**TSDL-JAVA ASSIGNMENT-6 (JDBC Connectivity)**

|  |  |  |
| --- | --- | --- |
| **SRN** | **NAME** | **ROLL NO** |
| 202202232 | Alfiya Sayyed | 58 |

***TOPIC-9 Online Shopping Management***

**Introduction:**

**Online shopping has revolutionized the retail landscape, offering** convenience and accessibility to consumers worldwide. To facilitate seamless transactions and efficient management, the development of an online shopping management system is imperative. This system aims to streamline user interactions, manage inventory, and ensure secure transactions, thereby enhancing the overall shopping experience. This project demonstrates the implementation of such a system using Java programming language and MySQL database.

**Methodology**

The methodology reflects the approach taken to design and implement the online shopping management system.

1. Requirements Analysis:

The system requirements include user authentication, item management, shopping cart functionality, and secure database handling.

This was translated into specific functionalities such as user signup and signin, item addition, browsing, and shopping cart management.

2. Database Design:

MySQL is utilized for database management.

Tables for users and items are designed, with appropriate fields for storing user credentials and item details.

The DatabaseHandler class encapsulates database interactions, including user signup and signin operations.

3. Software Implementation:

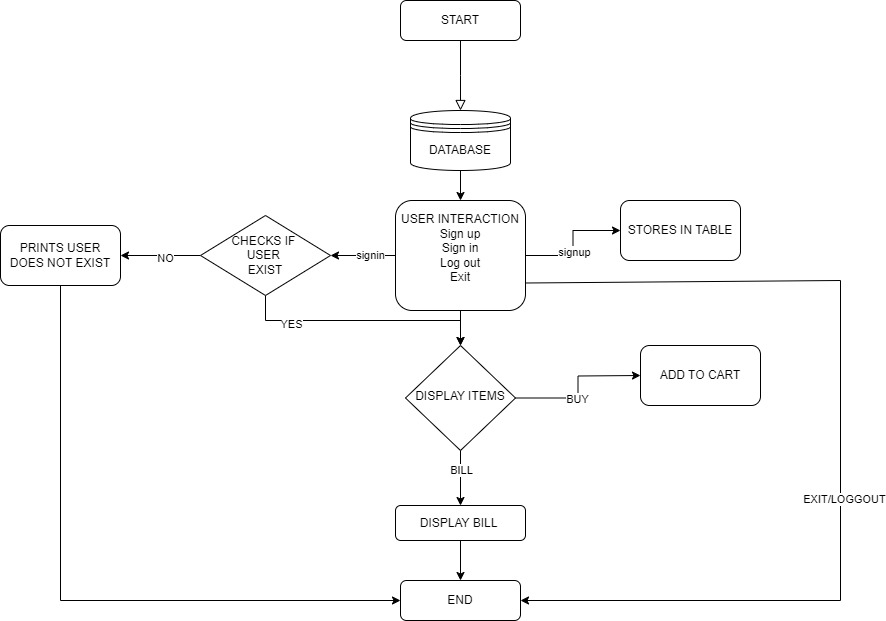
Java is chosen for system implementation.

Object-oriented principles are applied to design classes for users (User), items (Item and Stationary), and shopping cart management (ShoppingCart).

User interactions are handled through the DBDemo class, which serves as the entry point for the application.

Database queries are executed using JDBC within appropriate methods such as signupUser() and signinUser().

**Flowchart:**

****

**Implementation**:

The implementation section details the functionalities of the system and explains how they are realized in the provided code.

1. User Authentication:

Users can sign up (signupUser()) or sign in (signinUser()) using a username and password.

Passwords are securely stored and validated against the database.

2. Item Management:

Administrators can add items to the database through the system interface.

Item details (name, price, attribute) are stored in the database and retrieved when browsing available items (displayAvailableItems()).

3. Shopping Cart:

Users can add items to their shopping carts (addItem()), view cart contents (displayItems()), and calculate the total price (getTotalPrice()).

Item addition triggers database updates through the saveItem() method.

