Cart

Cart

Home / Cart

Product	Price	Quantity	Total
Gusto Nostro Marble Wine Bottle	Rs. 2,000	1	Rs. 2000.00
Teddy Bear Stuffed Animal	Rs. 1,400	1	Rs. 1400.00

SHIPPING ADDRESS

Same as your current address?

First Name *

Last Name *

City *

Select

Address Line 1 *

House number and street name

Address Line 2 *

Apartment, suite, unit, etc.

Mobile *

Postal Code *

SUBTOTAL

CART TOTALS

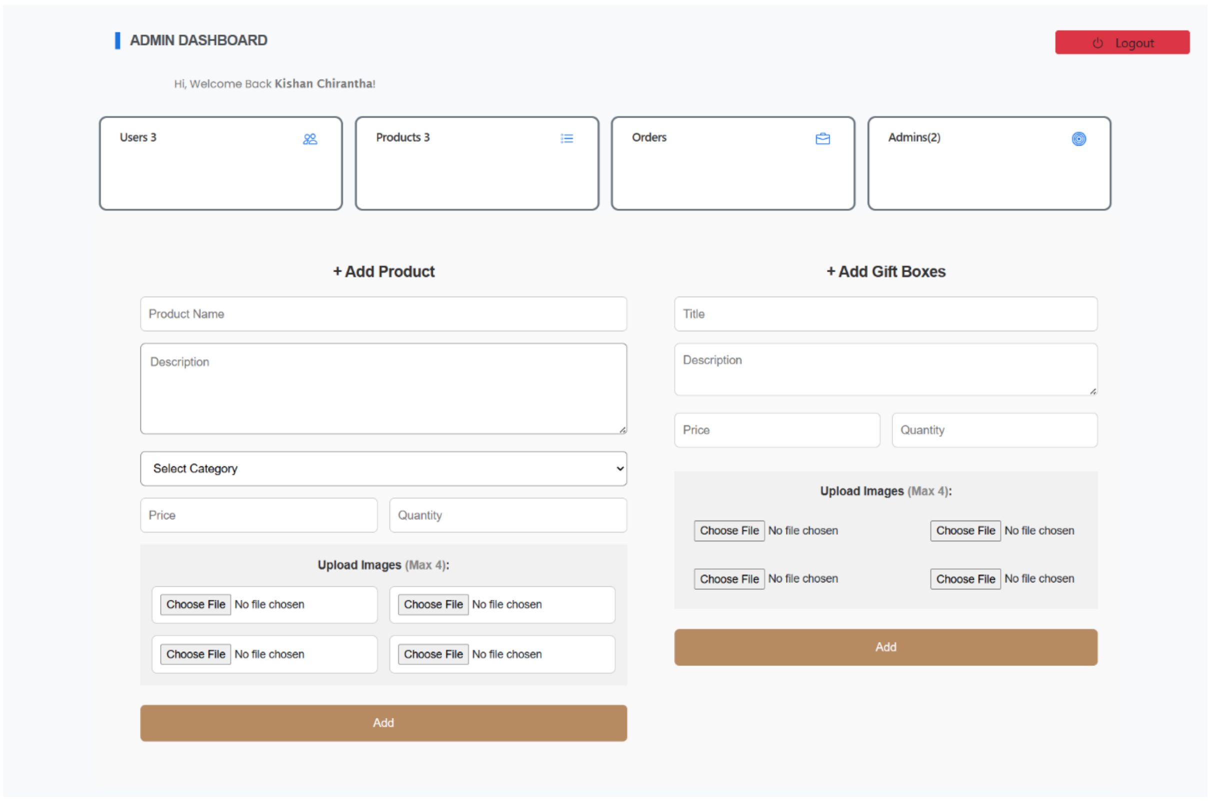
+(Box) Shangri - La 48 PCS CHOCOLATE BOX	Rs. 1,200
Gusto Nostro Marble Wine Bottle x 1	Rs. 2,000
Teddy Bear Stuffed Animal x 1	Rs. 1,400
Sub Total	Rs. 4,600.00
(+) Shipping	Rs. 500.00
Total	Rs. 5,100

● Visa/Master Card VISA

PAYMENTS

PROCEED TO CHECKOUT

10. adminDashboard.html


ADMIN DASHBOARD

Logout

Hi, Welcome Back Kishan Chirantha!

Users 3

Products 3

Orders

Admins(2)

+ Add Product

Product Name

Description

Select Category

Price

Quantity

Upload Images (Max 4):

Choose File No file chosen

Choose File No file chosen

Choose File No file chosen

Choose File No file chosen

Add

+ Add Gift Boxes

Title

Description

Price

Quantity

Upload Images (Max 4):

Choose File No file chosen

Choose File No file chosen

Choose File No file chosen

Choose File No file chosen

Add

9. Project experience and learning points

The current version of the gift box ordering web application, while functional and user-friendly, has some limitations. It requires a stable internet connection and relies on third-party payment gateways, which may introduce transaction fees or regional restrictions. Delivery tracking is not built into the platform, requiring customers to rely on external courier updates, and the system currently supports only one language and currency, limiting its reach to international users. The selection of products and gift boxes is restricted to those added by the administrator, with no option for customers to request completely custom items.

In future updates, the platform could be enhanced by adding real-time delivery tracking, multi-language and multi-currency support, product recommendations, and expanded customization options such as gift-wrapping styles and personalized messages. Additional features like loyalty rewards, social media integration, and advanced analytics for administrators could further improve customer engagement and business growth.

10. Limitations and future enhancements

- Performing comprehensive business research to ensure the system aligned with actual market requirements.
- Applying Hibernate ORM for handling database interactions, eliminating the need for raw SQL queries.
- Gaining awareness of the challenges involved in transitioning traditional businesses to digital platforms.

11. Conclusion

The gift box ordering web application provides a simple, structured, and customizable way for customers to create complete gift packages online while giving administrators an efficient platform to manage products and orders. It improves convenience, enhances the gifting experience, and supports business growth, with potential for further enhancements like delivery tracking, multi-language support, and expanded customization features.

12. Appendices

I. Appendix D – Hibernate CRUD Operations

01. AddGiftBox Servlet

```
  @Override
  protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

      String boxName = request.getParameter("boxName");
      String boxDesc = request.getParameter("boxDesc");
      String boxprice = request.getParameter("boxprice");
      String boxQty = request.getParameter("boxQty");

      System.out.println(boxName);
      System.out.println(boxDesc);
      System.out.println(boxprice);
      System.out.println();

      JsonObject responseObject = new JsonObject();
```

```
responseObject.addProperty("status", false);

if (boxName.isEmpty()) {
   responseObject.addProperty("message", "Please Enter your Box Name first!");
} else if (boxDesc.isEmpty()) {
   responseObject.addProperty("message", "Please Enter Description!");
} else if (boxprice.isEmpty()) {
   responseObject.addProperty("message", "Please Enter Price!");
} else if (!Util.isDouble(boxprice)) {
   responseObject.addProperty("message", "Invaid Price!");
} else if (boxQty.isEmpty()) {
   responseObject.addProperty("message", "Please Enter Price!");
} else if (!Util.isInteger(boxQty)) {
   responseObject.addProperty("message", "Invaid Quantity!");
} else {

   SessionFactory sf = HibernateUtil.getSessionFactory();
   Session s = sf.openSession();

   GiftBox giftBox1 = new GiftBox();
   giftBox1.setName(boxName);
   giftBox1.setDescription(boxDesc);
   giftBox1.setPrice(Double.parseDouble(boxprice));
   giftBox1.setQty(Integer.parseInt(boxQty));

   int id = (int) s.save(giftBox1);
   s.beginTransaction().commit();

   Part part1 = request.getPart("Boximg1");
```

```
        Part part2 = request.getPart("Boximg2");
        Part part3 = request.getPart("Boximg3");
        Part part4 = request.getPart("Boximg4");

        String appPath = getServletContext().getRealPath("");

        String newPath = appPath.replace("build" + File.separator + "web", "web" +
File.separator + "assets" + File.separator + "product-images" + File.separator + "giftBox-
images");

        File productFolder = new File(newPath, String.valueOf(id));
        productFolder.mkdir();

        File file1 = new File(productFolder, "image1.jpeg");
        Files.copy(part1.getInputStream(), file1.toPath(),
StandardCopyOption.REPLACE_EXISTING);

        File file2 = new File(productFolder, "image2.jpeg");
        Files.copy(part2.getInputStream(), file2.toPath(),
StandardCopyOption.REPLACE_EXISTING);

        File file3 = new File(productFolder, "image3.jpeg");
        Files.copy(part3.getInputStream(), file3.toPath(),
StandardCopyOption.REPLACE_EXISTING);

        File file4 = new File(productFolder, "image4.jpeg");
        Files.copy(part4.getInputStream(), file4.toPath(),
StandardCopyOption.REPLACE_EXISTING);

        responseObject.addProperty("status", true);
        responseObject.addProperty("message", "Gift Box Added Succefully");
```

```
            s.close();
        }

        Gson gson = new Gson();
        String responseText = gson.toJson(responseObject);
        response.setContentType("application/json");
        response.getWriter().write(responseText);

    }
```

