

VIETNAM NATIONAL UNIVERSITY HCMC  
INTERNATIONAL UNIVERSITY



## FINAL PROJECT REPORT

# MOVIE SYSTEM

**Course:** Principles of Database Management– S1\_2023-2024\_G01

**Instructor:** Assoc. Prof. Nguyen Van Sinh

**Lab:** MSc. Nguyen Quang Phu

***prepared by:***

Nguyễn Phúc Vinh - ITITIU21350

Hoàng Gia Huy - ITCSIU21186

Lê Khánh Nguyên - ITCSIU21211

Submission date: 18/1/2023

# Contents

<b>A. Abstract .....</b>	<b>2</b>
<b>1. Objectives .....</b>	<b>3</b>
<b>2. Tool and Programing Language.....</b>	<b>3</b>
<b>3. Member.....</b>	<b>3</b>
<b>4. Information .....</b>	<b>4</b>
<b>B. Outline .....</b>	<b>4</b>
<b>1. Direction .....</b>	<b>5</b>
<b>2. Elements .....</b>	<b>6</b>
<b>3. ER model .....</b>	<b>7</b>
<b>4. View the database .....</b>	<b>8</b>
<b>5. Change from ER model to relationship model .....</b>	<b>11</b>
<b>6. Create SQL server .....</b>	<b>12</b>
<b>7. Random data generation .....</b>	<b>14</b>
<b>8. Create database and insert data.....</b>	<b>16</b>
<b>9. Java coding.....</b>	<b>18</b>
<b>10. Conclusion .....</b>	<b>39</b>
<b>C. REFERENCE.....</b>	<b>40</b>

# A.Abstract

## 1. Objective

The project aim is to recreate a fully movie system that hadle all basic job that a moive system by applying the knowledge of Principal of Database Management course.

In short, this project is to:

- Create and redesigning the movie system that existed.
- Apply all the knowledge of The Principal of Database Management course in real application.
- Have a brief look in website/app development and database management project.
- Learn to build and manage a whole project from the beginning.
- Evaluate the ability to build a big project.

## 2. Tool and Programing language

IDEs for programming and debugging: Apache NetBeans IDE 19 and IntelliJ IDEA 2022.3.2

Code version management: Git

Project management: [Github](#)

Programing language: Java and HTML

Style sheep language: CSS

### 3. Member

Name	ID Student
Nguyễn Phúc Vinh	ITITIU21315
Hoàng Gia Huy	ITCSIU21186
Lê Khánh Nguyên	ITCSIU21211

### 4. Information

**Movie System** is a system to manage the schedule of movie in each cinema and help the customer to booking their favorite movies more easily. The user of the system include: customers, staffs, and the manager.

Customer is the one use the system at the end or the end-user. With that website, customers could easily register and then log in to book their favorite movie at the cinema as they want. Moreover, the system also help customer to book the seat as well as to pay for ticket. Customers could book more than one ticket and booking many schedules as they want. Furthermore, customer could manage to see their booking history and the bill.

The staffs are who care most for managing the system. They could add or remove movies, cinemas for the booking system as well as manage the schedule by add more schedules and the information of this schedule by adding categories, prices, and actors,...

The admin in this case is the manager, his job is to add or remove staffs for the system.

The major concern in the movie system is to control the flow of information when the user as customers, staffs, or admin create new information. By this brand-new movie system, we could handle all job that a movie system that need. Including booking movie, manage the schedule as well as cinemas system, manage staffs of company,...

## **B. Outline**

### **1.Direction:**

a/ Start

To begin with, our main vision is to design a systematic movie management system and to validate the design using a tool called SQL Server.



Therefore, our group decided that we will have a meeting via Discord every week to have a discussion about our project such as: share the information we know to the others and report about what we have done to present to our leader.

b/ Brainstorm

Below are the process that we design the database (Figure 1)

Sua chui	Movie	Customer	Booking
- ID	- ID	- ID	- ID movie
- Name	- Name	- Name	- ID customer
- Thời gian Khung giờ	- Category	- Username	- Room
- Thể loại	- Đạo diễn	- Password	- Phòng
- Rạp nào	- Diễn viên	- Tuổi	- <del>Thị</del>
- Phòng	- Đánh giá	- Favorite	- Ghế
- <del>Gia</del>	- Rating	- I	- Thời lượng
- <del>Gia</del>	- Thời gian chiếu		- Hình thức thanh toán
- <del>Gia</del>	- Thời lượng		- <del>Thời gian</del>
- Lưu phòng	- <del>Price</del>		- Thời gian thanh toán / đặt
- Ngày	- Giá		
	Gia	Cinema	
	- ID sua chui	- ID	
	- <del>Thể loại</del>	- Name	
	- Ghế	- Location	
	- Lưu phòng		

Figure 1- First design for the database

## 2. Elements

i. The list of elements incorporated with the “movie management system” is:

1. Schedule: The information about the movie such as when, price, cinema,...about the movie ticket.
2. Admin: The manager whose job is to add/remove staff from the system
3. Staff: The ones who care about the system such as updating schedule, set time, set cinemas,...
4. Customer: The end-user of the system, who use the system to booking the ticket

### 3. ER model:

ER Model stands for Entity-Relationship Model, also known as a high-level data model that shows the relationship among the entity sets. ER Model is used to define the entities and the relationships between them.

So far, we have done the ER model to help us analyze the data, identify the basic units of information needed, and describe their structure and relationships. Here is our ER model (Figure 2).

