MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Mini Project Report
of
Database Systems Lab (CSE 2262)

E-Shopping Cart

SUBMITTED
BY

Nikhil Nair, 9, 220962013, AIML B
Arnav Karnik, 10, 220962021, AIML B

Department of Computer Science and Engineering
Manipal Institute of Technology, Manipal.
April 2024

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Manipal
16/04/2024

CERTIFICATE

This is to certify that the project titled **Shopping Cart** is a record of the bonafide work done by **Nikhil Nair (Reg. No. 220962013), Arnav Karnik (Reg. No. 220962021)** submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology (B.Tech.) in COMPUTER SCIENCE & ENGINEERING of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Institute of Manipal Academy of Higher Education), during the academic year 2024-2025.

Name and Signature of Examiners:

1. **Dr. Anup Bhat B, Assistant Professor, Dept. of CSE**
2. **Dr. Vijaya Arjunan, Additional Professor, Dept. of CSE**
3. **Prof. Musica Supriya, Assistant Professor, CSE Dept.**

TABLE OF CONTENTS

Introduction:

An e-shopping cart, also known as an electronic shopping cart or online shopping cart, is a fundamental component of e-commerce websites that allows users to select and store items they wish to purchase during their online shopping experience. It mimics the functionality of a physical shopping cart in a brick-and-mortar store, where customers can gather products before proceeding to checkout.

Here are the key features and functionalities of an e-shopping cart:

1. **Product Selection**: Users can browse through a catalog of products presented on the e-commerce website and select items they want to purchase.

2. **Adding Items to Cart**: Each selected item is added to the virtual shopping cart. Users can specify quantities and options (e.g., size, color) for each item.

3. **Viewing Cart Contents**: Users can view the contents of their shopping cart at any time during their shopping session. This includes a summary of selected items, quantities, prices, and subtotal.

4. **Cart Management**: Users can modify their cart by adjusting quantities, removing items, or adding more items.

Overall, an e-shopping cart system streamlines the online shopping experience, providing users with a convenient way to select and organize items for purchase, leading to higher customer satisfaction and improved sales for e-commerce businesses.

Problem Statement and Objectives

Unit 2.1- Problem Statement

In the rapidly growing e-commerce industry, there is a need to develop an efficient and user-friendly e-shopping cart system using Java, JavaFX, and SQL. The goal is to create a robust application that allows customers to browse products, add items to their cart, and complete purchases securely. This project aims to address the following challenges and requirements.

1. User Interface Development: Design and implement a visually appealing user interface using JavaFX that enables customers to easily navigate through product categories, view item details, manage their shopping cart, and proceed to checkout.

2. Back-end Logic: Develop the back end functionality in Java to handle business logic such as product retrieval, cart management (add/remove items, update quantities), order processing, and integration with the database.

3. Database Integration: Implement SQL database operations to store and manage product information, user data (including cart contents), and order details. Ensure efficient data retrieval and storage for seamless application performance.

4. **Cart Management Features**: Enable users to perform various cart management tasks, including adding/removing items, updating quantities, calculating total costs, and handling inventory updates in real-time.

Unit 2.2 – Objectives:

The primary objectives of developing the e-shopping cart system using Java, JavaFX, and SQL are:

1. User-Friendly Interface: Create an intuitive and responsive user interface that enhances the overall shopping experience and encourages user engagement.

2. Efficient Back-end Processing: Implement efficient back end processes to handle complex business logic, database operations, and seamless integration with the user interface.

3. Database Management: Establish a robust database schema and optimize SQL queries to support seamless data management, retrieval, and storage for the e-commerce application.

By achieving these objectives, the developed e-shopping cart system will offer a reliable and feature-rich platform for users to browse, select, and purchase products online, contributing to the success of e-commerce businesses in today's competitive market.

Methodology:

Creating an e-shopping cart using JavaFX for the user interface, Java for the back end logic, and SQL for database operations involves several steps. Below is a high-level methodology to guide through the process:

1. **Design the Database Schema**

First, design the database schema to store products, users, and orders. You can use a tool like SQL plus to design and visualize your database schema.