02. AddtoBox Servlet

```
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

        String prId = request.getParameter("id");
        String qty = request.getParameter("qty");
        String boxId = request.getParameter("boxId");
        String itemPrice = request.getParameter("itemPrice");
        String size = request.getParameter("size");

        Gson gson = new Gson();
        JsonObject responseObject = new JsonObject();
        responseObject.addProperty("status", false);

        if (!Util.isInteger(prId)) {
```

```
        responseObject.addProperty("message", "Invalid product Id");
    } else if (!Util.isInteger(qty)) {
        responseObject.addProperty("message", "Invalid product Quantity");
    } else {

        //add to cart start
        SessionFactory sf = HibernateUtil.getSessionFactory();
        Session s = sf.openSession();
        Transaction tr = s.beginTransaction();

        GiftBox giftBox = (GiftBox) s.get(GiftBox.class, Integer.valueOf(boxId));

        if (giftBox == null) {
            responseObject.addProperty("message", "Gift Box is not found!");
        } else {
            Item item = (Item) s.get(Item.class, Integer.valueOf(prId));
            if (item == null) {
                responseObject.addProperty("message", "Product is not found!");
            } else {

                // Calculate total amount for this product
                double totalAmount = Integer.parseInt(qty) * Double.parseDouble(itemPrice);
                System.out.println(totalAmount);
                responseObject.addProperty("totalAmount", totalAmount);

                User user = (User) request.getSession().getAttribute("user");

                if (user != null) { // add to db cart
```

```
Criteria c1 = s.createCriteria(Cart.class);
c1.add(Restrictions.eq("user", user));
c1.add(Restrictions.eq("giftBox", giftBox));

if (c1.list().isEmpty()) {

    Cart cart = new Cart();
    cart.setGiftBox(giftBox);
    cart.setUser(user);
    cart.setStatus(1);
    cart.setAdded_at(new Date());

    CartItem cartItem = new CartItem();
    cartItem.setCart(cart);
    cartItem.setItem(item);
    cartItem.setQty(Integer.parseInt(qty));

    s.save(cart);
    s.save(cartItem);
    tr.commit();

    responseObject.addProperty("status", true);
    responseObject.addProperty("message", "Product Added to the Box");

} else {

    Cart cart1 = (Cart) c1.list().get(0);
    Criteria c2 = s.createCriteria(CartItem.class);
    c2.add(Restrictions.eq("cart", cart1));
```

```
if (c2.list().isEmpty()) {

    CartItem cartItem = new CartItem();
    cartItem.setCart(cart1);
    cartItem.setItem(item);
    cartItem.setQty(Integer.parseInt(qty));

    s.save(cartItem);
    tr.commit();

    responseObject.addProperty("status", true);
    responseObject.addProperty("message", "Cart Updated");

} else {
    if (Integer.parseInt(qty) <= item.getQty()) { // product quantity available

        Criteria c3 = s.createCriteria(CartItem.class);
        c3.add(Restrictions.eq("item", item));

        if (c3.list().isEmpty()) {

            CartItem cartItem = new CartItem();
            cartItem.setCart(cart1);
            cartItem.setItem(item);
            cartItem.setQty(Integer.parseInt(qty));

            s.save(cartItem);
            tr.commit();
```

```
                        responseObject.addProperty("status", true);
                        responseObject.addProperty("message", "Product Added to the
Box");

                    } else {

                        CartItem cartItem = (CartItem) c3.list().get(0);

                        int newQty = cartItem.getQty() + Integer.parseInt(qty);
                        if (newQty <= item.getQty()) {

                            cartItem.setQty(newQty);
                            s.update(cartItem);
                            tr.commit();

                            responseObject.addProperty("status", true);
                            responseObject.addProperty("message", "Cart Updated");

                        } else {
                            responseObject.addProperty("message", "Insufficient Product
Quantity!");
                        }
                    }
                    responseObject.addProperty("status", true);
                    responseObject.addProperty("message", "Product Added to the Box");
                } else {
                    responseObject.addProperty("message", "Insufficient Product
Quantity!");
                }
```

```
        }

    }

} else {// add to session cart

    HttpSession ses = request.getSession();

    if (ses.getAttribute("sessionBox") == null) {

        if (Integer.parseInt(qty) <= item.getQty()) {

            ArrayList<CartItem> sesItemBox = new ArrayList<>();
            CartItem cartItem = new CartItem();
            cartItem.setQty(Integer.parseInt(qty));
            cartItem.setItem(item);
            sesItemBox.add(cartItem);

            ArrayList<Cart> sesCart = new ArrayList<>();
            Cart cart = new Cart();
            cart.setGiftBox(giftBox);
            cart.setStatus(1);
            cart.setUser(null);
            sesCart.add(cart);

            ses.setAttribute("sesItemBox", sesItemBox);
            ses.setAttribute("sessionCart", sesCart);

            responseObject.addProperty("status", true);
```

```
                        responseObject.addProperty("message", "product added to the box");
                    } else {
                        responseObject.addProperty("message", "Insufficient Product Quantity!");
                    }
                } else {
                    ArrayList<CartItem> sessList = new ArrayList<>();
                    CartItem foundedItemBox = null;
                    Cart foundedCart = null;

                    for (CartItem cartItem : sessList) {
                        if (cartItem.getItem().getId() == item.getId()) {
                            foundedItemBox = cartItem;
                            break;
                        }
                    }

                    if (foundedItemBox != null) {
                        int newQty = foundedItemBox.getQty() + Integer.parseInt(qty);
                        if (newQty <= item.getQty()) {

                            foundedItemBox.setQty(newQty);
                            responseObject.addProperty("status", true);
                            responseObject.addProperty("message", "Box Upated");

                            totalAmount = newQty * Double.parseDouble(itemPrice);
                            responseObject.addProperty("totalAmount", totalAmount);

                        } else {
                            responseObject.addProperty("message", "Insufficient Product
Quantity!");
```

```
                        }


                    } else {
                        foundedItemBox = new CartItem();
                        foundedCart = new Cart();


                        if (Integer.parseInt(qty) <= item.getQty()) {


                            foundedItemBox.setQty(Integer.parseInt(qty));
                            foundedCart.setUser(null);
                            foundedItemBox.setItem(item);
                            foundedCart.setGiftBox(giftBox);
                            sessList.add(foundedItemBox);
                            responseObject.addProperty("status", true);
                            responseObject.addProperty("message", "product added to the box");
                        } else {
                            responseObject.addProperty("message", "Insufficient Product
Quantity!");
                        }

                    }
                }


            }
        }
    }


}
responseObject.addProperty("status", true);
```

```
        //add to cart end
        String toJson = gson.toJson(responseObject);

        response.setContentType(
                "application/json");
        response.getWriter()
                .write(toJson);
    }
```