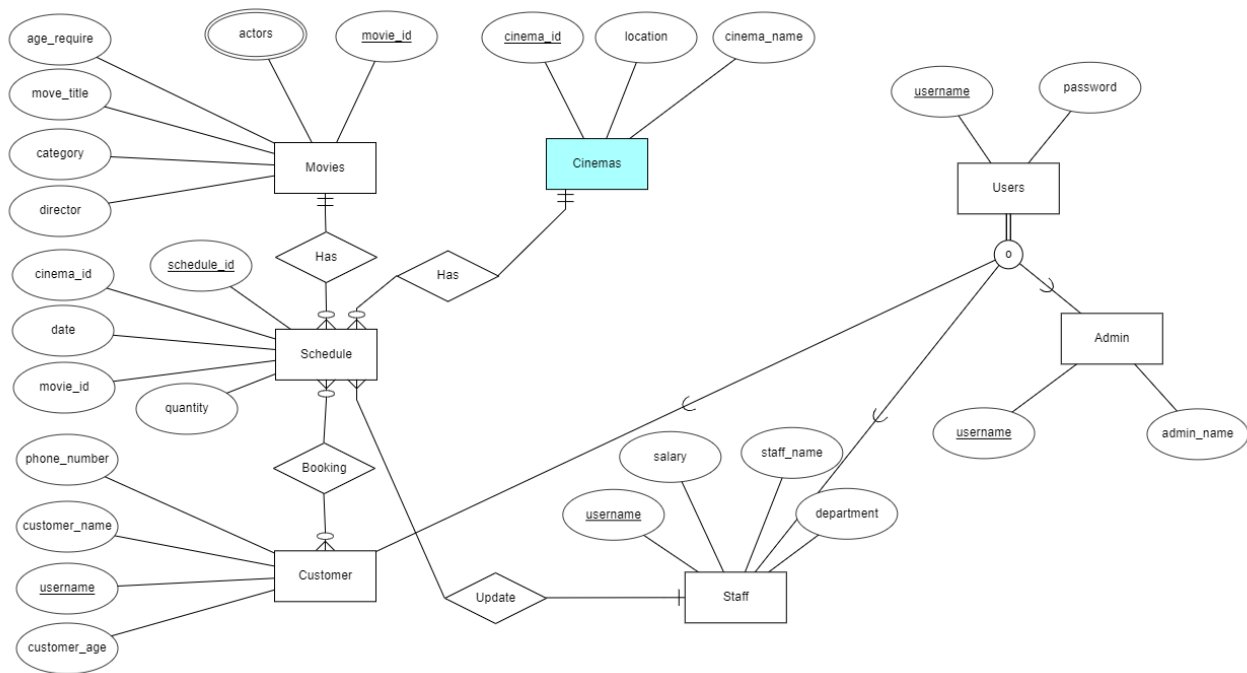
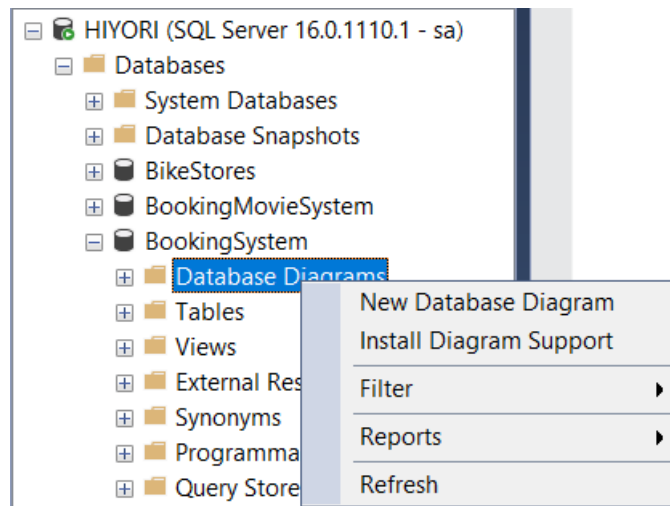


Figure 2.ERD model for Movie System website

#### 4. View in Database:

*To have the ERD in database, we manually create by following steps:*

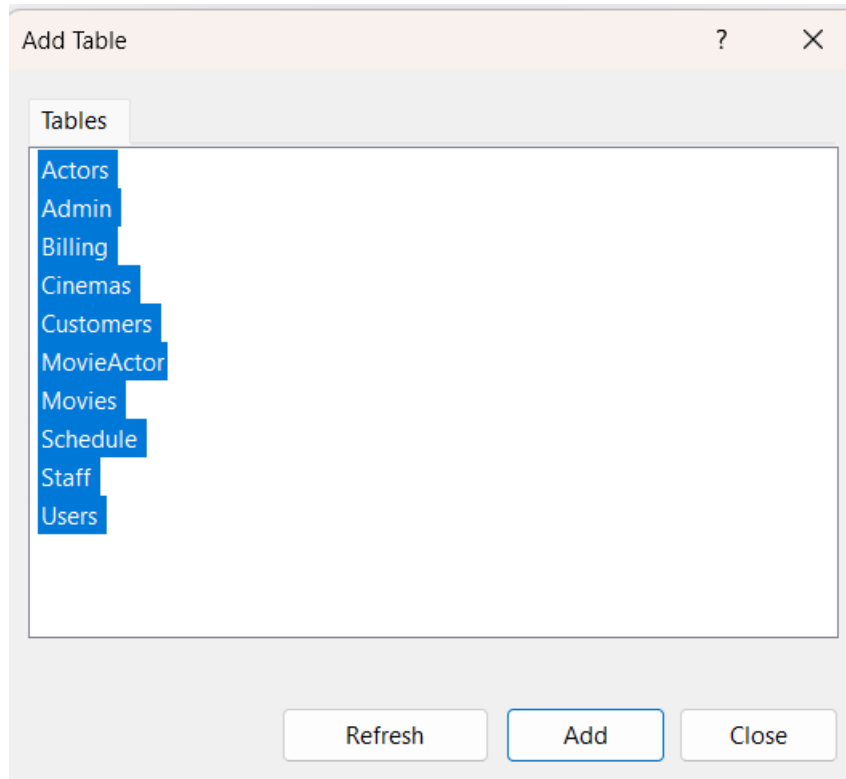
- i. To have the MovieSystem diagram, Right-click on Database Diagrams and choose NewDatabase Diagram (Figure 3).



*Figure 3. The database diagram design menu*

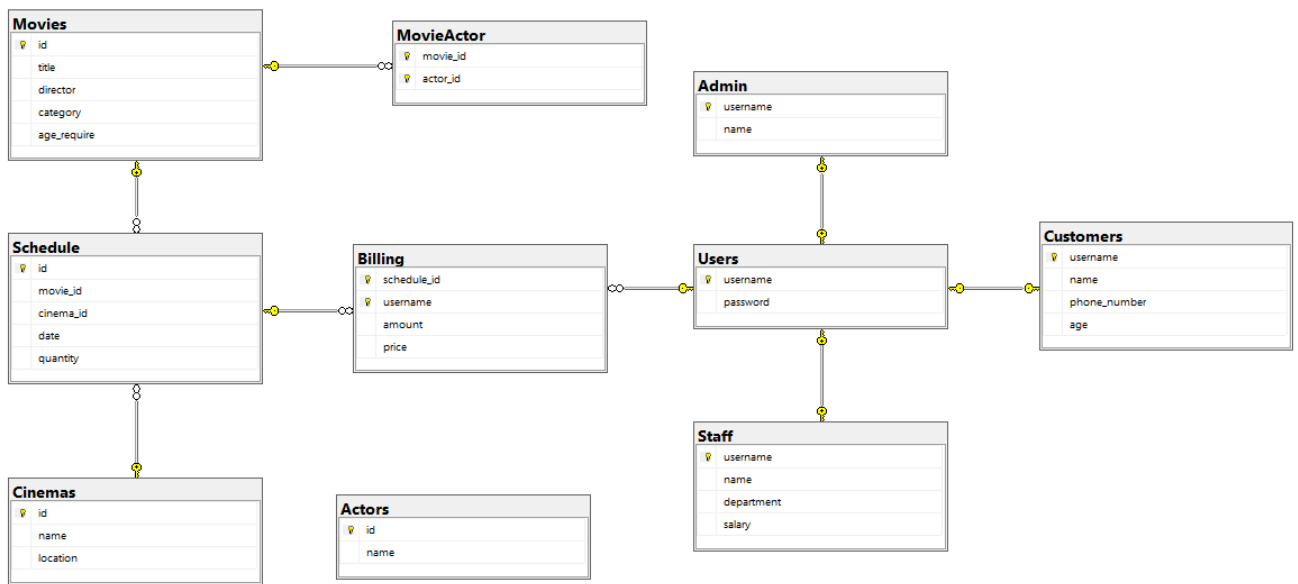


- ii. Select all tables that we want to display in ERD, then choose Add.