1. Inheritance:

Inheritance is demonstrated through the relationship between the Item class and the Stationary class. Stationary inherits from Item, which means it inherits the properties and behavior defined in Item. This allows Stationary to reuse the name and price fields and override the displayDetails() method with its own implementation.

2. Polymorphism (Method Overriding):

Polymorphism is showcased through method overriding in Java. The displayDetails() method in the Item class is declared as abstract, meaning it must be implemented by any subclass of Item. In the Stationary class, this method is overridden to provide specific behavior for displaying details of stationary items.

3. Encapsulation:

Encapsulation is evident in the way data hiding and abstraction are implemented. For example, the fields name and price in the Item class are declared as protected, allowing access within the class and its subclasses but not directly from outside. Similarly, the User class encapsulates the user's information (username and password) within private fields and provides getter methods for controlled access.

4. Exception Handling:

Exception handling is employed to deal with potential errors that may occur during database operations. Methods like getConnection(), signupUser(), signinUser(), displayAvailableItems(), and getItemFromDatabase() all handle SQLExceptions that could be thrown during database interactions. They catch these exceptions and provide error handling, such as printing stack traces or returning appropriate values.

5. Database Connectivity (JDBC):

The code utilizes Java Database Connectivity (JDBC) to interact with a MySQL database. It establishes a connection to the database using the DriverManager.getConnection() method, executes SQL queries using PreparedStatement, and processes query results using ResultSet. Database-related operations like saving items, signing up users, signing in users, and retrieving items from the database are performed using JDBC.

6. Constructor:

When the name of class and function is Same its called as Constructor,

here in Item class we defined item() function same for Stationary class .

7. Abstraction:

We have defined abstract class Item and in it we made a function called displayDetails() which is abstract meaning it will show important details and hide usless info.

**CODE:**

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

abstract class Item {

protected String name;

protected double price;

public Item(String name, double price) {

this.name = name;

this.price = price;

}

public abstract void displayDetails();

}

class Stationary extends Item {

private String attribute;

public Stationary(String name, double price, String attribute) {

super(name, price);

this.attribute = attribute;

}

@Override

public void displayDetails() {

System.out.println("Name: " + name);

System.out.println("Price: $" + price);

System.out.println("Attribute: " + attribute);

}

}

class User {

private String username;

private String password;

public User(String username, String password) {

this.username = username;

this.password = password;

}

public String getUsername() {

return username;

}

public String getPassword() {

return password;

}

}

class DatabaseHandler {

private static final String DB\_URL = "jdbc:mysql://localhost:3306/shopping\_db";

private static final String DB\_USER = "root";

private static final String DB\_PASSWORD = "avani2004#$#";

public static Connection getConnection() throws SQLException {

return DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);

}

public static boolean saveItem(Item item) {

// Implement saving item to the database

return true; // Placeholder for now

}

public static boolean signupUser(User user) {

String query = "INSERT INTO users (username, password) VALUES (?, ?)";

try (Connection conn = getConnection();

PreparedStatement statement = conn.prepareStatement(query)) {

statement.setString(1, user.getUsername());

statement.setString(2, user.getPassword());

int rowsAffected = statement.executeUpdate();

return rowsAffected > 0;

} catch (SQLException e) {

e.printStackTrace();

return false;

}

}

public static User signinUser(String username, String password) {

String query = "SELECT \* FROM users WHERE username = ? AND password = ?";

try (Connection conn = getConnection();

PreparedStatement statement = conn.prepareStatement(query)) {

statement.setString(1, username);

statement.setString(2, password);

try (ResultSet resultSet = statement.executeQuery()) {

if (resultSet.next()) {

return new User(username, password);

}

}

} catch (SQLException e) {

e.printStackTrace();

}

return null;

}

}

class ShoppingCart {

private List<Item> items;

public ShoppingCart() {

items = new ArrayList<>();

}

public void addItem(Item item) {

items.add(item);

// Save item to the database

if (DatabaseHandler.saveItem(item)) {

System.out.println("Item added successfully to the database.");

} else {

System.out.println("Failed to add item to the database.");