03. Checkout Servlet

```
    private static final int SELECTOR_DEFAULT_VALUE = 0;
    private static final int ORDER_PENDING = 5;
//    private static final int ORDER_PENDING = 5;

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

        Gson gson = new Gson();
        JsonObject resJsonObject = gson.fromJson(request.getReader(), JsonObject.class);

        boolean isCurrentAddress = resJsonObject.get("isCurrentAddress").getAsBoolean();
        String firstName = resJsonObject.get("firstName").getAsString();
        String lastName = resJsonObject.get("lastName").getAsString();
        String citySelect = resJsonObject.get("citySelect").getAsString();
        String lineOne = resJsonObject.get("lineOne").getAsString();
        String lineTwo = resJsonObject.get("lineTwo").getAsString();
        String mobile = resJsonObject.get("mobile").getAsString();
        String postalCode = resJsonObject.get("postalCode").getAsString();
        String amount = resJsonObject.get("amount").getAsString();
        String boxId = resJsonObject.get("boxId").getAsString();

        JsonObject responseObject = new JsonObject();
        responseObject.addProperty("status", false);

        SessionFactory sf = HibernateUtil.getSessionFactory();
        Session s = sf.openSession();
        Transaction tr = s.beginTransaction();

        User user = (User) request.getSession().getAttribute("user");
        if (user == null) {
          responseObject.addProperty("message", "Session Expired. Please sign in again");
        } else {
          if (isCurrentAddress) {
            Criteria c1 = s.createCriteria(Address.class);
            c1.add(Restrictions.eq("user", user));
            c1.addOrder(Order.desc("id"));
            if (c1.list().isEmpty()) {
              responseObject.addProperty("message", "Yoy current address is not found.
Please add a new Address");
            } else {
              Address address = (Address) c1.list().get(0);
              processCheckout(s, tr, user, address, responseObject, amount, boxId);
            }
          } else {

            if (firstName.isEmpty()) {
```

```
    responseObject.addProperty("message", "Please Enter your First Name first!");
} else if (lastName.isEmpty()) {
    responseObject.addProperty("message", "Please Enter your Last Name!");
} else if (!Util.isInteger(citySelect)) {
    responseObject.addProperty("message", "Invalid city!");
} else if (Integer.parseInt(citySelect) == Checkout.SELECTOR_DEFAULT_VALUE) {
    responseObject.addProperty("message", "Please Select City!");
} else {
    City city = (City) s.get(City.class, Integer.valueOf(citySelect));
    if (city == null) {
        responseObject.addProperty("message", "Invalid City name!");
    } else {
        if (lineOne.isEmpty()) {
            responseObject.addProperty("message", "Please Enter your Line 01!");
        } else if (lineTwo.isEmpty()) {
            responseObject.addProperty("message", "Please Enter your Line 02!");
        } else if (postalCode.isEmpty()) {
            responseObject.addProperty("message", "Please Enter Postal code!");
        } else if (!Util.isPostalCodeValid(postalCode)) {
            responseObject.addProperty("message", "Invalid Postal Code!");
        } else if (mobile.isEmpty()) {
            responseObject.addProperty("message", "Please Enter Mobile Number!");
        } else if (!Util.isValidMobile(mobile)) {
            responseObject.addProperty("message", "Invalid Mobile Number!");
        } else {

            Address address = new Address();
            address.setLine1(lineOne);
            address.setLine2(lineTwo);
```

```
                    address.setCity(city);
                    address.setMobile(mobile);
                    address.setPostalCode(postalCode);
                    address.setUser(user);
                    address.setFirstName(firstName);
                    address.setLastName(lastName);
                    s.save(address);

                    processCheckout(s, tr, user, address, responseObject, amount, boxId);

                }

            }

        }

    }

}

response.setContentType("application/json");
String toJson = gson.toJson(responseObject);
response.getWriter().write(toJson);

}

private void processCheckout(Session s, Transaction tr, User user, Address address,
JsonObject responseObject, String amount, String boxId) {

    try {
    Orders orders = new Orders();
    orders.setAddress(address);
    orders.setCreated_at(new Date());
    orders.setUser(user);

    int orderId = (int) s.save(orders);

    GiftBox box = (GiftBox) s.get(GiftBox.class, Integer.parseInt(boxId));

    Criteria c1 = s.createCriteria(Cart.class);
    c1.add(Restrictions.eq("user", user));
    c1.add(Restrictions.eq("giftBox", box));
    c1.add(Restrictions.eq("status", 1));

    List<Cart> cart = c1.list();
    OrderStatus orderStatus = (OrderStatus) s.get(OrderStatus.class,
Checkout.ORDER_PENDING);

    for (Cart cart1 : cart) {
        Criteria c2 = s.createCriteria(CartItem.class);
        c2.add(Restrictions.eq("cart", cart1));

        List<CartItem> cartItems1 = c2.list();

        for (CartItem cartItem : cartItems1) {

            OrderItems orderItems = new OrderItems();
            orderItems.setOrders(orders);
            orderItems.setCartItem(cartItem);
            orderItems.setOrderStatus(orderStatus);
            orderItems.setQty(cartItem.getQty());

            s.save(orderItems);

            Item item = cartItem.getItem();
            if (item != null) {
                int newQty = item.getQty() - cartItem.getQty();
                if (newQty < 0) {
                    newQty = 0;
                }
                item.setQty(newQty);
                s.update(item);
            }

        }

        cart1.setStatus(0);
        s.update(cart1);

    }

    tr.commit();

    String amountStr = amount.replace(",", "");
    System.out.println(amountStr);
//

//          //payhere process
    String merahantID = "1227033";
    String merchantSecret =
"MzczMjgwNzc1NDE5NzM3NzU2NjQwMTk2MzEzOTMyMDkxMDQxNTI2";
    String orderID = "#000" + orderId;
    String currency = "LKR";
    DecimalFormat df = new DecimalFormat("0.00");
    String amountFormatted = df.format(Double.parseDouble(amountStr));
    String merchantSecretMD5 = PayHere.generateMD5(merchantSecret);
    String hash = PayHere.generateMD5(merahantID + orderID + amountFormatted +
currency + merchantSecretMD5);

    JsonObject payHereJson = new JsonObject();
    payHereJson.addProperty("sandbox", true);
    payHereJson.addProperty("merchant_id", merahantID);

    payHereJson.addProperty("return_url", "");
    payHereJson.addProperty("cancel_url", "");
    payHereJson.addProperty("notify_url", " https://9567371a1c1c.ngrok-
free.app/Web2Viva/VerifyPayments");

    payHereJson.addProperty("order_id", orderID);
    payHereJson.addProperty("items", box.getName());
    payHereJson.addProperty("amount", amountFormatted);
    payHereJson.addProperty("currency", currency);
    payHereJson.addProperty("hash", hash);

    payHereJson.addProperty("first_name", user.getFirstName());
    payHereJson.addProperty("last_name", user.getLastName());
    payHereJson.addProperty("email", user.getEmail());

    payHereJson.addProperty("phone", address.getMobile());
```

```
            payHereJson.addProperty("address", address.getLine1() + ", " + address.getLine2());
            payHereJson.addProperty("city", address.getCity().getName());
            payHereJson.addProperty("country", "Sri Lanka");

            responseObject.addProperty("message", "Order Successful");
            responseObject.addProperty("status", true);
            responseObject.add("payhereJson", new Gson().toJsonTree(payHereJson));

        } catch (Exception e) {
            e.printStackTrace();
            tr.rollback();
        }

    }
```