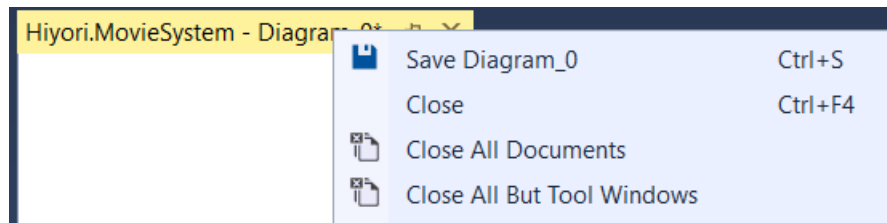
*Figure 4. Add table to the diagram*

- iii. Then Close Add Table, after that we will get an ERD Diagram on the database.



*Figure 5. The diagram of the Movie System*

- iv. Right-click on the Diagram tab and choose Save Diagram to save the newly created ER Diagram.



*Figure 6. Save diagram*

## **5. Change from ER model to relationship model:**

### **5.1.1. Relationship model definition:**

The relational model is an abstract model used to organize data within a database. In order to control access to a database, write data, run queries, or perform any other tasks related to database management, a database management system must have some kind of underlying model that defines how the data within it are organized.

### **5.1.2. Why do we transfer data from SQLSever into mySQL:**

SQLServer is an application that is difficult to set up a method to log into the server, takes a long time to download, consumes a lot of cache to run the application, and lacks documentation to support the use of drivers ( JDBC). MySQL is a popular application now and there are many resources to support its use and learn about issues

related to it, especially useful for beginners. So we decided to use MySQL as the main application to use in this project.



*Figure 7. MySQL logo*

## **6. Create SQL Server:**

### **6.1.1. What is SQL server:**

**SQL Server** is a relational database management system (RDBMS) developed by Microsoft. It is primarily designed and developed to compete with MySQL and Oracle databases. SQL Server supports ANSI SQL, which is the standard SQL (Structured Query Language) language.

### **6.1.2. What is mySQL:**

MySQL is an open source relational database management system (RDBMS) with

a client-server model. RDBMS is a software or service used to create and manage databases based on a relational model.

We can design the table and put it in SQL server as follow or using query

Hiyori.BookingSystem - dbo.Schedule			
	Column Name	Data Type	Allow Nulls
PK	Sc_ID	int	<input type="checkbox"/>
	Cine_ID	int	<input checked="" type="checkbox"/>
	Room_ID	int	<input checked="" type="checkbox"/>
	Mov_ID	int	<input checked="" type="checkbox"/>
	Date	date	<input type="checkbox"/>
	Start_Time	time(7)	<input type="checkbox"/>
	End_Time	time(7)	<input type="checkbox"/>
	Price	decimal(10, 2)	<input type="checkbox"/>
			<input type="checkbox"/>

*Figure 8. Booking table*

	Column Name	Data Type	Allow Nulls
PK	schedule_id	int	<input type="checkbox"/>
	username	varchar(50)	<input type="checkbox"/>
	amount	int	<input type="checkbox"/>
	price	int	<input type="checkbox"/>
			<input type="checkbox"/>

*Figure 9. Schedule Table*

Hiyori.MovieSystem - dbo.Customers		Hiyori.MovieSystem - dbo.Billing	
	Column Name	Data Type	Allow Nulls
🔑	username	varchar(50)	<input type="checkbox"/>
	name	varchar(50)	<input type="checkbox"/>
	phone_number	int	<input type="checkbox"/>
	age	int	<input type="checkbox"/>
▶			<input type="checkbox"/>

*Figure 10. Customer table*

Hiyori.MovieSystem - dbo.Users		Hiyori.MovieSystem - dbo.Customers	
	Column Name	Data Type	Allow Nulls
🔑	username	varchar(50)	<input type="checkbox"/>
	password	varchar(255)	<input type="checkbox"/>
			<input type="checkbox"/>

*Figure 11. MovieSystem table*

## 7. Random Data Generation:

Data is then randomly generated as following

### 2.7.1. Code render data:

In order to simulate the real case, a large amount of sample information is needed to execute queries and solve runtime problems. So our team decided to make an information render to make it look more like the real thing (Figure 12 and 13).

```

3
4 def generate_random_customer():
5     names = ["Huy", "Vinh", "Nguyen", "Tram", "Nghia", "An", "Thao", "Huong", "Anh", "Duc", "Dung"]
6     name = random.choice(names)
7
8     username = name.lower() + ''.join(random.choice(string.ascii_lowercase + string.digits) for _ in range(3))
9
10    customer_age = random.randint(10, 45)
11
12    phone_number = ''.join(random.choice(string.digits) for _ in range(10))
13
14    customer = {
15        "name": name,
16        "username": username,
17        "customer_age": customer_age,
18        "phone_number": phone_number
19    }
20
21    return customer
22

```

*Figure 12. Python code to generate customer information*

```

redering_data.py X
redering_data.py > generate_password
1 import random
2 import string
3 def generate_password(username):
4
5     random_chars_digits = ''.join(random.choice(string.ascii_letters + string.digits) for _ in range(5))
6
7     password = username + random_chars_digits
8
9     return password
10
11
12
13

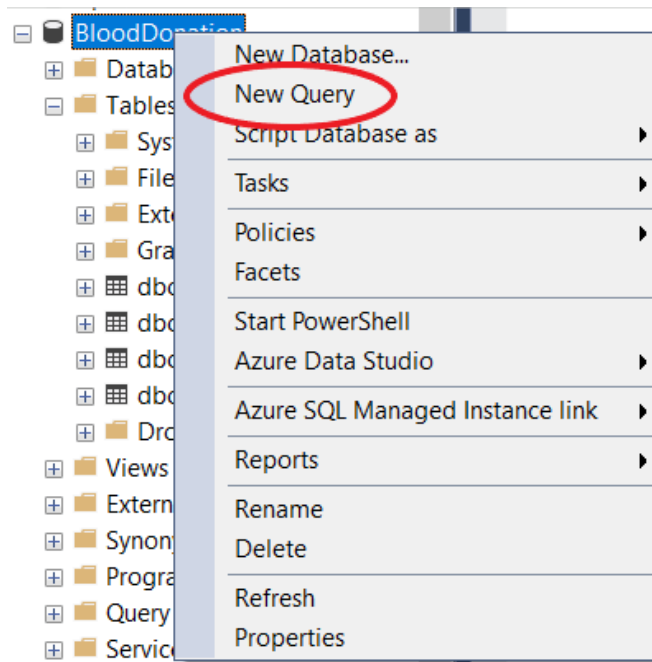
```

