

CS-303 Assignment-3

Josyula V N Taraka Abhishek - 200010021

JP Karthik - 200010022

October 2022

Contents

1	Q1) Design a database for an automobile company	2
1.1	schema	2
2	Q2) Design a database for a world-wide package delivery company	3
2.1	schema	3
3	Q3) Consider a carelessly written Web application	4
4	Q4) Consider another carelessly written Web application	4
5	Q5) Hackers may be able to fool you	4
6	Q6) Write a servlet and associated HTML for sessions	5
6.1	Index.jsp	5
6.2	LoginServlet.java	6
6.3	Results	7
7	Q7) What is an XSS attack?	9

List of Figures

1	ER model for auto company	2
2	ER model for package delivery company	3
3	Q6) Login	7
4	Q6) Response session	8
5	Q6) User doesnot exist or wrong password	8
6	Q6) LoginFail	9

1 Q1) Design a database for an automobile company

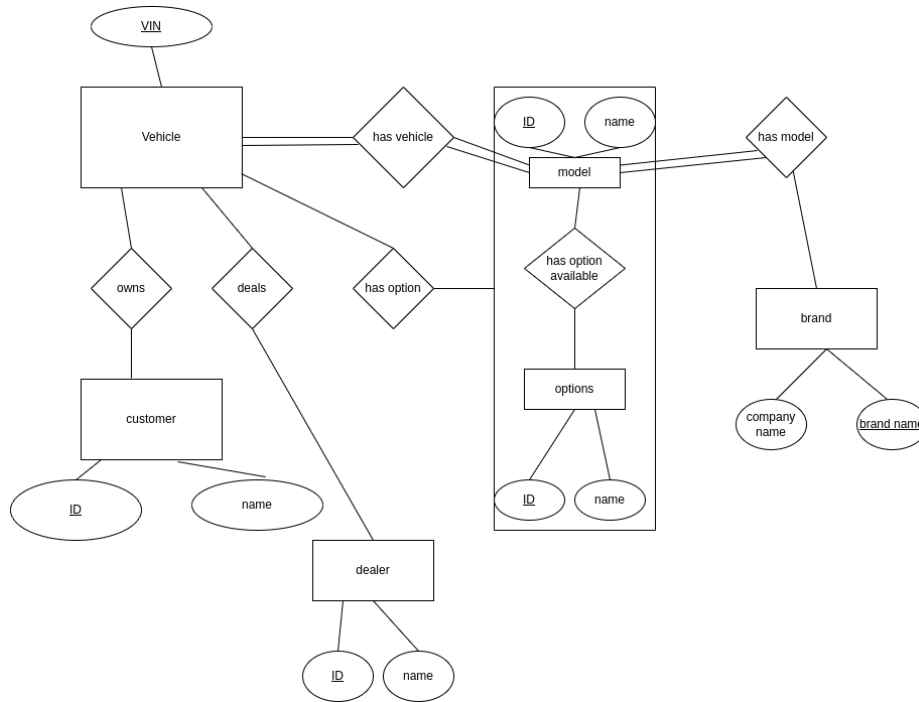


Figure 1: ER model for auto company

1.1 schema

1. brand(brand name),
2. model(model id, model name)
3. vehicle(VIN, dealer id, customer id)
4. option(option id, specification)
5. customer(customer id, customer name, address)
6. dealer(dealer id, dealer name, address)
7. has model(brand name, model id , foreign key brand name references brand, foreign key model id references model)
8. has vehicle(model id, VIN, foreign key VIN references vehicle, foreign key model id references model)

9. has available option(model id, option id, foreign key option id references option, foreign key model id references model)
10. has option(VIN, model id, option id, foreign key VIN references vehicle, foreign key (model id, option id) references available option)
11. has dealer(VIN, dealer id , foreign key dealer id references dealer, foreign key VIN references vehicle)
12. owned by(VIN, customer id, foreign key customer id references customer, foreign key VIN references vehicle)