04. CheckSessionBox Servlet

```
@Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

//
        User user = (User) request.getAttribute("user");
        if (user != null) {

            ArrayList<CartItem> sessionBoxes = (ArrayList<CartItem>)
request.getSession().getAttribute("sesItemBox");
            if (sessionBoxes != null && !sessionBoxes.isEmpty()) {

                SessionFactory sf = HibernateUtil.getSessionFactory();
        Session s = sf.openSession();
        Transaction tr = s.beginTransaction();

        try {
            for (CartItem sessBox : sessionBoxes) {

                Item item = (Item) s.get(Item.class, sessBox.getItem().getId());

                // Create a new Criteria for each item
                Criteria c1 = s.createCriteria(CartItem.class);
                c1.add(Restrictions.eq("user", user));
                c1.add(Restrictions.eq("item", sessBox.getItem()));

                CartItem dbBox = (CartItem) c1.uniqueResult();

                if (dbBox == null) {
//                    sessBox.setUser(user);
                    s.save(sessBox);
                } else {
                    int newQty = sessBox.getQty() + dbBox.getQty();
                    if (newQty <= item.getQty()) {
                        dbBox.setQty(newQty);
//                        dbBox.setUser(user);
                        s.update(dbBox);
                    }
                }
            }

            tr.commit();
```

```
                // Clear session cart after saving
                request.getSession().setAttribute("sesItemBox", null);

            } catch (Exception e) {
                tr.rollback();
                e.printStackTrace();
            } finally {
                s.close();
            }

        }

    } else {
        System.out.println("sign in plz");
    }

  }

}
```

05. CityData Servlet

```
  @Override
  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

      SessionFactory sf = HibernateUtil.getSessionFactory();
```

```
        Session s = sf.openSession();

        Criteria c = s.createCriteria(City.class);
        List<City> cityList = c.list();

        Gson gson = new Gson();
        String toJson = gson.toJson(cityList);
        response.setContentType("application/json");
        response.getWriter().write(toJson);
        s.close();

    }
```

06. DeleteCartItems Servlet

```
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

        String prId = request.getParameter("id");
        String boxId = request.getParameter("boxId");

        Gson gson = new Gson();
        JsonObject responseObejct = new JsonObject();
        responseObejct.addProperty("status", false);

        User user = (User) request.getSession().getAttribute("user");
        if (user != null) {

            SessionFactory sf = HibernateUtil.getSessionFactory();
            Session s = sf.openSession();
            Transaction tr = s.beginTransaction();

            GiftBox giftBox = (GiftBox) s.get(GiftBox.class, Integer.valueOf(boxId));
            Item item = (Item) s.get(Item.class, Integer.valueOf(prId));

            Criteria c1 = s.createCriteria(Cart.class);
            c1.add(Restrictions.eq("user", user));
            c1.add(Restrictions.eq("giftBox", giftBox));

            Cart cart = (Cart) c1.uniqueResult();
            if (cart != null) {

                Criteria c2 = s.createCriteria(CartItem.class);
                c2.add(Restrictions.eq("cart", cart));
                c2.add(Restrictions.eq("item", item));

                if (!c2.list().isEmpty()) {
                    List<CartItem> CartItems = c2.list();

                    for (CartItem cartItem1 : CartItems) {
                        s.delete(cartItem1);
                    }

                } else {
                    responseObejct.addProperty("message", "Something went wrong. Please try
again");
```

```
            }

          }

          tr.commit();
          responseObejct.addProperty("status", true);
          responseObejct.addProperty("message", "Item Deleted");
          s.close();

        } else {
          responseObejct.addProperty("message", "no-session");
        }

        String toJson = gson.toJson(responseObejct);
        response.setContentType("application/json");
        response.getWriter().write(toJson);

    }
```

07. LoadAdminData Servlet

```
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

        Gson gson = new Gson();
        JsonObject responseObject = new JsonObject();

        responseObject.addProperty("status", false);
    SessionFactory sf = HibernateUtil.getSessionFactory();
    Session s = sf.openSession();

    User user = (User) request.getSession().getAttribute("adminUser");

    if (user != null) {
      System.out.println(user.getFirstName());
      //category list
      Criteria c1 = s.createCriteria(Category.class);
      List<Category> categoryList = c1.list();

      user.setEmail(null);
      user.setPassword(null);
      user.setVerification(null);
      responseObject.addProperty("status", true);

      s.close();
      responseObject.add("userData", gson.toJsonTree(user));
      responseObject.add("categoryList", gson.toJsonTree(categoryList));

      String toJson = gson.toJson(responseObject);
      response.setContentType("application/json");
      response.getWriter().write(toJson);

    } else {
      System.out.println("Session Expired !");
    }

  }
```

08. LoadBoxData Servlet

```
  @Override
  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

      JsonObject responseObject = new JsonObject();

      Gson gson = new Gson();

      SessionFactory sf = HibernateUtil.getSessionFactory();
      Session s = sf.openSession();

      Criteria c = s.createCriteria(GiftBox.class);
      List<GiftBox> boxList = c.list();
      if (!c.list().isEmpty()) {

          responseObject.add("boxList", gson.toJsonTree(boxList));

      } else {
          responseObject.addProperty("message", "Something went wrong! Please try again
later!");
      }

      String toJson = gson.toJson(responseObject);
      response.setContentType("application/json");
      response.getWriter().write(toJson);

  }
```

09. LoadCartItems Servlet

```
 @Override
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

     Gson gson = new Gson();
     JsonObject responseObejct = new JsonObject();
     responseObejct.addProperty("status", false);

     String boxId = request.getParameter("boxId");

     User user = (User) request.getSession().getAttribute("user");
     if (user != null) {

         SessionFactory sf = HibernateUtil.getSessionFactory();
         Session s = sf.openSession();

         GiftBox giftBox = (GiftBox) s.get(GiftBox.class, Integer.valueOf(boxId));

         Criteria c1 = s.createCriteria(Cart.class);
         c1.add(Restrictions.eq("user", user));
         c1.add(Restrictions.eq("giftBox", giftBox));
         c1.add(Restrictions.eq("status", 1));

         Cart cart = (Cart) c1.uniqueResult();
         if (cart != null) {

             Criteria c2 = s.createCriteria(CartItem.class);
```

```
            c2.add(Restrictions.eq("cart", cart));
            List<CartItem> CartItems = c2.list();


            List<Cart> cartList = c1.list();


            for (Cart cart1 : cartList) {
                cart1.setUser(null);
            }
            responseObejct.addProperty("status", true);
            responseObejct.addProperty("message", "Cart Item Loaded");
            responseObejct.add("cartItems", gson.toJsonTree(CartItems));


        } else {
            responseObejct.addProperty("message", "Cart is Empty!!");
        }


    } else {//session cart


        ArrayList<CartItem> sessionCarts = (ArrayList<CartItem>)
request.getSession().getAttribute("sesItemBox");
        if (sessionCarts != null) {
            if (sessionCarts.isEmpty()) {
                responseObejct.addProperty("message", "Cart is Empty!");
            } else {



                responseObejct.addProperty("status", true);
                responseObejct.addProperty("message", "Cart Item Loaded");
                responseObejct.add("cartItems", gson.toJsonTree(sessionCarts));
            }
        } else {
            responseObejct.addProperty("message", "Cart is Empty!");
        }

    }

    String toJson = gson.toJson(responseObejct);
    response.setContentType("application/json");
    response.getWriter().write(toJson);

}
```