*Figure 13. Python code to generate password information*

## 8. Create database and insert data:

### 8.1.1. Creating table query:

First, we create a SQL file to work with SQL database



*Figure 14. Create new query to create database*

Then, we use SQL commands to create database and table

For example

Step 1: Drop the database name 'MovieSystem' if it does exist (Figure 15)



```
--Check if the MovieSystem database exist or not, if exist delete it
IF EXISTS (SELECT 1 FROM sys.databases WHERE name = 'MovieSystem')
BEGIN
    DROP DATABASE MovieSystem;
END
```

*Figure 15. Delete the existing database*

Step 2: Creating the database by the CREATE DATABASE command (Figure 16).

```
CREATE DATABASE MovieSystem;
```

*Figure 16. Create command*

Step 3: Creating table

```
CREATE TABLE Cinemas(
    id INT PRIMARY KEY IDENTITY(300, 1) NOT NULL,
    name VARCHAR(255) NOT NULL,
    location VARCHAR(255) NOT NULL,
);

CREATE TABLE Schedule(
    id INT PRIMARY KEY IDENTITY(400, 1) NOT NULL,
    movie_id INT,
    cinema_id INT,
    date DATE NOT NULL,
    quantity INT NOT NULL,
    FOREIGN KEY (movie_id) REFERENCES Movies(id),
    FOREIGN KEY (cinema_id) REFERENCES Cinemas(id)
);
```

*Figure 17. The command to create table*

### 8.1.2.Insert data

Using command “Insert into TABLE NAME(Field 1, Filed 2, Field 3,...) values”

For example:

```

-- Inserting sample data into Users and Admin table
INSERT INTO Users(username,password) values ('admin','admin');
INSERT INTO Admin VALUES('admin', 'admin1234');

-- Inserting sample data into Actors table
INSERT INTO Actors (name) VALUES
('Morgan Freeman'),
('Marlon Brando'),
('Heath Ledger'),
('John Travolta'),
('Tom Hanks');

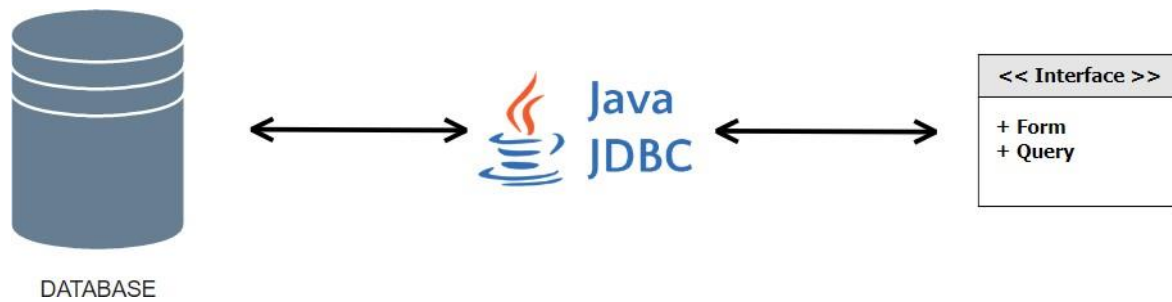
-- Inserting sample data into Cinemas table
INSERT INTO Cinemas (name, location) VALUES
('Cineplex 1', 'Downtown'),
('AMC Theatres 2', 'Midtown'),
('Regal Cinemas 3', 'Uptown'),
('Vue Cinema 4', 'Westside'),
('Odeon 5', 'Eastside');

```

*Figure 18. Command to insert the data*

## 9. Java Coding:

### 9.1.1. Structure:



*Figure 19. The flow of database and web applications*

That Structure showed the most basic idea while we tried to build a java website to deal with Database.

#### 9.1.2. JDBC(what is JDBC):

JDBC stands for Java Database Connectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.

- Making a connection to a database.
- Creating SQL or MySQL statements.
- Executing SQL or MySQL queries in the database.
- Viewing & Modifying the resulting records.

```

public class DBConnect {

    private static final String JDBC_DRIVER = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
    private static final String DB_URL = "jdbc:sqlserver://localhost:1433;databaseName=MovieSystem;encrypt=false";
    private static final String USER = "sa";
    private static final String PASSWORD = "huy1234";

    static {
        try {
            Class.forName(className:JDBC_DRIVER);
        } catch (ClassNotFoundException e) {
            System.out.println(x: e);
        }
    }

    public static Connection getConnection() {
        Connection conn = null;
        try {
            conn = DriverManager.getConnection(url: DB_URL, user:USER, password: PASSWORD);
        } catch (SQLException e) {
            System.out.println(x: e);
        }
        return conn;
    }
}

```

Figure 20. Connect database to web application

The program control flow logic is as follows;