Example tables:

- **`Product`**: id, name, price, quantity
- **`Customer`**: id, name, email
- **`Order`**: id, customer_id, total_amount, order_date

2. **Set Up Database Connectivity**

Establish a connection to the database within the Java application using JDBC (Java Database Connectivity). This will allow to perform CRUD (Create, Read, Update, Delete) operations on the database from Java.

```
public class ShoppingCart extends Application {

private static final String JDBC_URL = "jdbc:oracle:thin:@localhost:1521:XE";

private static final String USERNAME = "system";

 private static final String PASSWORD = "dnn12345";
 private Connection connection;



Eg-   private void fetchProducts(ListView<String> productList) {

        try (Statement statement = connection.createStatement();

             ResultSet resultSet = statement.executeQuery("SELECT name FROM Product")) {

            while (resultSet.next()) {

                productList.getItems().add(resultSet.getString("name"));

            }

        } catch (SQLException e) {

            e.printStackTrace();

            showAlert("Failed to fetch products from the database.");

        }

    }
```

3. **Create JavaFX User Interface**

Designing the e-shopping cart UI using JavaFX components like buttons, labels, text fields, and tables. We can use Scene Builder to visually create the UI layout and then integrate it with your Java code.

4. **Implement Backend Logic**

Product Management: Implement functionalities to add, remove, update, and display products from the database.

- Shopping Cart Operations: Implement operations to add items to the cart, update quantities, and place orders.

```
Eg - public class ProductDAO {

    public void addProduct(Product product) {

        // SQL INSERT operation to add product to database

    }

      public void removeProduct(int productId) {

        // SQL DELETE operation to remove product from database

    }

    // Implement methods to update and retrieve products

}
```

5. **Handle Events and Interactions**

Set up event handlers for UI components to respond to user interactions (e.g., button clicks).

6. **Integrate Database Operations**

Use JDBC to execute SQL queries and update database records based on user actions (e.g., adding/removing items from cart, placing an order).

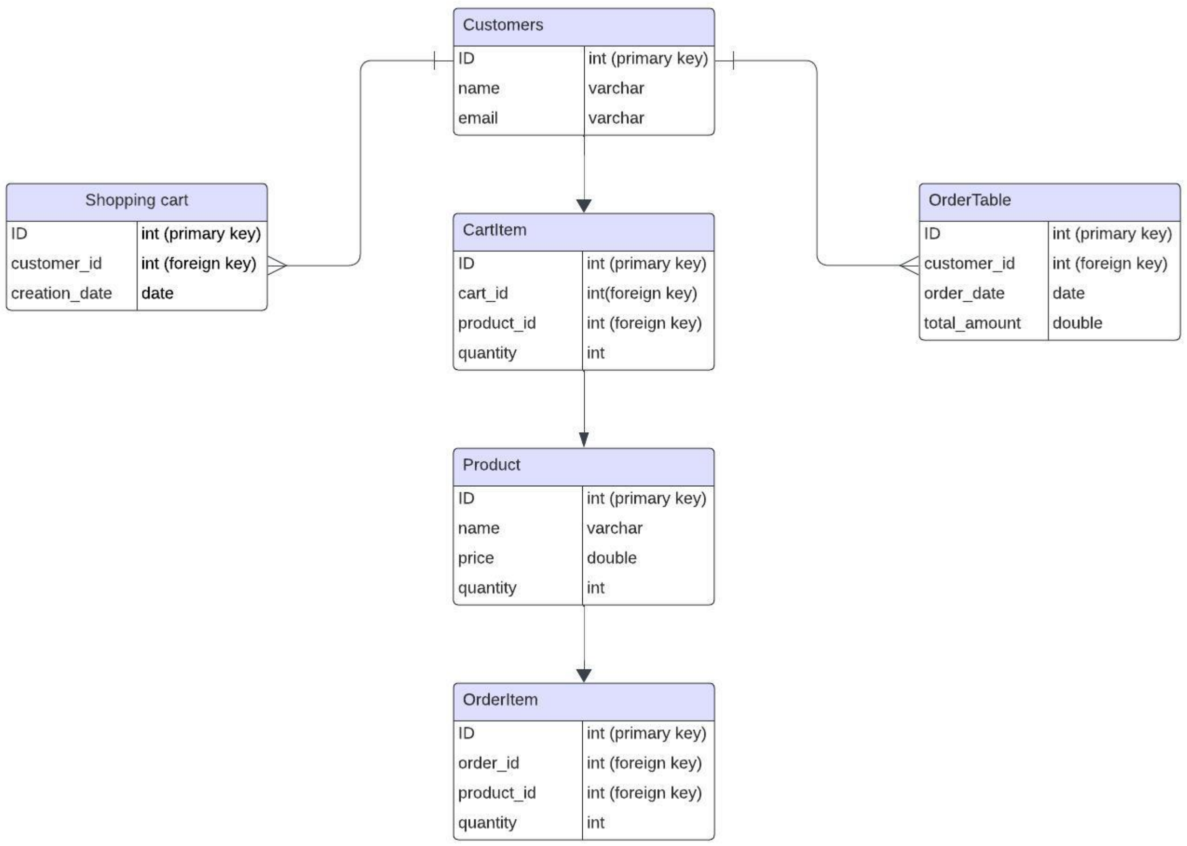
7. **Test the Application**

Test the e-shopping cart application to ensure that all functionalities work as expected, including product management, user authentication, cart operations, and order placement.