}

}

public double getTotalPrice() {

double total = 0;

for (Item item : items) {

total += item.price;

}

return total;

}

public void displayItems() {

for (Item item : items) {

item.displayDetails();

System.out.println();

}

}

}

public class DBDemo {

public static void main(String[] args) {

try (Connection conn = DatabaseHandler.getConnection()) {

System.out.println("Connected to database");

// Your application logic here

Scanner scanner = new Scanner(System.in);

ShoppingCart cart = new ShoppingCart();

User currentUser = null;

while (true) {

System.out.println("1. Signup");

System.out.println("2. Signin");

System.out.println("3. Logout");

System.out.println("4. Exit");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1:

System.out.print("Enter username: ");

String signupUsername = scanner.nextLine();

System.out.print("Enter password: ");

String signupPassword = scanner.nextLine();

if (DatabaseHandler.signupUser(new User(signupUsername, signupPassword))) {

System.out.println("Signup successful.");

} else {

System.out.println("Signup failed.");

}

break;

case 2:

System.out.print("Enter username: ");

String signinUsername = scanner.nextLine();

System.out.print("Enter password: ");

String signinPassword = scanner.nextLine();

currentUser = DatabaseHandler.signinUser(signinUsername, signinPassword);

if (currentUser != null) {

System.out.println("Signin successful.");

} else {

System.out.println("Invalid username or password.");

}

break;

case 3:

currentUser = null;

System.out.println("Logged out.");

break;

case 4:

System.out.println("Exiting...");

return;

default:

System.out.println("Invalid choice.");

}

if (currentUser != null) {

// Display available items

displayAvailableItems(conn);

String itemName;

while (true) {

// Prompt user to choose an item

System.out.print("Enter the name of the item you want to buy (Enter 'BILL' to finish): ");

itemName = scanner.nextLine();

if (itemName.equalsIgnoreCase("BILL")) {

break;

}

// Retrieve item details from the database

Item selectedItem = getItemFromDatabase(conn, itemName);

// Check if the item exists

if (selectedItem != null) {

// Add the selected item to the cart

cart.addItem(selectedItem);

} else {

System.out.println("Item not found.");

}

}

// Display items in the cart

System.out.println("Items in the shopping cart:");

cart.displayItems();

// Display total price

System.out.println("Total price: $" + cart.getTotalPrice());

}

}

} catch (SQLException e) {

System.out.println("ERROR: Could not connect to the database");

e.printStackTrace();

}

}

private static void displayAvailableItems(Connection conn) throws SQLException {

System.out.println("Available items:");

String query = "SELECT \* FROM items";

try (PreparedStatement statement = conn.prepareStatement(query);

ResultSet resultSet = statement.executeQuery()) {

while (resultSet.next()) {

String name = resultSet.getString("name");

double price = resultSet.getDouble("price");

String attribute = resultSet.getString("attribute");

System.out.println("Name: " + name + ", Price: $" + price + ", Attribute: " + attribute);

}

}

}

private static Item getItemFromDatabase(Connection conn, String itemName) throws SQLException {

String query = "SELECT \* FROM items WHERE name = ?";

try (PreparedStatement statement = conn.prepareStatement(query)) {

statement.setString(1, itemName);

try (ResultSet resultSet = statement.executeQuery()) {

if (resultSet.next()) {

String name = resultSet.getString("name");

double price = resultSet.getDouble("price");

String attribute = resultSet.getString("attribute");

return new Stationary(name, price, attribute);