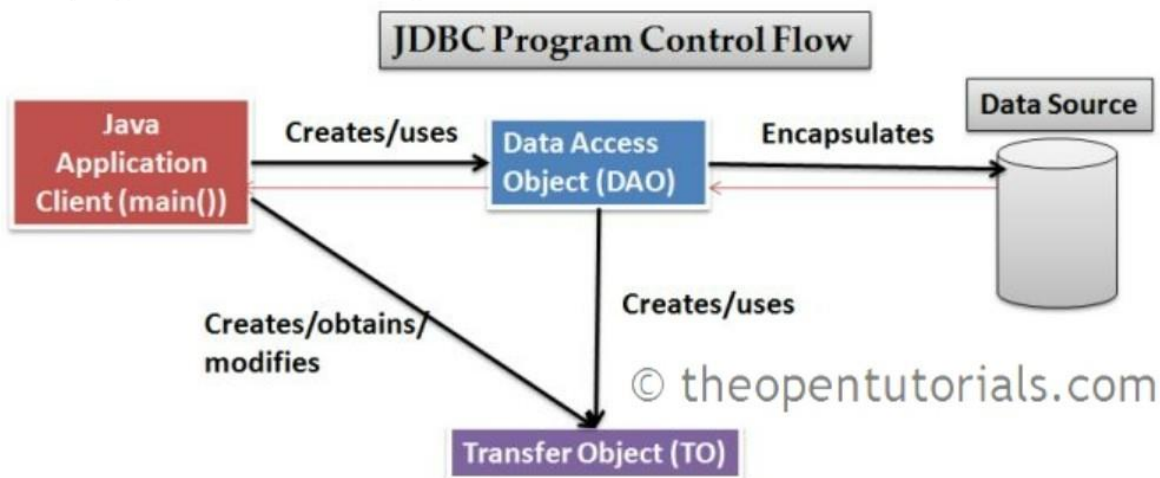


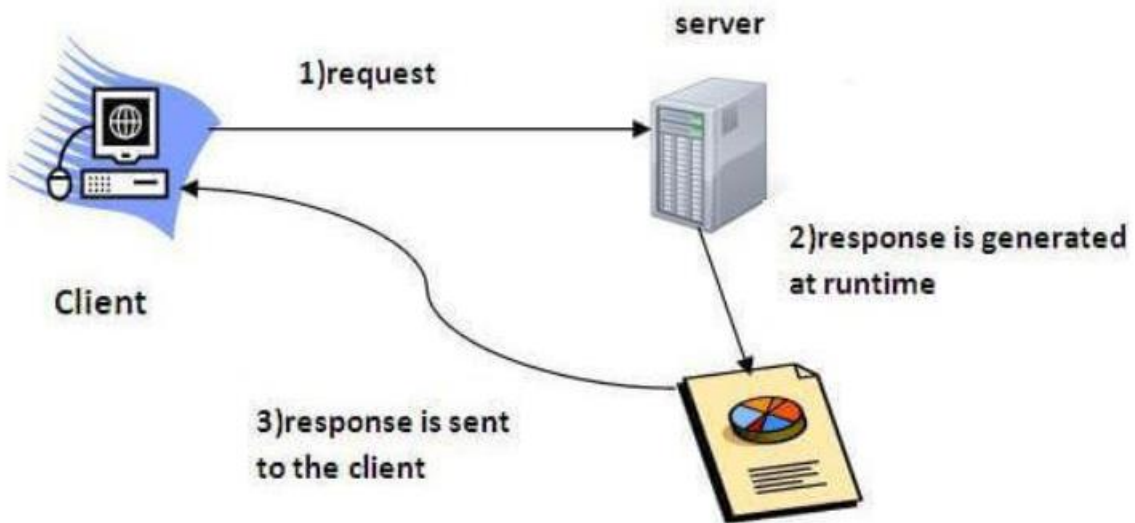
Figure 21. The JDBC control flow

### 9.1.3. Servlet:

Servlet can be described in many ways, depending on the context.

- Servlet is a technology which is used to create a web application.
- Servlet is an API that provides many interfaces and classes including documentation.

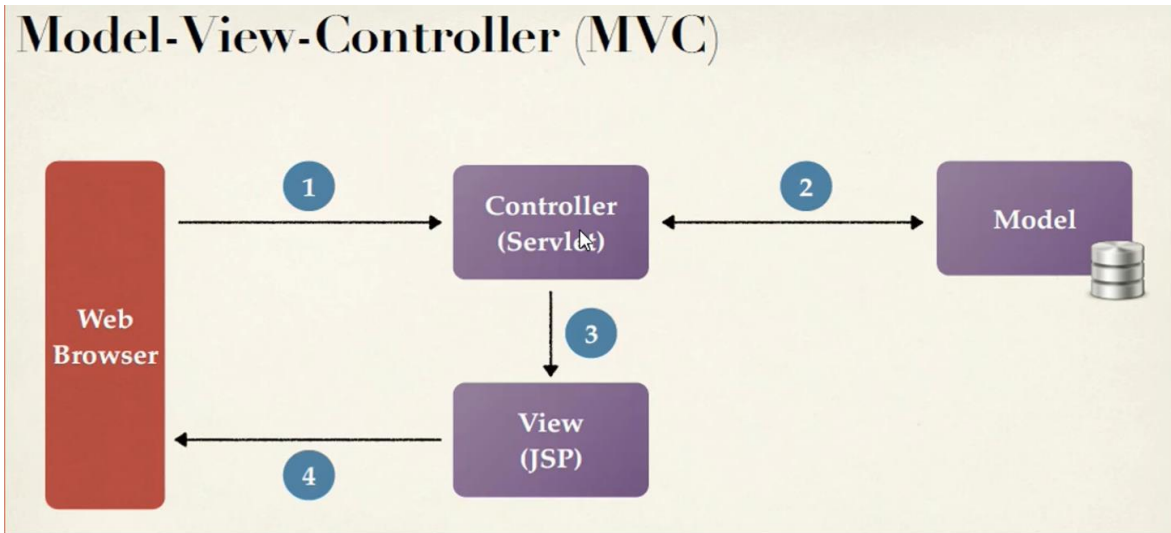
- Servlet is an interface that must be implemented for creating any Servlet.
- Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
- Servlet is a web component that is deployed on the server to create a dynamic web page.



*Figure 22. Servlet flow*

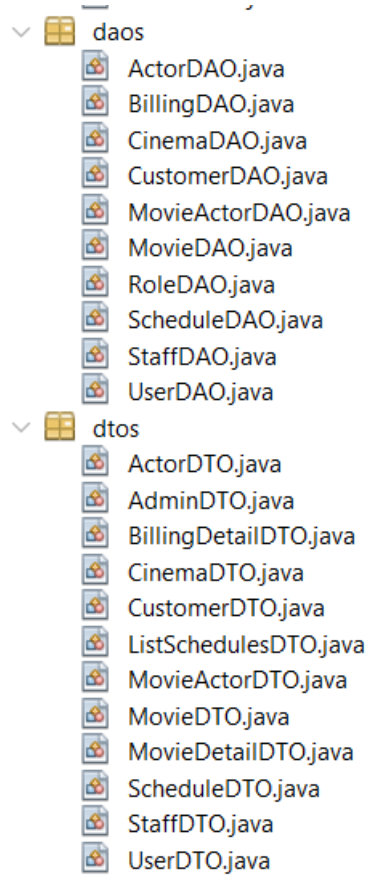
#### 9.1.4.MVC (Model-View-Controller) design pattern:

-This is a pattern in software design commonly used to implement user interfaces, data, and controlling logic. It emphasizes a separation between the software's business logic and display. This "separation of concerns" provides for a better division of labor and improved maintenance.



*Figure 23. The basic idea of MVC design pattern*

-It divides the source code into three part: model as DAO (Data Access Object) classes and DTO (Data Transfer Object) classes (Figure 23, 24), controller servlet as logic behind , and view as UI for users to communicate with system behind .