8. **Enhance and Deploy**

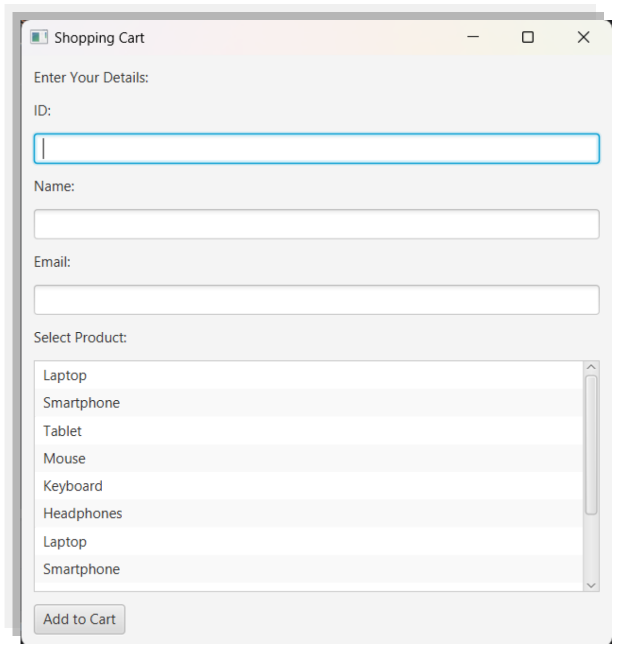
Refine the application by adding error handling, validation, and additional features (e.g., search functionality, sorting). Finally, package the application for deployment.

ER Diagram

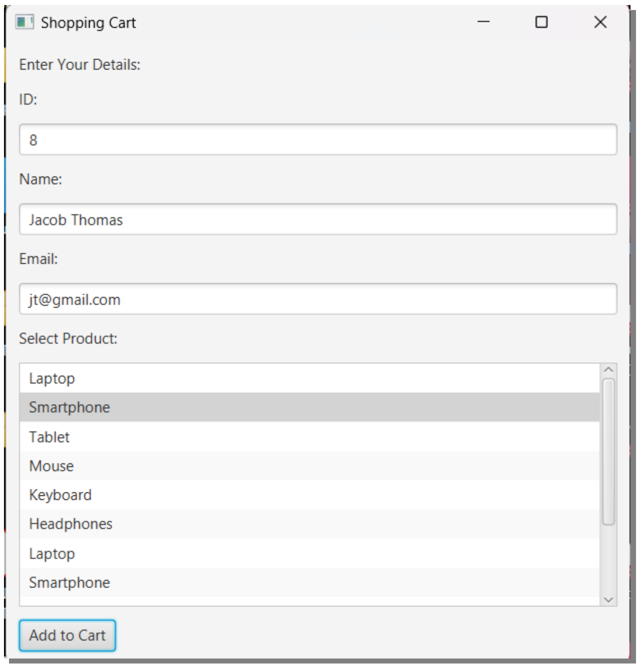


Customers
ID | int (primary key)
name | varchar
email | varchar
Shopping cart
ID | int (primary key)
customer_id | int (foreign key)
creation_date | date
CartItem
ID | int (primary key)
cart_id | int(foreign key)
product_id | int (foreign key)
quantity | int
OrderTable
ID | int (primary key)
customer_id | int (foreign key)
order_date | date
total_amount | double
Product
ID | int (primary key)
name | varchar
price | double
quantity | int
OrderItem
ID | int (primary key)
order_id | int (foreign key)
product_id | int (foreign key)
quantity | int

Results & Snapshots:



The provided JavaFX application, ShoppingCart, connects to an Oracle database using JDBC and allows users to enter their details (ID, name, and email) and select a product from a list. Upon clicking the "Add to Cart" button, the application inserts the customer details into the Customers table and creates a new entry in the ShoppingCart table, associating the customer with the selected product.


