CS303 : DataBases and Information Systems

Assignment 3

Sourabh Bhosale
200010004

October 18, 2022

1 Problem 1

Design a database for an automobile company to provide to its dealers to assist them in maintaining customer records and dealer inventory and to assist sales staff in ordering cars.

Given Information: Each vehicle is identified by a vehicle identification number (VIN). Each individual vehicle is a particular model of a particular brand offered by the company (e.g., the XF is a model of the car brand Jaguar of Tata Motors). Each model can be offered with a variety of options, but an individual car may have only some (or none) of the available options. The database needs to store information about models, brands, and options, as well as information about individual dealers, customers, and cars.
Your design should include an E-R diagram, a set of relational schemas, and a list of constraints, including primary-key and foreign-key constraints

(a) E-R Diagram

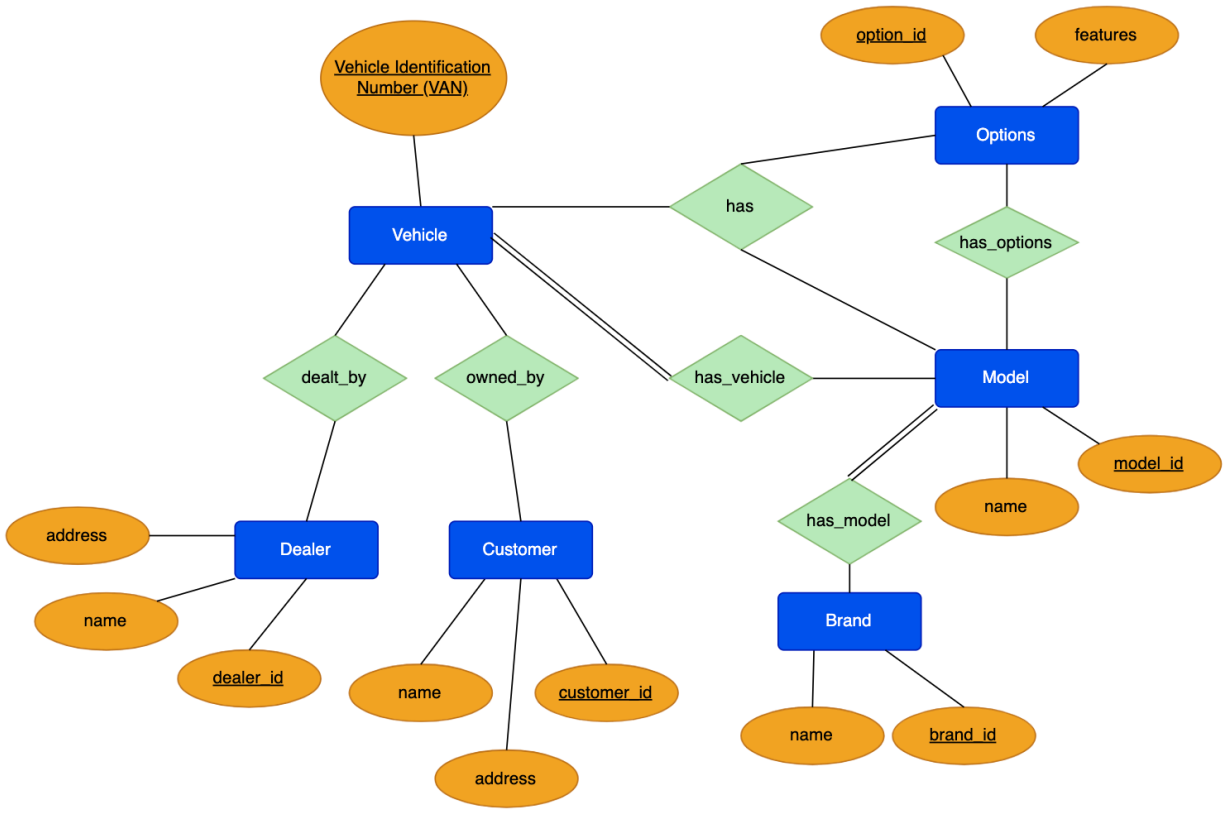

Figure 1: E-R Diagram

(b) Relational schemas

```
brand(name, brand_id)
model(name, model_id)
vehicle(VIN)
option(option_id, features)
customer(customer_id, name, address)
dealer(dealer_id, name, address)
has_model(brand_id, model_id,
    foreign key brand_id references brand,
    foreign key model_id references model)
has_vehicle(model_id, VIN,
    foreign key VIN references vehicle,
    foreign key model_id references model)
has_options(model_id, option_id,
    foreign key option_id references option,
    foreign key model_id references model)
has(VIN, model_id, option_id,
    foreign key VIN references vehicle,
    foreign key (model_id, option_id) references has_options)
dealt_by(VIN, dealer_id,
    foreign key dealer_id references dealer,
    foreign key VIN references vehicle)
owned_by(VIN, customer_id,
    foreign key customer_id references customer,
    foreign key VIN references vehicle)
```

Assumption:

- For handling the given conditon, i.e. each model can be offered with a variety of options, but an individual car may have only some (or none) of the available options we have made relationship **has_vehicles** between Model and Vehicle, **has_options** between Model and Options to ensure model has various options and **has** between Vehicle and (Options, Model) considering second quantity as single entity to make sure that individual car may have some available options.
- An alternate design could be made by connecting the has relation to an aggregated entity, formed by "joining" the tables of Model, has_options, and Options. And then we would connect has only to vehicle and has_options.

(c) Table of constraints

Table	Primary Key	Domain of P.K.	Foreign keys (Referencing table)	Not Null
c Brand	brand_id	VARCHAR	-	-
Model	model_id	VARCHAR	-	-
Vehicle	VIN	VARCHAR	-	-
Options	option_id	VARCHAR	-	-
Customer	customer_id	VARCHAR	-	-
Dealer	dealer_id	VARCHAR	-	-
has_model	model_id brand_id	VARCHAR VARCHAR	model_id references Model brand_id references Brand	-
has_vehicle	model_id VIN	VARCHAR VARCHAR	model_id references Model VIN references Vehicle	-
has_options	model_id option_id	VARCHAR VARCHAR	model_id references Model option_id references Options	-
has	VIN model_id option_id	VARCHAR VARCHAR VARCHAR	VIN references Vehicle (model_id,option_id) references has_options	-
dealt_by	VIN dealer_id	VARCHAR VARCHAR	VIN references Vehicle dealer_id references Dealer	-
owned_by	VIN customer_id	VARCHAR VARCHAR	VIN references Vehicle customer_id references Customer	-

Table 1: Table of constraints

2 Problem 2

Design a database for a world-wide package delivery company (e.g., DHL or Fed EX).

Given Information: The database must be able to keep track of customers (who ship items) and customers (who receive items); some customers may do both. Each package must be identifiable and trackable, so the database must be able to store the location of the package and its history of locations.
Locations include trucks, planes, airports, and warehouses. Your design should include an E-R diagram, a set of relational schemas, and a list of constraints, including primary-key and foreign-key constraints.

