BuyBot - E-Commerce CLI App (Java + MySQL)

1. Project Structure

```
BuyBot/
├── src/
│   └── BuyBot.java
├── sql/
│   └── schema.sql
├── .gitignore
└── README.md
```

2. src/BuyBot.java

```
import java.sql.*;
import java.util.*;

public class BuyBot {
    static final String DB_URL =
"jdbc:mysql://localhost:3306/ecommerce";
    static final String USER = "root";
    static final String PASS = "your_password";  // Replace with your
actual DB password

    static Scanner scanner = new Scanner(System.in);
    static Connection conn;
    static int currentUserId = -1;

    public static void main(String[] args) {
        try {
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            while (true) {
                System.out.println("\n1. Register\n2. Login\n3.
Exit");
                int choice = scanner.nextInt();
                scanner.nextLine();
                switch (choice) {
                    case 1 -> register();
                    case 2 -> login();
                    case 3 -> {
                        conn.close();
                        return;
                    }
                    default -> System.out.println("Invalid choice.");
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    static void register() throws SQLException {
        System.out.print("Username: ");
        String username = scanner.nextLine();
        System.out.print("Password: ");
        String password = scanner.nextLine();

        String sql = "INSERT INTO users (username, password) VALUES
(?, ?)";
        try (PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setString(1, username);
            ps.setString(2, password);
            ps.executeUpdate();
            System.out.println("Registration successful.");
        } catch (SQLIntegrityConstraintViolationException e) {
            System.out.println("Username already exists.");
        }
    }

    static void login() throws SQLException {
        System.out.print("Username: ");
        String username = scanner.nextLine();
        System.out.print("Password: ");
        String password = scanner.nextLine();

        String sql = "SELECT id FROM users WHERE username = ? AND
password = ?";
        try (PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setString(1, username);
            ps.setString(2, password);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                currentUserId = rs.getInt("id");
                System.out.println("Login successful.");
                userMenu();
            } else {
                System.out.println("Invalid credentials.");
            }
        }
    }
    static void userMenu() throws SQLException {
        while (true) {
            System.out.println("\n1. View Products\n2. Buy Product\n3.
View Orders\n4. Logout");
            int choice = scanner.nextInt();
            switch (choice) {
                case 1 -> viewProducts();
                case 2 -> buyProduct();
                case 3 -> viewOrders();
                case 4 -> {
                    currentUserId = -1;
                    return;
                }
                default -> System.out.println("Invalid choice.");
            }
        }
    }

    static void viewProducts() throws SQLException {
        String sql = "SELECT * FROM products";
        try (Statement stmt = conn.createStatement(); ResultSet rs =
stmt.executeQuery(sql)) {
            while (rs.next()) {
                System.out.printf("ID: %d | %s | Rs. %.2f | Stock:
%d\n",
                        rs.getInt("id"), rs.getString("name"),
                        rs.getDouble("price"), rs.getInt("quantity"));
            }
        }
    }

    static void buyProduct() throws SQLException {
        System.out.print("Enter Product ID: ");
        int pid = scanner.nextInt();
        System.out.print("Enter quantity: ");
        int qty = scanner.nextInt();

        String checkStock = "SELECT quantity FROM products WHERE id =
?";
        try (PreparedStatement ps = conn.prepareStatement(checkStock))
{
            ps.setInt(1, pid);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                int stock = rs.getInt("quantity");
                if (stock >= qty) {
                    String updateStock = "UPDATE products SET quantity
= quantity - ? WHERE id = ?";
                    try (PreparedStatement ups =
conn.prepareStatement(updateStock)) {
                        ups.setInt(1, qty);
                        ups.setInt(2, pid);
                        ups.executeUpdate();
                    }

                    String insertOrder = "INSERT INTO orders (user_id,
product_id, quantity) VALUES (?, ?, ?)";
                    try (PreparedStatement ins =
conn.prepareStatement(insertOrder)) {
                        ins.setInt(1, currentUserId);
                        ins.setInt(2, pid);
                        ins.setInt(3, qty);
                        ins.executeUpdate();
                    }

                    System.out.println("Purchase successful!");
                } else {
                    System.out.println("Not enough stock available.");
                }
            } else {
                System.out.println("Product not found.");
            }
        }
    }

    static void viewOrders() throws SQLException {
        String sql = """
            SELECT p.name, o.quantity, p.price, o.quantity * p.price
AS total
            FROM orders o
            JOIN products p ON o.product_id = p.id
            WHERE o.user_id = ?
            """;
        try (PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setInt(1, currentUserId);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                System.out.printf("Product: %s | Qty: %d | Total: Rs.
%.2f\n",
                        rs.getString("name"),
                        rs.getInt("quantity"),
                        rs.getDouble("total"));
            }
        }
    }
```

```
}
```

3. sql/schema.sql

```
CREATE DATABASE IF NOT EXISTS ecommerce;
USE ecommerce;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(50) NOT NULL
);

CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    price DECIMAL(10, 2),
    quantity INT
);

CREATE TABLE orders (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    product_id INT,
    quantity INT,
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);

INSERT INTO products (name, price, quantity) VALUES
('Laptop', 75000.00, 10),
('Smartphone', 25000.00, 20),
('Headphones', 1500.00, 50);
```

4. .gitignore

```
*.class
*.log
*.jar
.idea/
*.iml
out/
```

5. README.md

```
# BuyBot - E-Commerce CLI App (Java + MySQL)

## Features:
- User registration and login
- Product Browse
- Purchase products (inventory update)
- Order history tracking

## Technologies:
- Java
- MySQL
- JDBC
```

Technologies Used:
 * Java:
   * What it is: A high-level, class-based, object-oriented programming language designed to have as few implementation dependencies as possible. It's known for its "Write Once, Run Anywhere" (WORA) philosophy.
   * How it's used here: The core application logic, user interface (command-line interface), and database interactions are all implemented in Java.
 * MySQL:
   * What it is: A popular open-source relational database management system (RDBMS). It's widely used for web applications due to its speed, reliability, and ease of use.
   * How it's used here: MySQL stores all the application data, including user accounts, product information, and order details.
 * JDBC (Java Database Connectivity):
   * What it is: A Java API that defines how a Java client can connect to and interact with a database. It provides a standard way to query and update data.
   * How it's used here: JDBC is the bridge between the Java application and the MySQL database. It allows BuyBot to execute SQL queries (e.g., INSERT, SELECT, UPDATE) and retrieve results from the database.