}

}

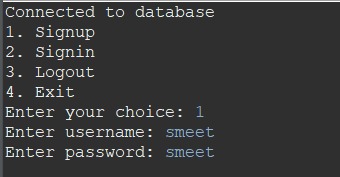
}

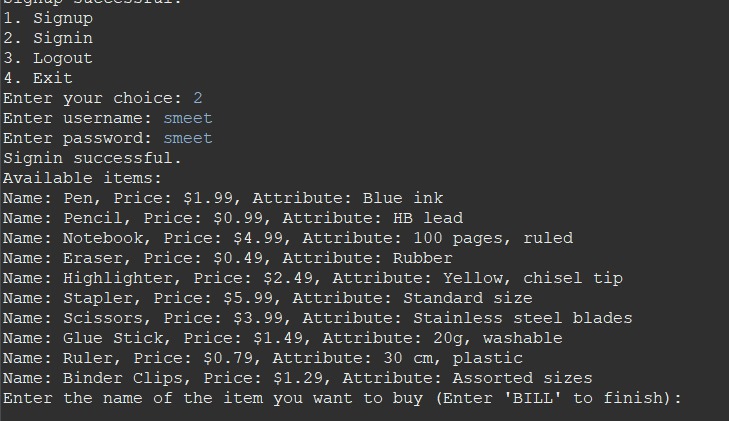
return null; // Item not found

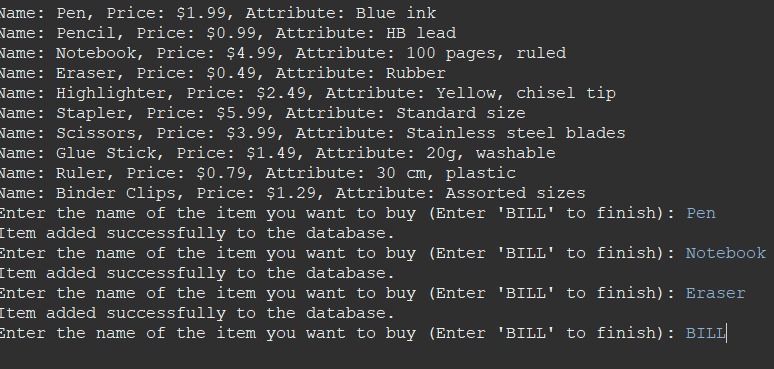
}

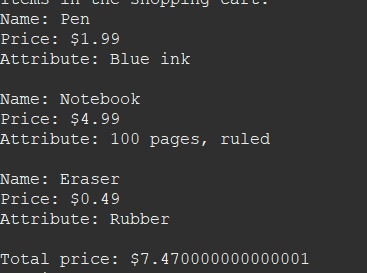
}

**OUTPUT:**









**SQL:**

create database shopping\_db;

use shopping\_db;

CREATE TABLE items (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

price DECIMAL(10, 2) NOT NULL,

type VARCHAR(50) NOT NULL,

brand VARCHAR(100),

size VARCHAR(50),

attribute VARCHAR(255)

);

INSERT INTO items (name, price, type, attribute)

VALUES

('Pen', 1.99, 'Stationary', 'Blue ink'),

('Pencil', 0.99, 'Stationary', 'HB lead'),

('Notebook', 4.99, 'Stationary', '100 pages, ruled'),

('Eraser', 0.49, 'Stationary', 'Rubber'),

('Highlighter', 2.49, 'Stationary', 'Yellow, chisel tip'),

('Stapler', 5.99, 'Stationary', 'Standard size'),

('Scissors', 3.99, 'Stationary', 'Stainless steel blades'),

('Glue Stick', 1.49, 'Stationary', '20g, washable'),

('Ruler', 0.79, 'Stationary', '30 cm, plastic'),

('Binder Clips', 1.29, 'Stationary', 'Assorted sizes');

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(255) NOT NULL,

password VARCHAR(255) NOT NULL

);

**Conclusion:**

In conclusion, the development of the online shopping management system showcases the successful integration of Java programming language and MySQL database to create a robust and user-friendly platform. Through meticulous requirements analysis, database design, and software implementation, the system now offers essential functionalities such as user authentication, item management, and shopping cart operations. By securely storing user credentials and item details, the system ensures data integrity and confidentiality, enhancing trust among users. Looking ahead, potential future enhancements including integration with payment gateways, implementation of recommendation systems, and database performance optimization could further elevate the user experience and expand the system's capabilities. Overall, this project represents a significant step towards revolutionizing online retail operations and meeting the evolving needs of consumers in the digital age.