(a) E-R Diagram

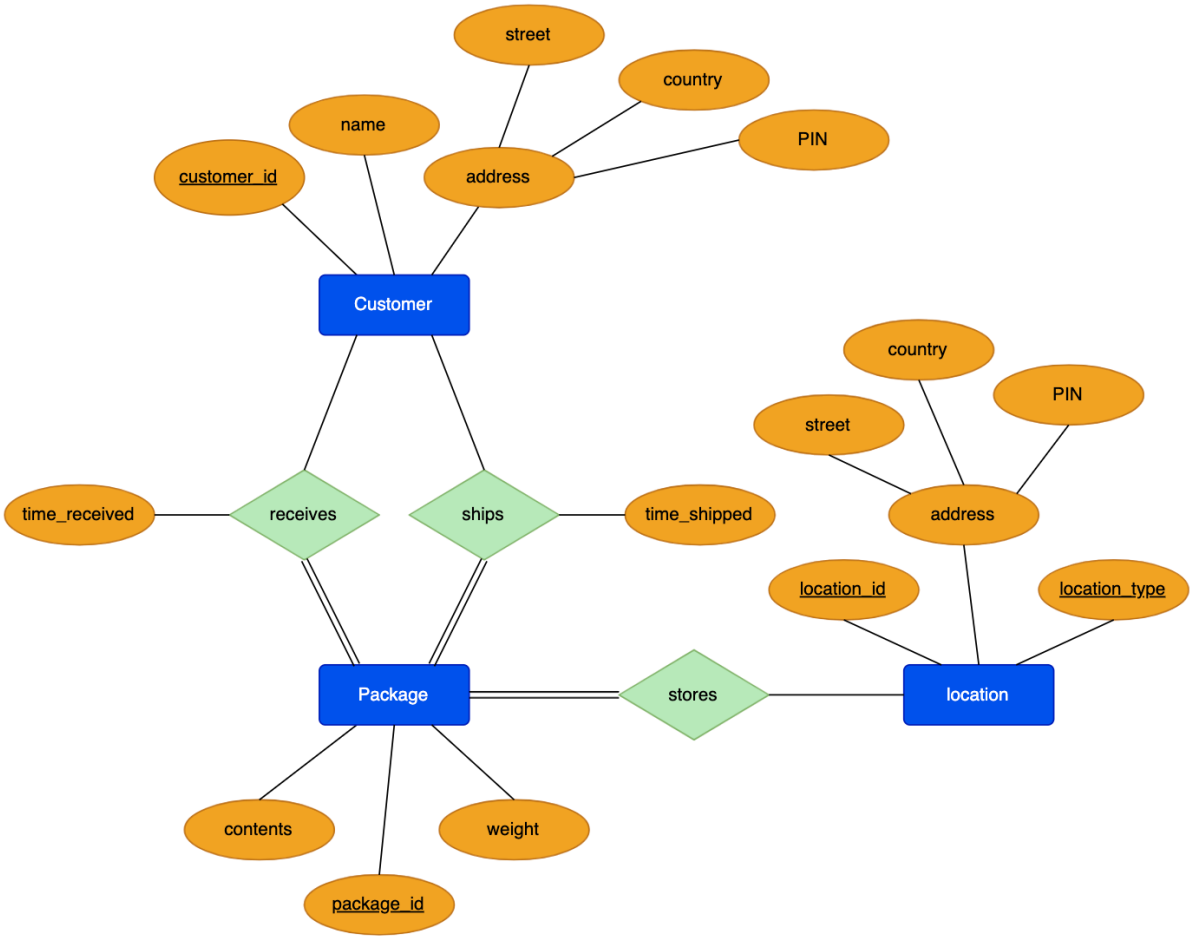

Figure 2: E-R Diagram

(b) Relational schemas

```
customer(customer_id, name, street, country, PIN)
package(package_id, weight, contents)
location(location_id, location_type, street, country, PIN)
stores(package_id, location_id, location_type
    foreign key package_id references package,
    foreign key location_id references location,
    foreign key location_type references location)
receives(customer_id, package_id, time_received,
    foreign key customer_id references customer,
    foreign key package_id references package)
ships(customer_id, package_id, time_shipped,
    foreign key customer_id references customer,
    foreign key package_id references package)
```

Assumption:

- Here to handle the given condition, i.e. Locations include trucks, planes, airports, and warehouses, we made location_id to identify each location history uniquely and location_type to ensure we include trucks, planes, airports, and warehouses. We could have also ellaborated location_type into futher subtypes such as transportation_type and storage_type. But here location_type along with location_id handles each case separately.

(c) Table of constraints

Table	Primary Key	Domain of P.K.	Foreign keys (Referencing table)	Not Null
Customer	customer_id	VARCHAR	-	-
Package	package_id	VARCHAR	-	-
Location	location_id location_type	VARCHAR VARCHAR	-	-
stores	package_id location_id location_type	VARCHAR VARCHAR VARCHAR	package_id references Package location_id references Location location_type references Location	-
receives	package_id customer_id time_received	VARCHAR VARCHAR VARCHAR	package_id references Package customer_id references Customer	-
ships	package_id customer_id time_shipped	VARCHAR VARCHAR VARCHAR	package_id references Package customer_id references Customer	-

Table 2: Table of constraints

3 Problem 3

Find loophole in given web application scheme

Question:

Consider a carelessly written Web application for an online-shopping site, which stores the price of each item as a hidden form variable in the Web page sent to the customer; when the customer submits the form, the information from the hidden form variable is used to compute the bill for the customer. What is the loophole in this scheme? (There was a real instance where the loophole was exploited by some customers of an online shopping site, before the problem was detected and fixed.)

Answer:

A customer or hacker can anonymously edit the HTML source code of the Web page by right clicking on webpage and then selecting inspect option, and after this they can replace the value of the hidden variable price with whatever value they want, and thus they can use the modified Web page to place an order. So the Web application would then use the user-modified value as the price of the product.