10. LoadCheckoutData Servlet

```
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

        String boxId = request.getParameter("boxId");

        Gson gson = new Gson();
        JsonObject responseObject = new JsonObject();
        responseObject.addProperty("status", false);

        User sessionUser = (User) request.getSession().getAttribute("user");

        if (sessionUser == null) {
            response.setStatus(HttpServletResponse.SC_UNAUTHORIZED); // 401 error
        } else {
```

```
        SessionFactory sf = HibernateUtil.getSessionFactory();
        Session s = sf.openSession();

        Criteria c1 = s.createCriteria(Address.class);
        c1.add(Restrictions.eq("user", sessionUser));
        c1.addOrder(Order.desc("id"));
        List<Address> addressList = c1.list();
        if (addressList.isEmpty()) {
            responseObject.addProperty("message", "Account details are incomplete. Please
fill your Address details");
        } else {
            Address address = (Address) c1.list().get(0);
            address.getUser().setEmail(null);
            address.getUser().setPassword(null);
            address.getUser().setVerification(null);
            address.getUser().setCreated_at(null);
            address.getUser().setUserType(null);
            responseObject.add("userAddress", gson.toJsonTree(address));
        }

        Criteria c2 = s.createCriteria(City.class);
        c2.addOrder(Order.asc("name"));
        List<City> citylist = c2.list();
        responseObject.add("cityList", gson.toJsonTree(citylist));

        GiftBox giftBox = (GiftBox) s.get(GiftBox.class, Integer.valueOf(boxId));
        responseObject.add("giftBox", gson.toJsonTree(giftBox));



        Criteria c3 = s.createCriteria(Cart.class);
```

```
            c3.add(Restrictions.eq("user", sessionUser));
            c3.add(Restrictions.eq("giftBox", giftBox));

            Cart cart = (Cart) c3.uniqueResult();
            if (cart != null) {

                Criteria c4 = s.createCriteria(CartItem.class);
                c4.add(Restrictions.eq("cart", cart));
                List<CartItem> CartItems = c4.list();

                List<Cart> cartList = c3.list();

                for (Cart cart1 : cartList) {
                    cart1.setUser(null);
                }
                responseObject.add("cartItems", gson.toJsonTree(CartItems));

            } else {
                responseObject.addProperty("message", "empty-cart");
            }
            s.close();

        }
        responseObject.addProperty("status", true);

        String toJson = gson.toJson(responseObject);
        response.setContentType("application/json");
        response.getWriter().write(toJson);
    }
```

11. LoadItemData Servlet

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    JsonObject responseObject = new JsonObject();
    responseObject.addProperty("status", false);

    SessionFactory sf = HibernateUtil.getSessionFactory();
    Session s = sf.openSession();

    //category list
    Criteria c1 = s.createCriteria(Category.class);
    List<Category> categoryList = c1.list();

    Status status = (Status) s.get(Status.class, 1);

    Criteria c = s.createCriteria(Item.class);

    c.addOrder(Order.desc("id"));
    c.add(Restrictions.eq("status", status));

    List<Item> itemList = c.list();
    for (Item item : itemList) {
        item.setUser(null);
    }
    Gson gson = new Gson();
```

```
        responseObject.add("categoryList", gson.toJsonTree(categoryList));
        responseObject.addProperty("productCount", c.list().size());
        responseObject.add("itemList", gson.toJsonTree(itemList));

        responseObject.addProperty("status", true);

        String toJson = gson.toJson(responseObject);
        response.setContentType("application/json");
        response.getWriter().write(toJson);
        s.close();
    }
```

12. LoadSingleProduct Servlet

```
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

        Gson gson = new Gson();
        JsonObject responseObject = new JsonObject();
        responseObject.addProperty("status", false);

        String productId = request.getParameter("id");
        String boxId = request.getParameter("boxId");

        SessionFactory sf = HibernateUtil.getSessionFactory();
        Session s = sf.openSession();

        if (productId != null) {
```

```
    if (Util.isInteger(productId)) {

        try {
            Item item = (Item) s.get(Item.class, Integer.valueOf(productId));
            if (item.getStatus().getName().equals("Active")) {

                item.getUser().setEmail(null);
                item.getUser().setPassword(null);
                item.getUser().setVerification(null);
                item.getUser().setCreated_at(null);
                item.getUser().setUserType(null);

                responseObject.add("item", gson.toJsonTree(item));
                responseObject.addProperty("status", true);

            } else {
                responseObject.addProperty("message", "Product not Found!");
            }

        } catch (Exception e) {
            responseObject.addProperty("message", "Product not Found!");
        }
    } else {
        System.out.println("a");
    }
} else if (boxId != null) {

    if (Util.isInteger(boxId)) {
        try {
                GiftBox giftBox = (GiftBox) s.get(GiftBox.class, Integer.valueOf(boxId));

                responseObject.add("giftBox", gson.toJsonTree(giftBox));
                responseObject.addProperty("status", true);

            } catch (Exception e) {
                responseObject.addProperty("message", "Product not Found!");
            }
        }

    }

    response.setContentType("application/json");
    String toJson = gson.toJson(responseObject);
    response.getWriter().write(toJson);
}
```

13. MyAccount Servlet

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    HttpSession ses = request.getSession(false);
    JsonObject responseObject = new JsonObject();

    if (ses != null && ses.getAttribute("user") != null) {
        User user = (User) ses.getAttribute("user");
    }

    responseObject.addProperty("firstName", user.getFirstName());
    responseObject.addProperty("lastName", user.getLastName());
    responseObject.addProperty("email", user.getEmail());

    String since = new SimpleDateFormat("MMM yyyy").format(user.getCreated_at());
    responseObject.addProperty("since", since);

    Gson gson = new Gson();

    SessionFactory sf = HibernateUtil.getSessionFactory();
    Session s = sf.openSession();

    Criteria c = s.createCriteria(Address.class);
    c.add(Restrictions.eq("user", user));

    if (!c.list().isEmpty()) {
        System.out.println(c.list());
        List<Address> addressList = c.list();
        responseObject.add("addressList", gson.toJsonTree(addressList));
    }

    String toJson = gson.toJson(responseObject);
    response.setContentType("application/json");
    response.getWriter().write(toJson);

} else {
    response.setStatus(HttpServletResponse.SC_UNAUTHORIZED); // 401 error
}
```

```
    }


    @Override
    protected void doPut(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {


        Gson gson = new Gson();
        JsonObject userData = gson.fromJson(request.getReader(), JsonObject.class);


        String firstName = userData.get("firstName").getAsString();
        String lastName = userData.get("lastName").getAsString();
        String email = userData.get("email").getAsString();
        String line1 = userData.get("line1").getAsString();
        String line2 = userData.get("line2").getAsString();
        int cityId = userData.get("cityId").getAsInt();
        String postalCode = userData.get("postalCode").getAsString();


        JsonObject responseObject = new JsonObject();
        responseObject.addProperty("status", false);


        if (firstName.isEmpty()) {
            responseObject.addProperty("message", "Please Enter your First Name first!");
        } else if (lastName.isEmpty()) {
            responseObject.addProperty("message", "Please Enter your Last Name!");
        } else if (email.isEmpty()) {
            responseObject.addProperty("message", "Please Enter your Email Address!");
        } else if (!Util.isEmailValid(email)) {
            responseObject.addProperty("message", "Invalid Email Address!");
        } else if (line1.isEmpty()) {
            responseObject.addProperty("message", "Please Enter your Line 01!");
```

```
} else if (line2.isEmpty()) {
    responseObject.addProperty("message", "Please Enter your Line 02!");
} else if (cityId == 0) {
    responseObject.addProperty("message", "Please Select City!");
} else if (postalCode.isEmpty()) {
    responseObject.addProperty("message", "Please Enter Postal code!");
} else if (!Util.isPostalCodeValid(postalCode)) {
    responseObject.addProperty("message", "Invalid Postal Code!");
} else {

    HttpSession ses = request.getSession();
    if (ses.getAttribute("user") != null) {

        User u = (User) ses.getAttribute("user");

        SessionFactory sf = HibernateUtil.getSessionFactory();
        Session s = sf.openSession();

        Criteria c = s.createCriteria(User.class);
        c.add(Restrictions.eq("email", u.getEmail()));

        User u1 = (User) c.list().get(0);

        u1.setFirstName(firstName);
        u1.setLastName(lastName);
        City city = (City) s.load(City.class, cityId);

        Criteria c1 = s.createCriteria(Address.class);
        c1.add(Restrictions.eq("user", u1));
```

```
Address address = (Address) c1.list().get(0);
if (c1.list().isEmpty()) {
    Address address1 = new Address();
    address1.setLine1(line1);
    address1.setLine2(line2);
    address1.setCity(city);
    address1.setPostalCode(postalCode);
    address1.setUser(u1);
    System.out.println("sd");

    s.save(address);

} else {

    address.setLine1(line1);
    address.setLine2(line2);
    address.setCity(city);
    address.setPostalCode(postalCode);
    address.setUser(u1);
    System.out.println("sdas");

    s.update(address);
}

ses.setAttribute("user", u1);

s.merge(u1);

s.beginTransaction().commit();
```

```
            responseObject.addProperty("status", true);
            responseObject.addProperty("message", "Account Details Updated");

            s.close();

          } else {
            response.setStatus(HttpServletResponse.SC_UNAUTHORIZED); // 401 error
          }

        }

        String toJson = gson.toJson(responseObject);
        response.setContentType("application/json");
        response.getWriter().write(toJson);

    }
```