*Figure 24. All the DAO and STO class of Movie System project*

```
public class ActorDAO {
    private Connection conn = null;
    private PreparedStatement stm = null;
    private ResultSet rs = null;

    public List<ActorDTO> getAll() {
        List<ActorDTO> list = new ArrayList<>();
        String sql = "SELECT * FROM Actors";

        try {
            conn = DBConnect.getConnection();
            stm = conn.prepareStatement(string:sql);
            rs = stm.executeQuery();

            while (rs.next()) {
                ActorDTO actor = new ActorDTO(id: rs.getInt(string:"id"), name: rs.getString(string:"name"));
                list.add(e: actor);
            }
            return list;
        } catch (SQLException e) {
            System.out.println(x: e);
        }
        return null;
    }
}
```

*Figure 25. A part of DAO class for Actor object*

-DAO classes have main function to access the database through query command to return the needed information for DTO classes. DTO classes, in the other hand, as a object to save the data of object after quering (Figure 25).

```
public class ActorDTO {  
    private int id;  
    private String name;  
  
    public ActorDTO(int id, String name) {  
        this.id = id;  
        this.name = name;  
    }  
  
    public int getId() {  
        return id;  
    }  
  
    public void setId(int id) {  
        this.id = id;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

*Figure 26. DTO class for Actor object*

-Servlet classes are classes that contain the behind logic each function of a page. For example. For example, the Login.java is a servlet class that contains the logic behind the login.jsp. (Figure 26, 27, and 28)



```

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    HttpSession session = request.getSession(bln:true);
    try {
        String username = request.getParameter(string:"username");
        String password = request.getParameter(string:"password");

        UserDAO dao = new UserDAO();
        UserDTO user = dao.checkLogin(username, password);
        String name = dao.getNameByUsername(username);
        if (user != null) {
            session.setAttribute(string:"USER", o: user);
            session.setAttribute(string:"NAME", o: name);

            response.sendRedirect(string:"Main");
        } else {
            session.setAttribute(string:"InvalidUser", o: "Invalid username or password");
            response.sendRedirect(string:"login");
        }
    } catch (IOException e) {
        System.out.println(x: e);
    }
}

```

*Figure 27. The logic behind the login button*

```

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    int movieId = Integer.parseInt(s: request.getParameter(string:"movie_id"));
    int cinemaId = Integer.parseInt(s: request.getParameter(string:"cinema_id"));
    Date date = Date.valueOf(s: request.getParameter(string:"date"));
    int quantity = Integer.parseInt(s: request.getParameter(string:"quantity"));

    ScheduleDAO scheduleDAO = new ScheduleDAO();
    boolean checkAdd = scheduleDAO.addSchedule(movie_id: movieId, cinema_id: cinemaId, date, quantity);
    if (checkAdd) {
        response.sendRedirect(string:"home");
    }
}

```

*Figure 28. The logic of adding schedule*

```

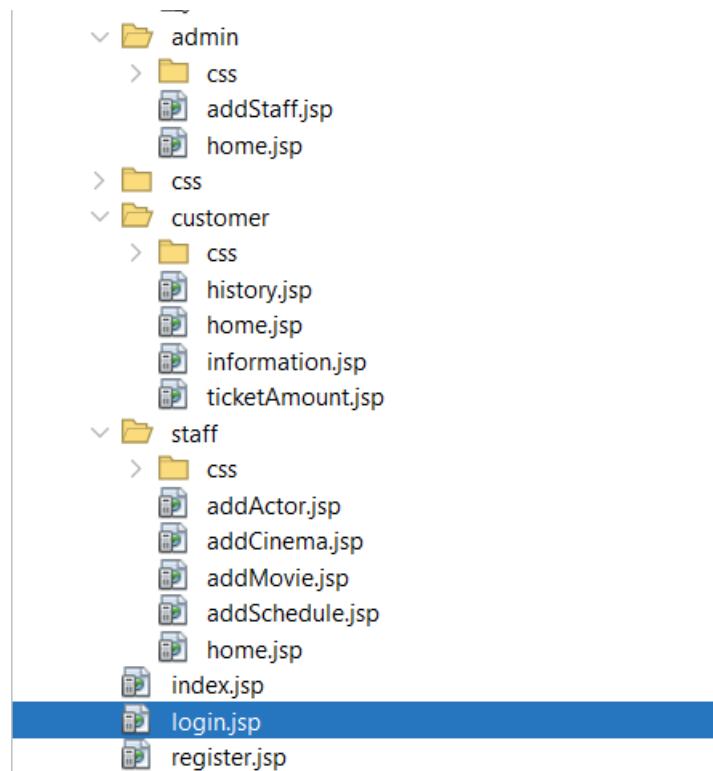
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    HttpSession session = request.getSession(true);
    UserDTO user = (UserDTO) session.getAttribute(string:"USER");
    String username = user.getUsername();

    BillingDAO billingDAO = new BillingDAO();
    List<BillingDetailDTO> bills = billingDAO.getBillByUsername(username);
    if (bills != null) {
        request.setAttribute(string:"BILLS", o: bills);
    }
    request.getRequestDispatcher(string:"history.jsp").forward(sr: request, srl: response);
}

```

*Figure 29. Logic behind show customer history at history.jsp*

-JSP file: the UI for communicating with user (in that case this is customer, staffs, and admin



*Figure 30. All the jsp file for the project*

GUI stands for Graphical User Interface in Java and is an easy-to-use visual experience builder for Java applications<sup>1</sup>. It comprises graphical units like buttons, labels, windows, etc., through which users can interact with an application<sup>2</sup>.

The User Interface in that case is the jsp file which is the combination of java and html language programming and CSS as the style sheet language.

- **Jakarta Server Pages (JSP):** is a collection of technologies that helps software developers create dynamically generated web pages based on HTML, XML, SOAP, or other document types.
- **HyperText Markup Language (HTML):** is the standard markup language for documents designed to be displayed in a web browser. It defines the content and structure of web content.
- **Cascading Style Sheets (CSS):** is a style sheet language used for specifying the presentation and styling of a document written in a markup language such as HTML or XML (including XML dialects such as SVG, MathML or XHTML) (Figure 31)

```

] .modal {
    position: fixed;
    top: 0;
    right: 0;
    left: 0;
    bottom: 0;
    display: flex;
- }

] .modal__overlay {
    position: absolute;
    width: 100%;
    height: 100%;
    background-color: rgba(36, 36, 36, 0.1);
- }

] .modal__body {
    width: 400px;
    height: 400px;
    background-color: #c9dcf2;
    margin: auto;
    position: relative;
    border-radius: 5px;
- }

] .auth-form {
    padding: 0 30px;
    width: auto;
- }

```

*Figure 31. A part of CSS for addSchedule.jsp*

In this project we use JSP, HTML, and CSS to make a simple Form Query. So that we can show data after fetching in the DataBase.

#### 9.1.6. Flow of process

In general, the process contain 3 step. First of all, user interact with the UI through jsp file. For example, customer clicks on the register button then the jsp file would send a request to the servlet, it depends on the GET or POST request when we define the button, it would run the corresponding method. In this case, the login button we define as POST request

(Figure 32) as I highlighted in blue color. When request was sent to the servlet, it would revoke the doPost method that implement the logic behind the login button (Figure 33) by map to the web.xml (Figure 34). In that case, the doPost method get the username and password by getParameter() method. Then it checks the existence of username and password in the Users table in the database through UserDao class by CheckLogin() method. If exist, then it would save the user and his/her name as sessionRequest and redirect the user to the Customer/Staff/Admin main page by requesting to the their jsp file path.