Some times Customer can even buy the items for free if he changes the price to zero. In any case, company will go in loss, it won't even know because of the loophole. To avoid this situation, they should use some encryption method for the form or they shouldn't send the price as an element in the form, instead they should set it from back- end by default.

4 Problem 4

Find risk in given web application scheme

Question:
Consider another carelessly written Web application, which uses a servlet that checks if there was an active session, but does not check if the user is authorized to access that page, instead depending on the fact that a link to the page is shown only to authorized users. What is the risk with this scheme? (There was a real instance where applicants to a college admissions site could, after logging into the Web site, exploit this loophole and view information they were not authorized to see; the unauthorized access was however detected, and those who accessed the information were punished by being denied admission.)

Answer:
There is a concept known as web proxy logs, their main purpose is to relay URL requests from clients to a server, receive the responses from the server and send them back to the appropriate clients.The web proxy acts as a gateway between the Internet and browsers on a local network. Thus, analysing web proxy logs can give unobtrusive insights to the browsing behavior of computer users and provide an overview of the Internet usage in an organization. So there is risk of web proxies happening here.

Here, even if the link to the page is shown only to authorized users, an unauthorized user may somehow come to know of the existence of the link (for example, from an unauthorized user, or via Web proxy logs). The user may then login to the system, and access the unauthorized page by entering its URL in the browser. If the check for user authorization was unintentionally left out from that page, the user will be able to see the result of the page. So the user can use that unauthorized data somewhere else, and in turn it may have bad effect on the organization or company.

Solution for that is to use The HTTP referer attribute to block a naive attempt to exploit such loopholes, by ensuring the referer value is from a valid page of the web site. However, the referer attribute is set by the browser, and can be spoofed, so a malicious user can easily work around the referer check.

5 Problem 5

Explain how the protocol might use digital certificates to verify authenticity of the site.

Question:
Hackers may be able to fool you into believing that their Web site is actually a Web site (such as a bank or credit card Web site) that you trust. This may be done by misleading email, or even by breaking into the network infrastructure and rerouting network traffic destined for, say mybank.com, to the hacker's site. If you enter your user name and password on the hacker's site, the site can record it, and use it later to break into your account at the real site. When you use a URL such as https://mybank.com , the HTTPS protocol is used to prevent such attacks. Explain how the protocol might use digital certificates to verify authenticity of the site.

Answer:
Authentication can be handled by a system of digital certificates, whereby public keys are signed by a certification agency, whose public key is well known. For example, the public keys of the root certification authorities are stored in standard Web browsers. A certificate issued by them can be verified by using the stored public keys.

Digital certificates are widely used to authenticate Web sites to users, to prevent malicious sites from masquerading as other Web sites. In the HTTPS protocol (the secure version of the HTTP protocol), the site provides its digital certificate to the browser, which then displays it to the user. If the user accepts the certificate, the browser then uses the provided public key to encrypt data. A malicious site will have access to the certificate, but not the private key, and will thus not be able to decrypt the data sent by the browser. Only the authentic site, which has the corresponding private key, can decrypt the data sent by the browser. We note that public-/private-key encryption and decryption costs are much higher than encryption/decryption costs using symmetric private keys. To reduce encryption costs, HTTPS actually creates a one-time symmetric key after authentication, and uses it to encrypt data for the rest of the session.

Digital certificates can also be used for authenticating users. The user must submit a digital certificate containing her public key to a site, which verifies that the certificate has been signed by a trusted authority. The user's public key can then be used in a challenge–response system to ensure that the user possesses the corresponding private key, thereby authenticating the user.

6 Problem 6

Write a servlet and associated HTML code for the following simple application: A user logins using userid and password and then a servlet authenticates the user (based on user ids and passwords stored in a database relation), and sets a session variable called userid after authentication

6.1 Query

The q6.sql file contents are shown here.

```
-- creating Assign3_users database
CREATE DATABASE Assign3_users;
SHOW DATABASES;
-- using Assign3_users database
USE Assign3_users;
SELECT DATABASE();
-- creating table
-- users(user_id, password);
CREATE TABLE users(
  userid VARCHAR(15),
  password VARCHAR(25),
  PRIMARY KEY (userid)
);
SHOW TABLES;
DROP TABLE users;
-- inserting data into users table
INSERT INTO users VALUES('10001', 'qwertyui');
INSERT INTO users VALUES('10002', 'asdfghjk');
INSERT INTO users VALUES('10003', 'zxcvbnmm');
INSERT INTO users VALUES('10004', '12345678');
INSERT INTO users VALUES('10005', 'asdf1234');
```