2 Q2) Design a database for a world-wide package delivery company

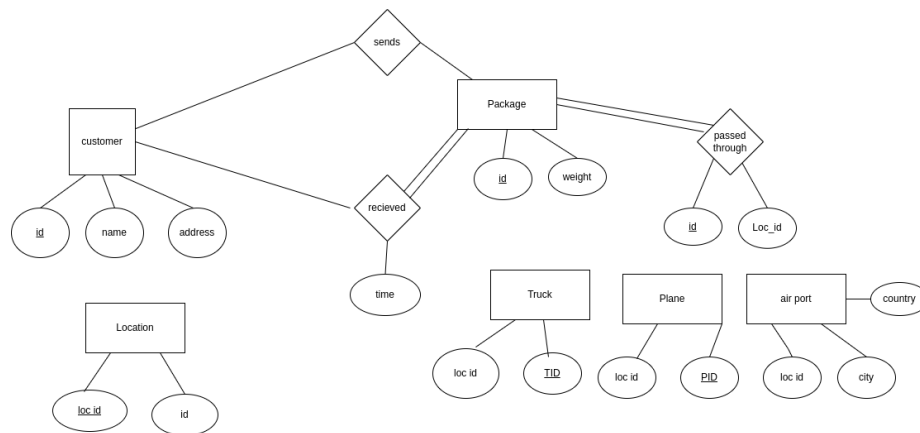


Figure 2: ER model for package delivery company

2.1 schema

1. customer(customer id, customer name, address)
2. package(package id, weight, contents)
3. location(loc id, package_id)
4. (foreign key package id references package)
5. truck(loc id, T_ID)
6. plane(loc id, P_ID)
7. airport(loc id, city, country)

8. warehouse(loc id, address)
9. at(package id, loc id, time in, time out)
10. passed_through(package_id, loc_id)
11. (foreign key package id references package, foreign key loc id references location)
12. receive(customer id, package id, time,)
13. (foreign key customer id references customer, foreign key package id references package)
14. send(customer id, package id, time)
15. (foreign key customer id references customer, foreign key package id references package)

3 Q3) Consider a carelessly written Web application

When the customer uses browser tools(F12 key) and changes the value of the hidden variable to a favourable value like 1 rupee instead of 1000 rupee. The app places order based on this favourable value and place the order. The web-app will use this favourable value, bill the user and place order.

4 Q4) Consider another carelessly written Web application

1. Even if the link is only shown to authorized users, unauthorized users can get the link from guess or authorized users etc
2. The unauthorized user can then use link to access the page. If user authorization is left out for this page the unauthorized user can get the result of this page.

5 Q5) Hackers may be able to fool you

1. In HTTPS website first sends a digital certificate to the users browser.
2. The browser decrypts the digital certificate using stored public key of trusted certificate authority and displays and displays site name(Here my-bank.com). And user can decide if the name matches.

3. HTTPS uses the conventional HTTP protocol and adds a layer of SSL/TSL over it. The SSL connection is responsible for the encryption and decryption of the data that is being exchanged in order to ensure data safety.
4. HTTPS transfers data in an encrypted format. As a result, HTTPS protects websites from having their information broadcast in a way that anyone eavesdropping on the network can easily see.
5. Malicious users will not be able to get access to our messages because they donot have access to the public key.

6 Q6) Write a servlet and associated HTML for sessions

6.1 Index.jsp

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>Welcome to simple Application!!!!</h1>

Login with your UserName and Password!
<form action="LoginServlet" method="post">
    <table style="width: 50%">
        <tr>
            <td>User Id</td>
            <td><input type="text" name="u_id" /></td>
        </tr>
        <tr>
            <td>Password</td>
            <td><input type="password" name="password" /></td>
        </tr>
    </table>
    <input type="submit" value="Submit" /></form>

</body>
</html>

```

6.2 LoginServlet.java

```
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    // TODO Auto-generated method stub

    // Get parameters
    String user_id = request.getParameter("u_id");
    String user_password = request.getParameter("password");

    Connection con = null;
    String url = "jdbc:mysql://localhost:3306/UserBase"; //MySQL URL
        and followed by the database name
    String username = "universityDB0021"; //MySQL username
    String password = "MysecretPass"; //MySQL password