```
<form action="login" method="post">
  <div class="auth-form__form">
    <div class="auth-form__group">
      <label for="username">Username:</label>
      <input type="text" class="auth-form__input" id="username" name="username" required >
    </div>

    <div class="auth-form__group">
      <label for="password">Password:</label>
      <input type="password" class="auth-form__input" id="password" name="password" required >
    </div>
  </div>

  <div class="auth-form__submit">
```

Figure 32. The implementation of login button on login.jsp

```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    HttpSession session = request.getSession(bln: true);
    try {
        String username = request.getParameter(string: "username");
        String password = request.getParameter(string: "password");

        UserDao dao = new UserDao();
        UserDTO user = dao.checkLogin(username, password);
        String name = dao.getNameByUsername(username);
        if (user != null) {
            session.setAttribute(string: "USER", o: user);
            session.setAttribute(string: "NAME", o: name);

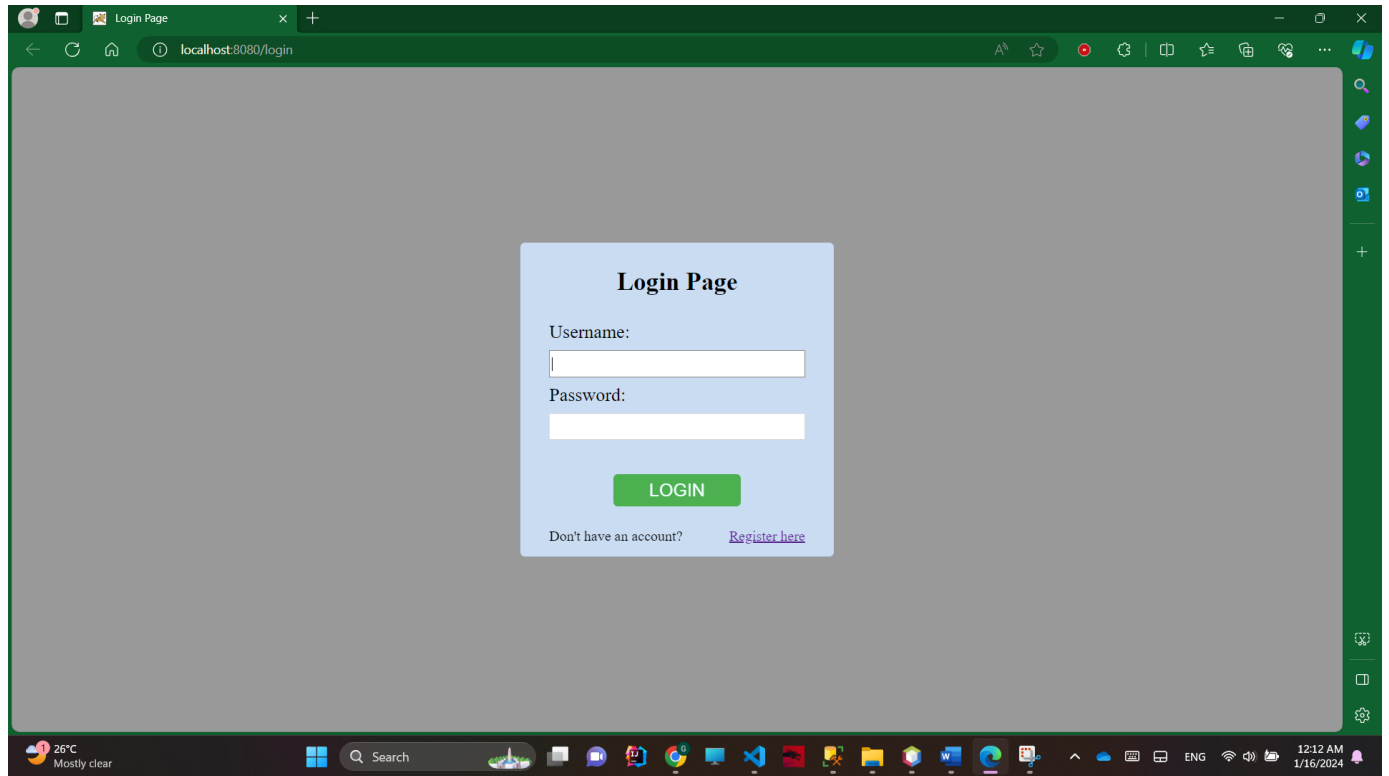
            response.sendRedirect(string: "Main");
        } else {
            session.setAttribute(string: "InvalidUser", o: "Invalid username or password");
            response.sendRedirect(string: "login");
        }
    } catch (IOException e) {
        System.out.println(x: e);
    }
}
```

*Figure 33. the implementation of doPost() method*

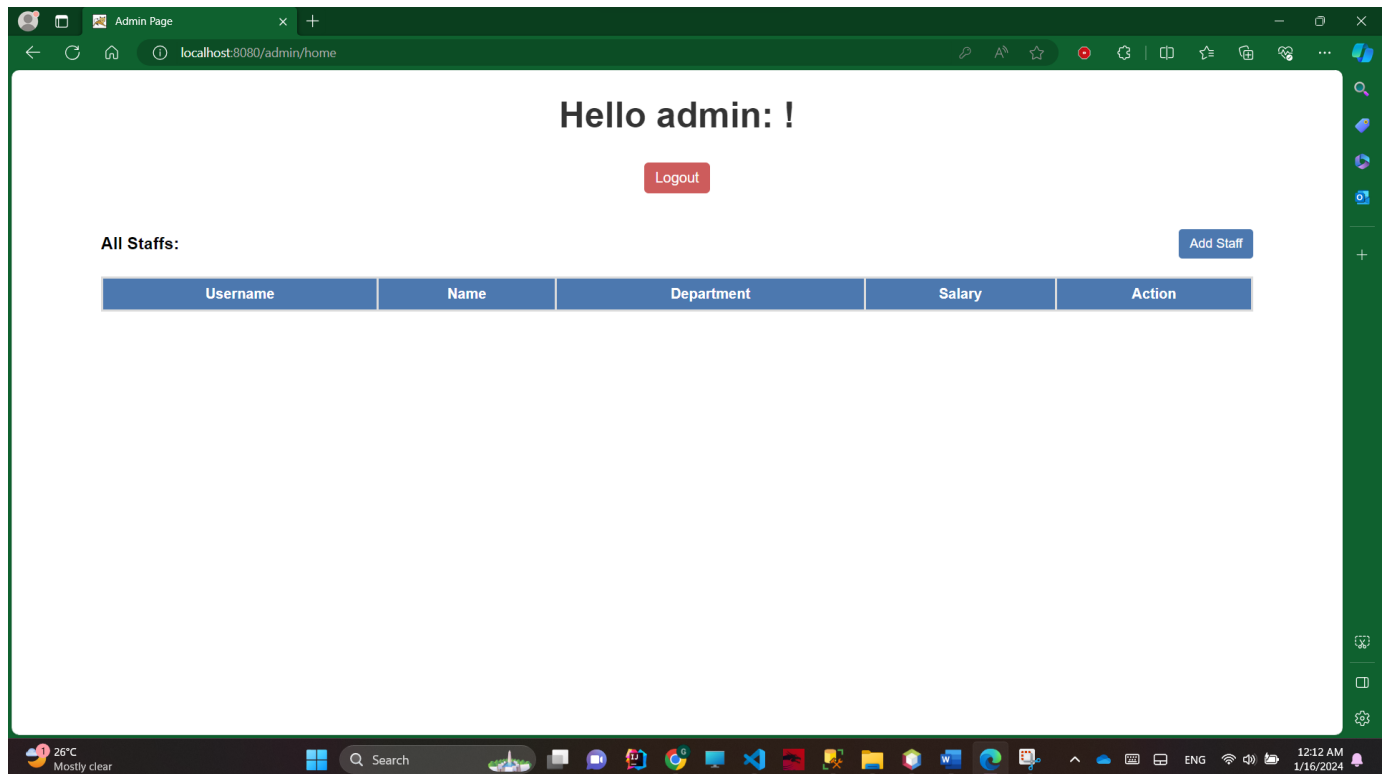
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="6.0" xmlns="https://jakarta.ee/xml/ns/jakartaee" xmlns:xsi="h
  <servlet>
    <servlet-name>Main</servlet-name>
    <servlet-class>controller.Main</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>Login</servlet-name>
    <servlet-class>controller.Login</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>Logout</servlet-name>
    <servlet-class>controller.Logout</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>Register</servlet-name>
    <servlet-class>controller.Register</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>CustomerInformation</servlet-name>
    <servlet-class>controller.CustomerInformation</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>CustomerHome</servlet-name>
```