The Login.jsp file contents are shown here.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<!-- <!DOCTYPE html> -->
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Login page</title>
<link
        href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/
        bootstrap.min.css"
        rel="stylesheet"
        integrity="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/
        et3URy9Bv1WTRi"
        crossorigin="anonymous">
<style>
form {
        width: 70%;
        margin: auto;
}
</style>
</head>
<body class="bg-warning p-3 mb-2">
<h1 class="display-4 fw-normal text-center m-5">Login Page</h1>
<form action="LoginServlet" method="post">
  <div class="form-group row ">
        <label for="userid" class="col-sm-2 col-form-label">
        User ID:</label>
        <div class="col-sm-10">
        <input type="text" name="userid" maxlength="15"
                class="form-control shadow p-2 mb-3" id="userid"
                placeholder="Enter user_id" required>
        </div>
  </div>
```

```
  <div class="form-group row">
        <label for="password" class="col-sm-2 col-form-label">
        Password:</label>
        <div class="col-sm-10">
        <input type="text" name="password" maxlength="25"
                class="form-control shadow p-2 mb-3" id="password"
                placeholder="Enter password" required>
        </div>
  </div>
  <button type="submit"
        style="width: 100%; margin: auto; display: block;"
        class="btn btn-primary form-control shadow p-2 my-3">
        Login</button>
</form>
<script
        src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/
        bootstrap.bundle.min.js"
        integrity="sha384-OERcA2EqjJCMA+/
        3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3"
        crossorigin="anonymous"></script>
</body>
</html>
```

The Result.jsp file contents are shown here.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="en">
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Result of login query</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/
          css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/
            et3URy9Bv1WTRi"
        crossorigin="anonymous">
<style>
</style>
</head>
<body class="bg-warning p-3 mb-2">
        <a><b>Logged in successfully !!</b></a>
        <%
        response.setContentType("text/html");
        if (session != null) {
                String name = (String) session.getAttribute("userid");
                out.print("Hello, " + name + " ! You Are Logged In.");
        } else {
                out.print("Please login first");
                request.getRequestDispatcher("Login.jsp").include(request,
                response);
        }
        out.close();
        %>
        <script
                src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/
                  js/bootstrap.bundle.min.js"
                integrity="sha384-OERcA2EqjJCMA+/
                3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3"
                crossorigin="anonymous"></script>
</body>
</html>
```

The ResultException.jsp file contents are shown here

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Result of login query</title>
<link
        href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/
              css/bootstrap.min.css"
        rel="stylesheet"
        integrity="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/
                  et3URy9Bv1WTRi"
        crossorigin="anonymous">
<style>
</style>
</head>
<body class="bg-warning p-3 mb-2">
        <a><b>Login failed !! Please check userid and password.</b></a>
        <script
                src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/
                dist/js/bootstrap.bundle.min.js"
                integrity="sha384-OERcA2EqjJCMA+/
                3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3"
                crossorigin="anonymous"></script>
</body>
</html>
```

The LoginServlet.java file contents are shown here

```
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.servlet.http.HttpSession;

@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
        private static final long serialVersionUID = 1L;
        // declaring variables for later use in making connection
        static final String DB_URL = "jdbc:mysql://localhost:3306/
                                      Assign3_users";
        static final String USER = "root";
        static final String PASS = "password";
        public LoginServlet() {
                super();
                // TODO Auto-generated constructor stub
        }
        protected void doPost(HttpServletRequest request,
                              HttpServletResponse response)
                        throws ServletException, IOException {
        try {
                // getting input values from jsp page
                String userid = request.getParameter("userid");
                String password = request.getParameter("password");
                Connection conn = null;
                Class.forName("com.mysql.jdbc.Driver");
                conn = DriverManager.getConnection(DB_URL, USER, PASS);
                System.out.println("Printing connection object " + conn);
```

```
  // Prepared Statement to check if the userid exists in the database.
  PreparedStatement checkUser = conn.prepareStatement(
      "select * from users where userid = ?");
  checkUser.setString(1, userid);
  ResultSet rs1 = checkUser.executeQuery();
        int countUser = 0;
        while (rs1.next()) {
                countUser += 1;
        }