    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }

    try {
        con = DriverManager.getConnection(url, username, password);
    } catch (SQLException e) {
        e.printStackTrace();
    }

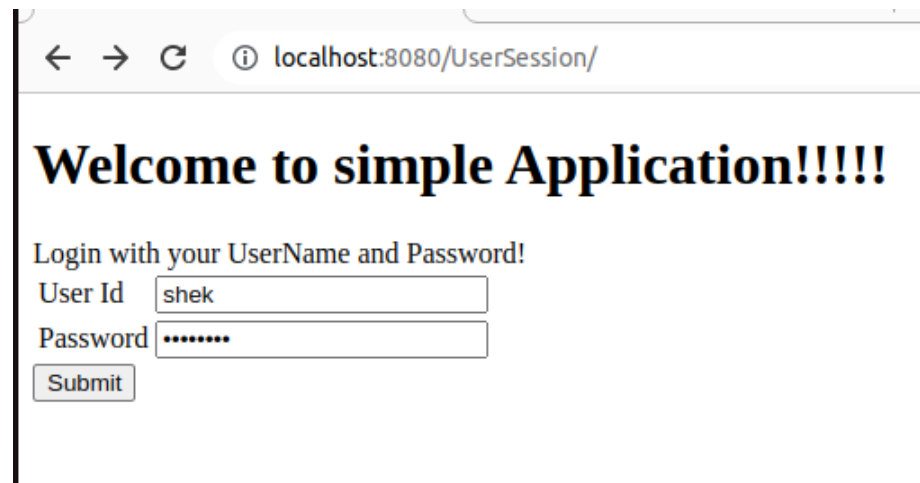
    try {
        PreparedStatement chk_pwd = con.prepareStatement("SELECT
            count(*) FROM user WHERE user.id=? and user.password=?");
        chk_pwd.setString(1, user_id);
        chk_pwd.setString(2, user_password);
        ResultSet result = chk_pwd.executeQuery();
        result.next();
        int count = Integer.parseInt(result.getString(1));

        if (count == 1) {
            response.setContentType("text/html");
            PrintWriter pwriter = response.getWriter();

            HttpSession session = request.getSession();
            session.setAttribute("user_name", user_id);
            session.setAttribute("user_password", password);
            pwriter.print("<h1>Welcome to your session " + user_id +
                "</h1>");
            pwriter.print("<a>Your session id is
                "+session.getId()+"</a>");
            pwriter.close();
        }
    }
}
```

```
    }else {  
        response.setContentType("text/html");  
        PrintWriter pwriter = response.getWriter();  
        pwriter.print("<h1>Failed to authenticate you!!!</h1>");  
        pwriter.close();  
    }  
  