14. ResetPassword Servlet

```
@Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

        Gson gson = new Gson();
        JsonObject user = gson.fromJson(request.getReader(), JsonObject.class);

        String newPassword = user.get("newPassword").getAsString();
        String confirmPassword = user.get("confirmPassword").getAsString();
        JsonObject responseObject = new JsonObject();
        responseObject.addProperty("status", false);

        if (newPassword.isEmpty()) {
            responseObject.addProperty("message", "Please Enter New your Password!");
        } else if (!Util.isPasswordValid(newPassword)) {
            responseObject.addProperty("message", "The Password must contains at least
uppercase, lowercase, number, special character and to be minimum 8 characters long!");
        } else if (confirmPassword.isEmpty()) {
            responseObject.addProperty("message", "Please Enter your Confirm Password!");
        } else if (!newPassword.equals(confirmPassword)) {
            responseObject.addProperty("message", "The Passwords Fields are not same!");
        } else {

            SessionFactory sf = HibernateUtil.getSessionFactory();
            Session s = sf.openSession();

            HttpSession ses = request.getSession();
            String email = ses.getAttribute("email").toString();

            Criteria c = s.createCriteria(User.class);
            c.add(Restrictions.eq("email", email));

            if (c.list().isEmpty()) {
                responseObject.addProperty("message", "Verification code Expired. Please try
again!");
            } else {

                User u = (User) c.list().get(0);
                u.setPassword(newPassword);
            responseObject.addProperty("status", true);

            responseObject.addProperty("message", "Verification Successfully");
          }

          s.close();

        }

        String responseText = gson.toJson(responseObject);
        response.setContentType("application/json");
        response.getWriter().write(responseText);

    }
```

15. SaveProduct Servlet

```
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        String prName = request.getParameter("prName");
        String prDesc = request.getParameter("prDesc");
        String categoryId = request.getParameter("category");
        String price = request.getParameter("price");
        String qty = request.getParameter("qty");

        JsonObject responseObject = new JsonObject();
        responseObject.addProperty("status", false);
```

```
Session s = HibernateUtil.getSessionFactory().openSession();

Transaction tr = s.beginTransaction();

if (prName.isEmpty()) {
   responseObject.addProperty("message", "Please Enter Product Name First!");
} else if (prDesc.isEmpty()) {
   responseObject.addProperty("message", "Please Enter Description!");
} else if (prDesc.isEmpty()) {
   responseObject.addProperty("message", "Please Enter Description!");
} else {
}

User user = (User) request.getSession().getAttribute("adminUser");
try {

   if (user != null) {
      Criteria c1 = s.createCriteria(Category.class);
      c1.add(Restrictions.eq("id", Integer.parseInt(categoryId)));

      if (!c1.list().isEmpty()) {
         Category category = (Category) c1.uniqueResult();
         Status status = (Status) s.get(Status.class, 1);

         Item item = new Item();
         item.setName(prName);
         item.setDescription(prDesc);
         item.setCategory(category);
```

```
            item.setSize(0);
            item.setPrice(Double.parseDouble(price));
            item.setQty(Integer.parseInt(qty));
            item.setUser(user);
            item.setCreated_at(new Date());
            item.setStatus(status);

            int id = (int)s.save(item);
            tr.commit();

            Part part1 = request.getPart("img1");
            Part part2 = request.getPart("img2");
            Part part3 = request.getPart("img3");
            Part part4 = request.getPart("img4");

            String appPath = getServletContext().getRealPath("");

            String newPath = appPath.replace("build" + File.separator + "web", "web" +
File.separator + "assets" + File.separator + "product-images");

            File productFolder = new File(newPath, String.valueOf(id));
            productFolder.mkdir();

            File file1 = new File(productFolder, "image1.jpeg");
            Files.copy(part1.getInputStream(), file1.toPath(),
StandardCopyOption.REPLACE_EXISTING);

            File file2 = new File(productFolder, "image2.jpeg");
            Files.copy(part2.getInputStream(), file2.toPath(),
StandardCopyOption.REPLACE_EXISTING);
```

```
                File file3 = new File(productFolder, "image3.jpeg");
                Files.copy(part3.getInputStream(), file3.toPath(),
StandardCopyOption.REPLACE_EXISTING);

                File file4 = new File(productFolder, "image4.jpeg");
                Files.copy(part4.getInputStream(), file4.toPath(),
StandardCopyOption.REPLACE_EXISTING);

                responseObject.addProperty("status", true);
                responseObject.addProperty("message", "Product Added Succefully");

              } else {
                responseObject.addProperty("message", "Invalid Category");
              }

            } else {
              responseObject.addProperty("message", "Session Expired. Sign in First!");
            }
        } catch (Exception e) {

            if (tr != null && tr.isActive()) {
              tr.rollback();
            }
        } finally {
            Gson gson = new Gson();
            String toJson = gson.toJson(responseObject);
            response.setContentType("application/json");
            response.getWriter().write(toJson);
            s.close();
        }

    }
```

16. SearchItems Servlet

```
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

        String searchText = request.getParameter("search");

        JsonObject responseObject = new JsonObject();
        responseObject.addProperty("status", false);

        Gson gson = new Gson();
        SessionFactory sf = HibernateUtil.getSessionFactory();
        Session s = sf.openSession();

        if (!searchText.isEmpty()) {
            Criteria c1 = s.createCriteria(Item.class);
            c1.add(Restrictions.ilike("name", "%" + searchText + "%"));

            List<Item> itemList = c1.list();

            responseObject.addProperty("status", true);
            responseObject.add("itemList", gson.toJsonTree(itemList));
        } else {
            responseObject.addProperty("message", "Search text is empty");
        }

        String toJson = gson.toJson(responseObject);
        response.setContentType("application/json");
        response.getWriter().write(toJson);
        s.close();
    }
```

17. SearchProduct Servlet

```
    public static final int MAX_RESULT = 4;
    public static final int ACTIVE_ID = 1;

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

        Gson gson = new Gson();
        JsonObject responseObject = new JsonObject();
        responseObject.addProperty("status", false);

        JsonObject requestJsonObject = gson.fromJson(request.getReader(), JsonObject.class);

        SessionFactory sf = HibernateUtil.getSessionFactory();
        Session s = sf.openSession();
```

```
Criteria c1 = s.createCriteria(Item.class);

if (requestJsonObject.has("categoryName")) {
   String categoryName = requestJsonObject.get("categoryName").getAsString();
   Criteria c2 = s.createCriteria(Category.class);
   c2.add(Restrictions.eq("name", categoryName));
   Category category = (Category) c2.uniqueResult();
   if (category != null) {
      c1.add(Restrictions.eq("category", category));
      System.out.println(category);

   } else {
      responseObject.addProperty("message", "Products not found!");
   }
}



if (requestJsonObject.has("priceStart") && requestJsonObject.has("priceEnd")) {
   double priceStart = requestJsonObject.get("priceStart").getAsDouble();
   double priceEnd = requestJsonObject.get("priceEnd").getAsDouble();

   c1.add(Restrictions.ge("price", priceStart));
   c1.add(Restrictions.le("price", priceEnd));

}

if (requestJsonObject.has("sortValue")) {
   String sortValue = requestJsonObject.get("sortValue").getAsString();
```

```
    if (sortValue.equals("Sort by Latest")) {
        c1.addOrder(Order.desc("id"));
    } else if (sortValue.equals("Sort by Oldest")) {
        c1.addOrder(Order.asc("id"));
    } else if (sortValue.equals("Sort by Name")) {
        c1.addOrder(Order.asc("name"));
    } else if (sortValue.equals("Sort by Price")) {
        c1.addOrder(Order.asc("price"));
    }
}


System.out.println(c1.list().size());
responseObject.addProperty("productCount", c1.list().size());

if (requestJsonObject.has("firstResult")) {
    int firstResult = requestJsonObject.get("firstResult").getAsInt();
    c1.setFirstResult(firstResult);
    c1.setMaxResults(SearchProduct.MAX_RESULT);
}

Status status = (Status) s.get(Status.class, ACTIVE_ID);
c1.add(Restrictions.eq("status", status));
List<Item> itemList = c1.list();

s.close();

responseObject.add("itemList", gson.toJsonTree(itemList));
responseObject.addProperty("status", true);
```

```
        String toJson = gson.toJson(responseObject);
        response.setContentType("application/json");
        response.getWriter().write(toJson);
    }
```