  // Prepared Statement to check if the password exists in the database.
  PreparedStatement checkPass = conn.prepareStatement(
        "select * from users where password = ?");
  checkPass.setString(1, password);
  ResultSet rs2 = checkPass.executeQuery();
        int countPass = 0;
        while (rs2.next()) {
                countPass += 1;
        }

  if(countUser > 0 && countPass > 0){
        RequestDispatcher rd =
                request.getRequestDispatcher("Result.jsp");
          HttpSession session = request.getSession();
          session.setAttribute("userid", userid);
          rd.forward(request, response);
  } else {
          RequestDispatcher rd =
            request.getRequestDispatcher("ResultException.jsp");
          rd.forward(request, response);
  }
  } catch (Exception e) {
        e.printStackTrace();
  }
    }
}
```

6.2 Screenshots

All the cases are handled in following screenshots:

- All the inserted data for userid and password by using sql file is displayed in the figure 6.2.
- Logging in with correct credentials.
- Logging in with in-correct credentials.

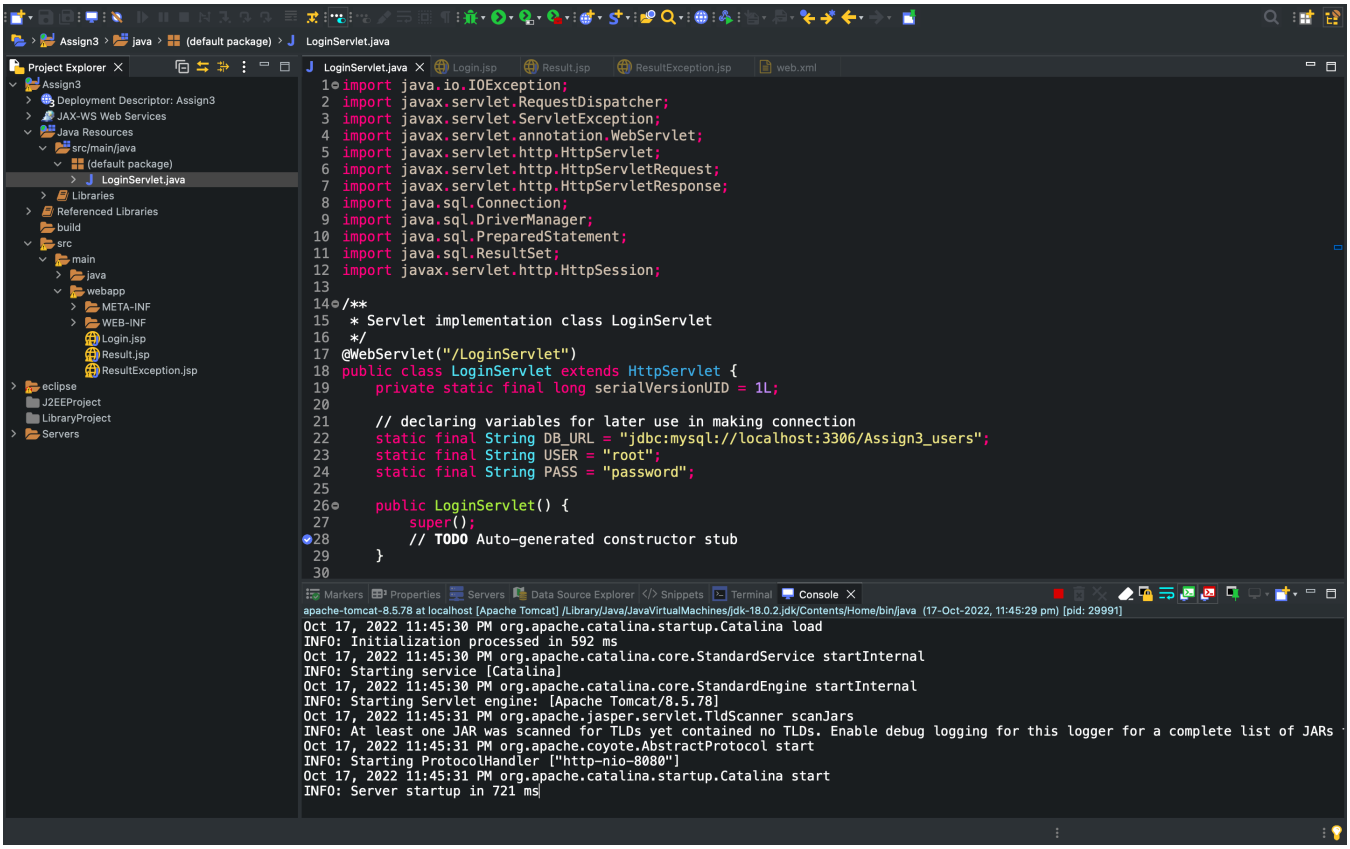

Figure 3: IDE Screenshot with running server

	* userid varchar(15)	password varchar(25)
1	10001	qwertyui
2	10002	asdfghjk
3	10003	zxcvbnmm
4	10004	12345678
5	10005	asdf1234

Figure 4: Data present inside `users` table