    } catch (SQLException e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
    }  
}
```

6.3 Results



The screenshot shows a web browser window with the address bar displaying "localhost:8080/UserSession/". The main content of the page is a login form. At the top, it says "Welcome to simple Application!!!!" in a large, bold, black serif font. Below this, it says "Login with your UserName and Password!" in a smaller, regular black serif font. There are two input fields: "User Id" with the text "shek" entered, and "Password" with "*****" entered. Below these fields is a "Submit" button.

Figure 3: Q6) Login

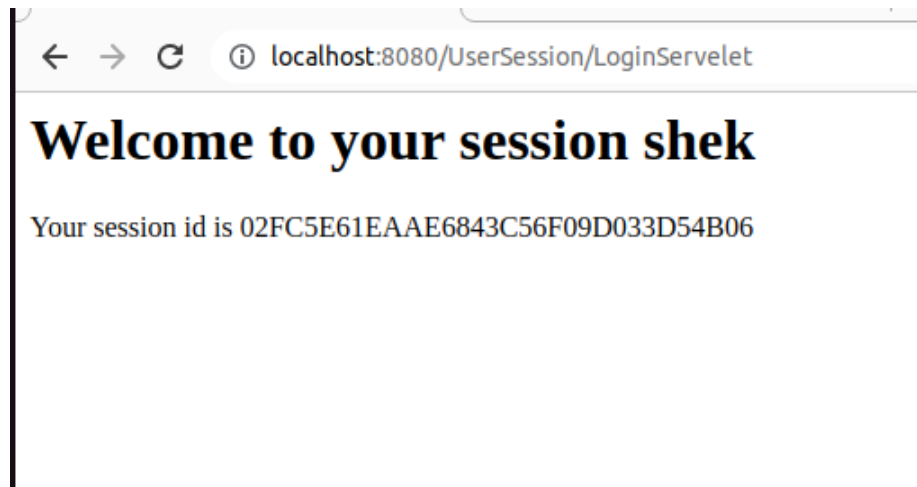


Figure 4: Q6) Response session

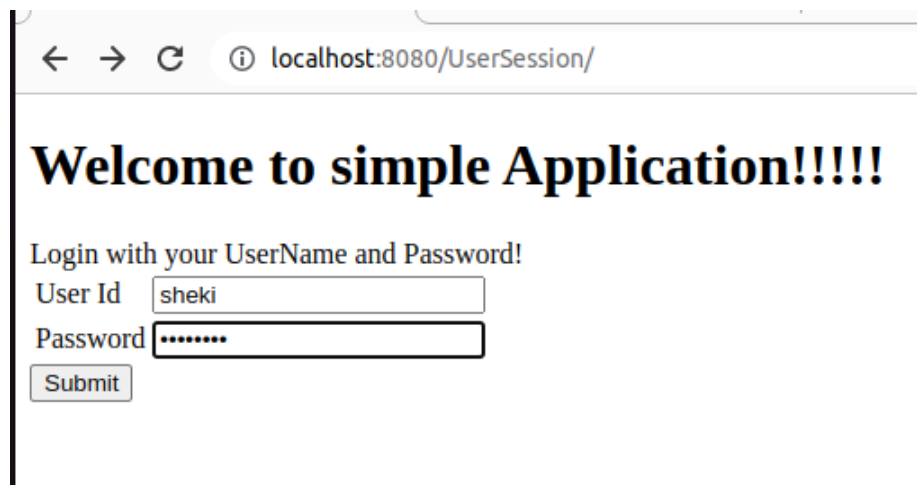


Figure 5: Q6) User doesnot exist or wrong password

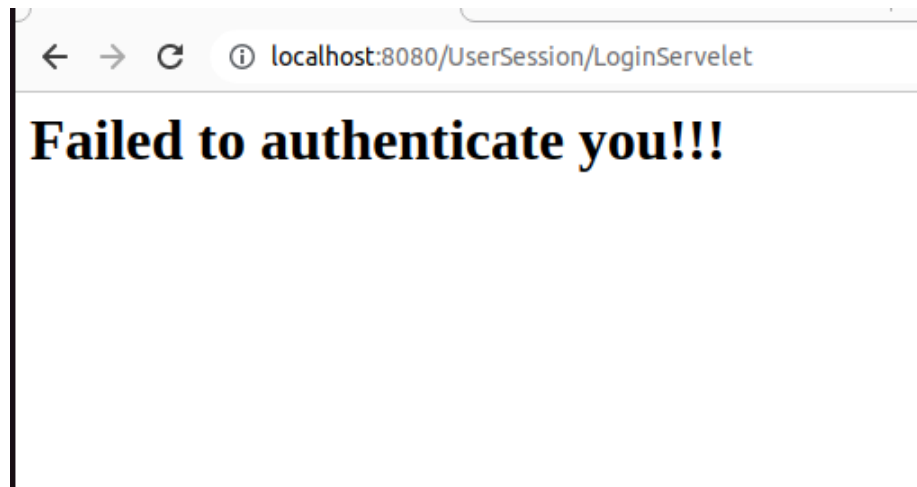


Figure 6: Q6) LoginFail

7 Q7) What is an XSS attack?

1. Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into trusted websites.
2. A malicious user enters code written in browser based scripting language like JS instead of proper comment and the browser of innocent user executes this code.
3. When the innocent user views this comment the script executes and steals private information from user like passwords or cookies etc.