18. SendEmail Servlet

```
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

        final String email = request.getParameter("email");

        JsonObject responseObject = new JsonObject();
        responseObject.addProperty("status", false);

        if (email.isEmpty()) {
            responseObject.addProperty("message", "Please Enter your Registered Email
Address");
        } else {

            SessionFactory sf = HibernateUtil.getSessionFactory();
            Session s = sf.openSession();

            Criteria c = s.createCriteria(User.class);
            c.add(Restrictions.eq("email", email));

            if (!c.list().isEmpty()) {
```

```
            User u = (User) c.list().get(0);

            final String verificationCode = Util.generateCode();
            u.setVerification(verificationCode);

            HttpSession ses = request.getSession();

            new Thread(new Runnable() {
                @Override
                public void run() {
                    Mail.sendMail(email, "Boxella - Verification", "<table border=\"0\"
cellpadding=\"0\" cellspacing=\"0\" width=\"100%\">\n"
                        + "        <!-- LOGO -->\n"
                        + "        <tr>\n"
                        + "            <td bgcolor=\"#FFA73B\" align=\"center\">\n"
                        + "                <table border=\"0\" cellpadding=\"0\" cellspacing=\"0\"
width=\"100%\" style=\"max-width: 600px;\">\n"
                        + "                    <tr>\n"
                        + "                        <td align=\"center\" valign=\"top\" style=\"padding: 40px
10px 40px 10px;\"> </td>\n"
                        + "                    </tr>\n"
                        + "                </table>\n"
                        + "            </td>\n"
                        + "        </tr>\n"
                        + "        <tr>\n"
                        + "            <td bgcolor=\"#FFA73B\" align=\"center\" style=\"padding: 0px
10px 0px 10px;\">\n"
                        + "                <table border=\"0\" cellpadding=\"0\" cellspacing=\"0\"
width=\"100%\" style=\"max-width: 600px;\">\n"
                        + "                    <tr>\n"
                        + "                    <td bgcolor=\"#ffffff\" align=\"center\" valign=\"top\" style=\"padding: 40px 20px 20px 20px; border-radius: 4px 4px 0px 0px; color: #111111; font-family: \"Lato\", Helvetica, Arial, sans-serif; font-size: 48px; font-weight: 400; letter-spacing: 4px; line-height: 48px;\">\n"
                        + "                        <h1 style=\"font-size: 48px; font-weight: 400; font-family: \"Lato\", Helvetica, Arial, sans-serif; margin: 2;\">Reset Password!</h1> <img src=\" https://img.icons8.com/clouds/100/000000/handshake.png\" width=\"125\" height=\"120\" style=\"display: block; border: 0px;\" />\n"
                        + "                    </td>\n"
                        + "                </tr>\n"
                        + "            </table>\n"
                        + "        </td>\n"
                        + "    </tr>\n"
                        + "    <tr>\n"
                        + "        <td bgcolor=\"#f4f4f4\" align=\"center\" style=\"padding: 0px 10px 0px 10px; \">\n"
                        + "            <table border=\"0\" cellpadding=\"0\" cellspacing=\"0\" width=\"100%\" style=\"max-width: 600px;\">\n"
                        + "                <tr>\n"
                        + "                    <td bgcolor=\"#ffffff\" align=\"left\" style=\"padding: 20px 30px 40px 30px; color: #666666; font-family: \"Lato\", Helvetica, Arial, sans-serif; font-size: 18px; font-weight: 400; line-height: 25px;\">\n"
                        + "                        <p style=\"margin: 0;  margin-top: 20px;\" align=\"center\">You are almost set to start enjoying <b>BOXELLA</b>. Simply copy the verofication code below to reset password and get started.</p>\n"
                        + "                    </td>\n"
                        + "                </tr>\n"
                        + "                <tr>\n"
                        + "                    <td bgcolor=\"#ffffff\" align=\"left\">\n"
                        + "                        <table width=\"100%\" border=\"0\" cellspacing=\"0\" cellpadding=\"0\">\n"
                        + "                            <tr>\n"
                        + "                                <td bgcolor=\"#ffffff\" align=\"center\" style=\"padding: 20px 30px 60px 30px;\">\n"
                        + "                    <table border=\"0\" cellspacing=\"0\"
        cellpadding=\"0\">\n"
                        + "                      <tr>\n"
                        + "                        <td align=\"center\"  bgcolor=\"#FFA73B\"
        target=\"_blank\" style=\"border-radius: 3px; font-size: 20px; font-family: Helvetica, Arial,
        sans-serif; color: #ffffff; text-decoration: none; \n"
                        + "                        color: #ffffff; text-decoration: none; padding:
        15px 25px; border-radius: 2px; margin-top:20px; border: 1px solid #FFA73B; display: inline-
        block;\">" + verificationCode + "</td>\n"
                        + "                      </tr>\n"
                        + "                    </table>\n"
                        + "                  </td>\n"
                        + "                </tr>\n"
                        + "              </table>\n"
                        + "            </td>\n"
                        + "          </tr> <!-- COPY -->\n"
                        + "          \n"
                        + "          <tr>\n"
                        + "            <td bgcolor=\"#ffffff\" align=\"left\" style=\"padding: 0px
        30px 20px 30px; color: #666666; font-family: \"Lato\", Helvetica, Arial, sans-serif; font-size:
        18px; font-weight: 400; line-height: 25px;\">\n"
                        + "              <p style=\"margin: 0;\">If you have any questions, just
        reply to this email&mdash;we,re always happy to help out.</p>\n"
                        + "            </td>\n"
                        + "          </tr>\n"
                        + "          <tr>\n"
                        + "            <td bgcolor=\"#ffffff\" align=\"left\" style=\"padding: 0px
        30px 40px 30px; border-radius: 0px 0px 4px 4px; color: #666666; font-family: \"Lato\",
        Helvetica, Arial, sans-serif; font-size: 18px; font-weight: 400; line-height: 25px;\">\n"
                        + "              <p style=\"margin: 0;\">Cheers,<br>Boxella
        Team</p>\n"
                        + "            </td>\n"
                        + "          </tr>\n"
```

```
            + "            </table>\n"
            + "         </td>\n"
            + "      </tr>\n"
            + "      <tr>\n"
            + "         <td bgcolor=\"#f4f4f4\" align=\"center\" style=\"padding: 30px 10px 0px 10px;\">\n"
            + "            <table border=\"0\" cellpadding=\"0\" cellspacing=\"0\" width=\"100%\" style=\"max-width: 600px;\">\n"
            + "               <tr>\n"
            + "                  <td bgcolor=\"#FFECD1\" align=\"center\" style=\"padding: 30px 30px 30px 30px; border-radius: 4px 4px 4px 4px; color: #666666; font-family: \"Lato\", Helvetica, Arial, sans-serif; font-size: 18px; font-weight: 400; line-height: 25px;\">\n"
            + "                     <h2 style=\"font-size: 20px; font-weight: 400; color: #111111; margin: 0;\">Need more help?</h2>\n"
            + "                     <p style=\"margin: 0;\"><a href=\"#\" target=\"_blank\" style=\"color: #FFA73B;\">We&rsquo;re here to help you out</a></p>\n"
            + "                  </td>\n"
            + "               </tr>\n"
            + "            </table>\n"
            + "         </td>\n"
            + "      </tr>\n"
            + "      <tr>\n"
            + "         <td bgcolor=\"#f4f4f4\" align=\"center\" style=\"padding: 0px 10px 0px 10px;\">\n"
            + "            <table border=\"0\" cellpadding=\"0\" cellspacing=\"0\" width=\"100%\" style=\"max-width: 600px;\">\n"
            + "               <tr>\n"
            + "                  <td bgcolor=\"#f4f4f4\" align=\"left\" style=\"padding: 0px 30px 30px 30px; color: #666666; font-family: \"Lato\", Helvetica, Arial, sans-serif; font-size: 14px; font-weight: 400; line-height: 18px;\"> <br>\n"
            + "                     </td>\n"
            + "               </tr>\n"
```

```
                    + "                </table>\n"
                    + "            </td>\n"
                    + "        </tr>\n"
                    + "    </table>");
            }
        }).start();

        ses.setAttribute("email", email);
        s.merge(u);
        s.beginTransaction().commit();
        responseObject.addProperty("message", "Check Your Email and Get your
Verification Code!");
        responseObject.addProperty("status", true);

      } else {
        responseObject.addProperty("message", "Something went wronr! Please
Register");
      }

    }

    Gson gson = new Gson();

    String toJson = gson.toJson(responseObject);
    response.setContentType("application/json");
    response.getWriter().write(toJson);

  }
```