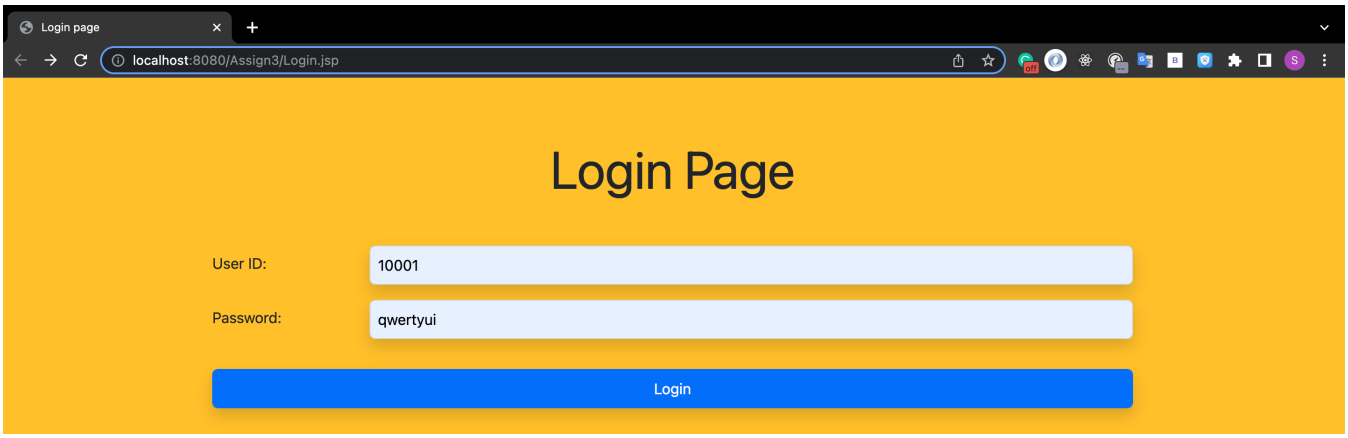

Figure 5: Logging in with correct credentials

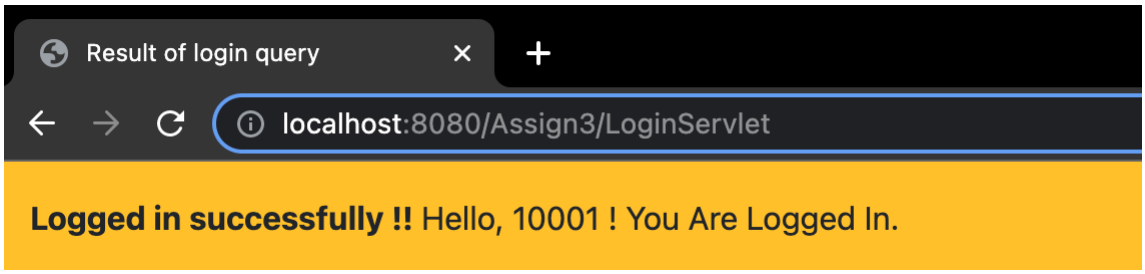

Figure 6: Result of correct input and displaying session variable

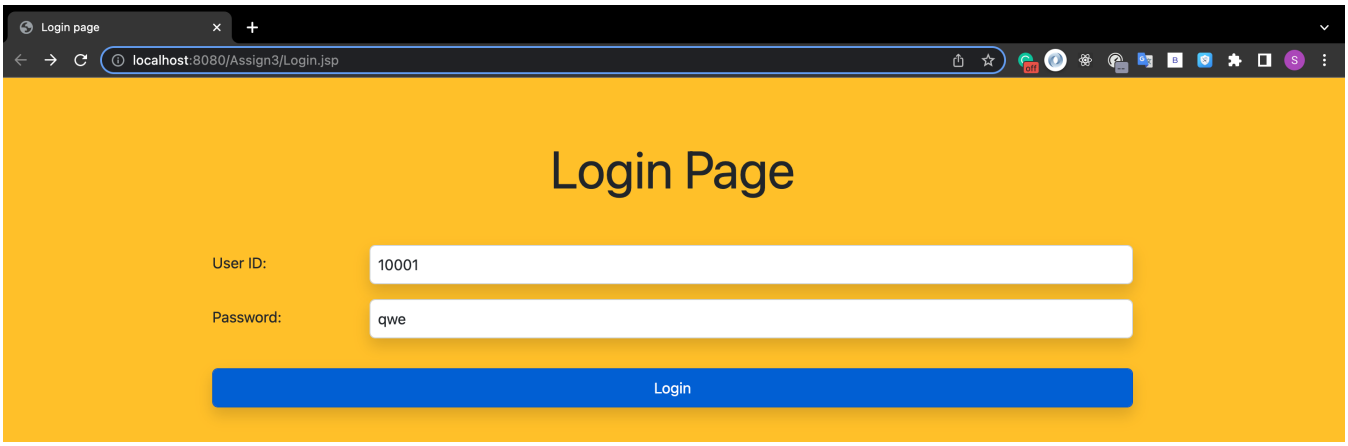

Figure 7: Logging in with incorrect credentials

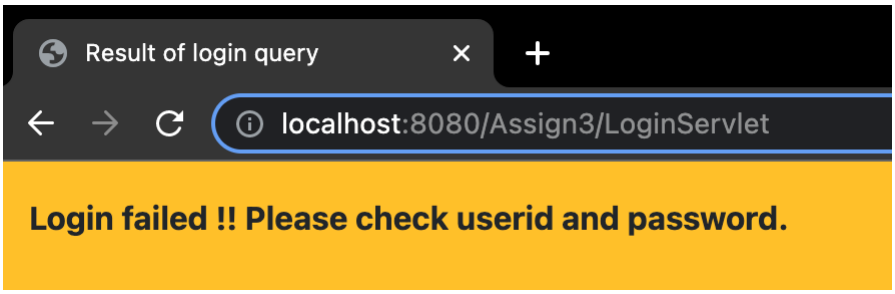

Figure 8: Result of incorrect input

7 Problem 7

XSS Attack

Question:
What is an XSS attack? Explain how it works, and what measures must be taken to detect or prevent XSS attacks?

Answer:
A Web site that allows users to enter text, such as a comment or a name, and then stores it and later displays it to other users, is potentially vulnerable to a kind of attack called a cross-site scripting (XSS) attack. In such attack, when a different user views the entered text, the browser would execute the script, which can carry out actions such as sending private cookie information back to the malicious user, or even executing an action on a different Web server that the user may be logged into.

We can take an example of bank website. Lets say that user logins to his bank account at the time the script executes. Then that script is able to send cookie information regarding his bank account login back to the malicious user, who could use the information to connect to the bank's Web server and may fool everyone to believe that the connection is from the original user.

Cross-site scripting (XSS) attack can be done in other ways also, like tempting a user into visiting a Web site that has malicious scripts embedded in its pages. Thus user gets caught in hacker's trap, and hacker gets the access to his data.

To avoid such attacks following things can be done:

1. Prevent your Web site from being used to launch XSS or XSRF attacks:
   This can be achieved by avoid taking text inputs from the users. We can also use functions that detect, or strip all such tags. These functions can be used to prevent HTML tags, and as a result, any scripts, from being displayed to the other users.

2. Protect your Web site from XSS or XSRF attacks launched from other sites:
   If user log into to our web site and then he visits different web site vulnerable to XSS, then in such case the malicious code executing on the user's browser could execute actions on our Web site and can have access to session information related to our site. So some methods are there to prevent this. First being to check if the referer generated by HTTP protocol is valid. Second solution can be instead of using only the cookie to identify a session, the session could also be restricted to the IP address from which it was originally authenticated. Another solution could be to never use a GET method to perform any updates.