Shopping Cart

Enter Your Details:

ID: 8

Name: Jacob Thomas

Email: jt@gmail.com

Select Product:

User Interface (UI):

- The UI consists of text fields for entering customer details (ID, name, email), a list view for selecting products, and an "Add to Cart" button.
- Users can input their details and select a product from the list view.

Database Interaction:

- The application establishes a connection to the Oracle database using JDBC.
- It fetches the list of products from the Product table and populates the list view with product names.
- When the user clicks the "Add to Cart" button, the application inserts the customer details into the Customers table and creates a new entry in the ShoppingCart table, associating the customer with the selected product.
- If successful, a message dialog is displayed confirming the creation of the shopping cart entry with the assigned ID.

Error Handling:

- The application includes error handling to deal with potential exceptions during database operations or user input validation. If an error occurs, an error message dialog is displayed to the user.

Conclusion:

This mini project demonstrates the implementation of a basic shopping cart system using JavaFX and JDBC to interact with an Oracle database. Here's a conclusion about the project:

Functionality: The application allows users to input their details (ID, name, email) and select products from a list to add to their shopping cart. It performs database operations to insert customer details into the Customers table and create shopping cart entries in the ShoppingCart table, associating customers with selected products.

Database Interaction: JDBC is used to establish a connection to the Oracle database and execute SQL queries for fetching product data and inserting customer details and shopping cart entries. The application demonstrates how to handle SQL exceptions and close database resources properly to avoid memory leaks.

User Interface: JavaFX is used to create a simple and intuitive user interface with text fields for inputting customer details, a list view for selecting products, and a button for adding products to the shopping cart. Error messages are displayed in alert dialogs to provide feedback to the user.

Error Handling: The application includes error handling to deal with potential exceptions during database operations or user input validation. It ensures that users receive informative error messages in case of any issues.

Scalability and Extensibility: While this mini project provides a basic implementation of a shopping cart system, it can be extended and enhanced in various ways. Additional features such as product categories, quantity selection, user authentication, and order processing could be implemented to improve functionality and user experience.

In conclusion, this mini project serves as a starting point for building a more comprehensive and robust e-commerce application. It demonstrates fundamental concepts of JavaFX UI development and JDBC database interaction while providing a functional shopping cart system.

Limitations & future work:

Limited Functionality:

The current implementation only allows users to add products to their shopping cart. Additional features such as removing products, updating quantities, and processing orders could enhance the functionality of the application.

Basic User Interface:

The user interface is simplistic and lacks visual appeal. Future work could involve improving the UI design by adding icons, styling components, and organizing the layout more effectively to enhance user experience.

Database Schema Complexity:

The current database schema is relatively simple and may not fully capture the complexity of a real-world e-commerce application. Future work could involve expanding the schema to include more entities such as orders, order items, payment information, and user authentication.

Error Handling Enhancements:

While the application includes basic error handling, it could be enhanced to provide more informative error messages and handle edge cases more effectively. Additionally, error logging could be implemented to track and analyze errors for debugging purposes.

Scalability Issues:

The current implementation may face scalability issues when dealing with a large number of users and products. Future work could involve optimizing database queries, implementing caching mechanisms, and scaling the application architecture to handle increased traffic and data volume.

Security Considerations:

Security aspects such as input validation, data encryption, and protection against SQL injection attacks are not addressed in the current implementation. Future work should include implementing security measures to safeguard user data and prevent unauthorized access.

Localization and Internationalization:

The application currently lacks support for localization and internationalization. Future work could involve adding support for multiple languages and currencies to cater to a global user base.

Testing and Quality Assurance:

The mini project may benefit from comprehensive testing to ensure robustness, reliability, and adherence to requirements. Automated unit tests, integration tests, and user acceptance tests could be implemented to verify the correctness and functionality of the application.

References:

1. JavaFX Documentation:
    - https://openjfx.io/javadoc/17/
2. Oracle JDBC Documentation:
    - https://docs.oracle.com/en/java/javase/17/docs/api/java.sql/java/sql/package-summary.html
3. Oracle Database Documentation:
    - https://docs.oracle.com/en/database/oracle/oracle-database/21/index.html
4. Stack Overflow:
    - Various posts related to JavaFX, JDBC, and Oracle database interactions.
5. Programming books:
    - Database System Concepts by Silberschatz, Henry F. Korth, and S. Sudarshan (6th Edition)