19. SignIn Servlet

```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

    Gson gson = new Gson();
    JsonObject user = gson.fromJson(request.getReader(), JsonObject.class);

    String email = user.get("email").getAsString();
    String password = user.get("password").getAsString();

    JsonObject responseObject = new JsonObject();
    responseObject.addProperty("status", false);

    if (email.isEmpty()) {
        responseObject.addProperty("message", "Please Enter your Email Address!");
    } else if (!Util.isEmailValid(email)) {
        responseObject.addProperty("message", "Invalid Email Address!");
    } else if (password.isEmpty()) {
        responseObject.addProperty("message", "Please Enter your Password!");
    } else if (!Util.isPasswordValid(password)) {
        responseObject.addProperty("message", "The Password must contains at least
uppercase, lowercase, number, special character and to be minimum 8 characters long!");
    } else {

        SessionFactory sf = HibernateUtil.getSessionFactory();
        Session s = sf.openSession();
        Criteria c1 = s.createCriteria(User.class);
        c1.add(Restrictions.eq("email", email));
        c1.add(Restrictions.eq("password", password));

        if (c1.list().isEmpty()) {
            responseObject.addProperty("message", "Invalid credentials!");
            responseObject.addProperty("status", false);

        } else {

            User u = (User) c1.list().get(0);
            HttpSession ses = request.getSession();

            if (u.getUserType().getId() == 1) {
                responseObject.addProperty("message", "Admin");
                ses.setAttribute("adminUser", u);
            } else {
                ses.setAttribute("user", u);
                responseObject.addProperty("message", "Client");
            }

            responseObject.addProperty("status", true);

        }
        s.close();
    }

    String responseText = gson.toJson(responseObject);
    response.setContentType("application/json");
```

```
        response.getWriter().write(responseText);

    }
```

20. SignOut Servlet

```
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

        JsonObject responseObject = new JsonObject();
        responseObject.addProperty("status", false);

        HttpSession ses = request.getSession();
        if (ses != null) {
            ses.invalidate();
            responseObject.addProperty("status", true);
        }else{
            System.out.println("alreaday signOut");
        }

        Gson gson = new Gson();
        String toJson = gson.toJson(responseObject);
        response.setContentType("application/json");
        response.getWriter().write(toJson);

    }
```

21. SignUp Servlet

```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

    Gson gson = new Gson();
    JsonObject user = gson.fromJson(request.getReader(), JsonObject.class);

    String firstName = user.get("firstName").getAsString();
    String lastName = user.get("lastName").getAsString();
    String email = user.get("email").getAsString();
    String password = user.get("password").getAsString();
    String confirmPassword = user.get("confirmPassword").getAsString();

    JsonObject responseObject = new JsonObject();
    responseObject.addProperty("status", false);

    if (firstName.isEmpty()) {
        responseObject.addProperty("message", "Please Enter your First Name first!");
    } else if (lastName.isEmpty()) {
        responseObject.addProperty("message", "Please Enter your Last Name!");
    } else if (email.isEmpty()) {
        responseObject.addProperty("message", "Please Enter your Email Address!");
    } else if (!Util.isEmailValid(email)) {
        responseObject.addProperty("message", "Invalid Email Address!");
    } else if (password.isEmpty()) {
        responseObject.addProperty("message", "Please Enter your Password!");
    } else if (!Util.isPasswordValid(password)) {
        responseObject.addProperty("message", "The Password must contains at least
uppercase, lowercase, number, special character and to be minimum 8 characters long!");
    } else if (confirmPassword.isEmpty()) {
        responseObject.addProperty("message", "Please Enter Confirm Password!");
    } else if (!password.equals(confirmPassword)) {
        responseObject.addProperty("message", "The Passwords Fields are not same!");
    } else {

        SessionFactory sf = HibernateUtil.getSessionFactory();
        Session s = sf.openSession();

        Criteria c = s.createCriteria(User.class);
        c.add(Restrictions.eq("email", email));

        if (!c.list().isEmpty()) {
            responseObject.addProperty("message", "User already exists!");
        } else {

            UserType userType = (UserType) s.get(UserType.class, 2);

            User u = new User();
            u.setFirstName(firstName);
            u.setLastName(lastName);
            u.setEmail(email);
            u.setPassword(password);
            u.setUserType(userType);
            u.setVerification("verified");

            u.setCreated_at(new Date());
            s.save(u);
            s.beginTransaction().commit();

            //session management
            HttpSession ses = request.getSession();
            ses.setAttribute("email", email);

            responseObject.addProperty("status", true);
            responseObject.addProperty("message", "Regsitration Successfully. Sign In Now");

        }

        s.close();

    }

    String responseText = gson.toJson(responseObject);
    response.setContentType("application/json");
    response.getWriter().write(responseText);

}
```

22. VerifyAccount Servlet

```
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
```

```
String vcode = request.getParameter("vcode");

JsonObject responseObject = new JsonObject();
responseObject.addProperty("status", false);

Gson gson = new Gson();

if (vcode.isEmpty()) {
  responseObject.addProperty("message", "Please Enter Verification code First!");
} else {

  SessionFactory sf = HibernateUtil.getSessionFactory();
  Session s = sf.openSession();

  Criteria c = s.createCriteria(User.class);
  c.add(Restrictions.eq("verification", vcode));

  if (c.list().isEmpty()) {
    responseObject.addProperty("message", "Invalid Verification Code!");
  } else {

    responseObject.addProperty("status", true);

    responseObject.addProperty("message", "Verification Successfully");

  }
  s.close();
}
```

```
        String responseText = gson.toJson(responseObject);

        response.setContentType("application/json");

        response.getWriter().write(responseText);


    }
```

23. VerifyPayments Servlet

```
    private static final int PAYHERE_SUCCESS = 2;


    @Override

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {


        String merchant_id = request.getParameter("merchant_id");

        String order_id = request.getParameter("order_id");

        String payHere_amount = request.getParameter("payHere_amount");

        String payHere_currency = request.getParameter("payHere_currency");

        String status_code = request.getParameter("status_code");

        String md5sig = request.getParameter("md5sig");


        String merchantSecret =
"MzczMjgwNzc1NDE5NzM3NzU2NjQwMTk2MzEzOTMyMDkxMDQxNTI2";

        String merchantSecretMD5 = PayHere.generateMD5(merchantSecret);

        String hash = PayHere.generateMD5(merchant_id + order_id + payHere_amount +
payHere_currency + merchantSecretMD5);


        if (md5sig.equals(hash) && Integer.parseInt(status_code) ==
VerifyPayments.PAYHERE_SUCCESS) {

            System.out.println("Payment Complete. Order id:"+ order_id);
        String orderId = order_id.substring(3);

        System.out.println(orderId);

    }


}
```