*Figure 34. Part of web.xml*

#### 9.1.7. Show Application:



*Figure 35. Login page*



*Figure 36. Admin main page*

The screenshot shows a web browser window with the title "Add Staff Page" and the URL "localhost:8080/admin/add\_staff?". The main content area is a light gray rectangle. In the center of this area is a light blue rectangular form titled "Add Staff". The form contains the following fields and values:

- Username:** A text input field containing the text "nguyen".
- Password:** A password input field with masked characters "\*\*\*\*\*".
- Name:** A text input field containing the text "Nguyen".
- Department:** A text input field containing the text "Analys".
- Salary:** A text input field containing the text "50000".

At the bottom right of the form is a blue button labeled "Add Staff". The browser's taskbar at the bottom shows the Windows logo, a search bar, and various application icons. The system tray on the right indicates a temperature of 26°C, "Mostly clear" weather, and the time 12:14 AM on 1/16/2024.

*Figure 37. Add staff page*



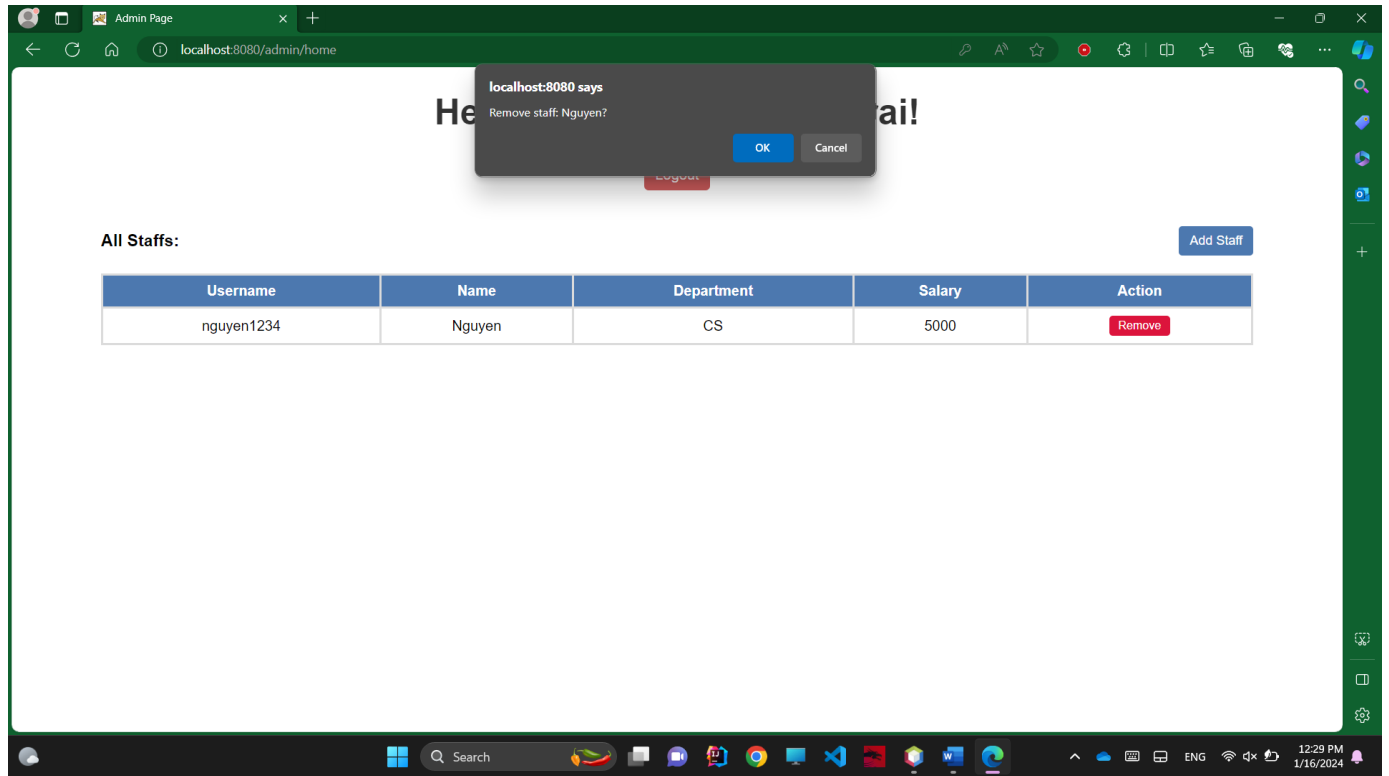
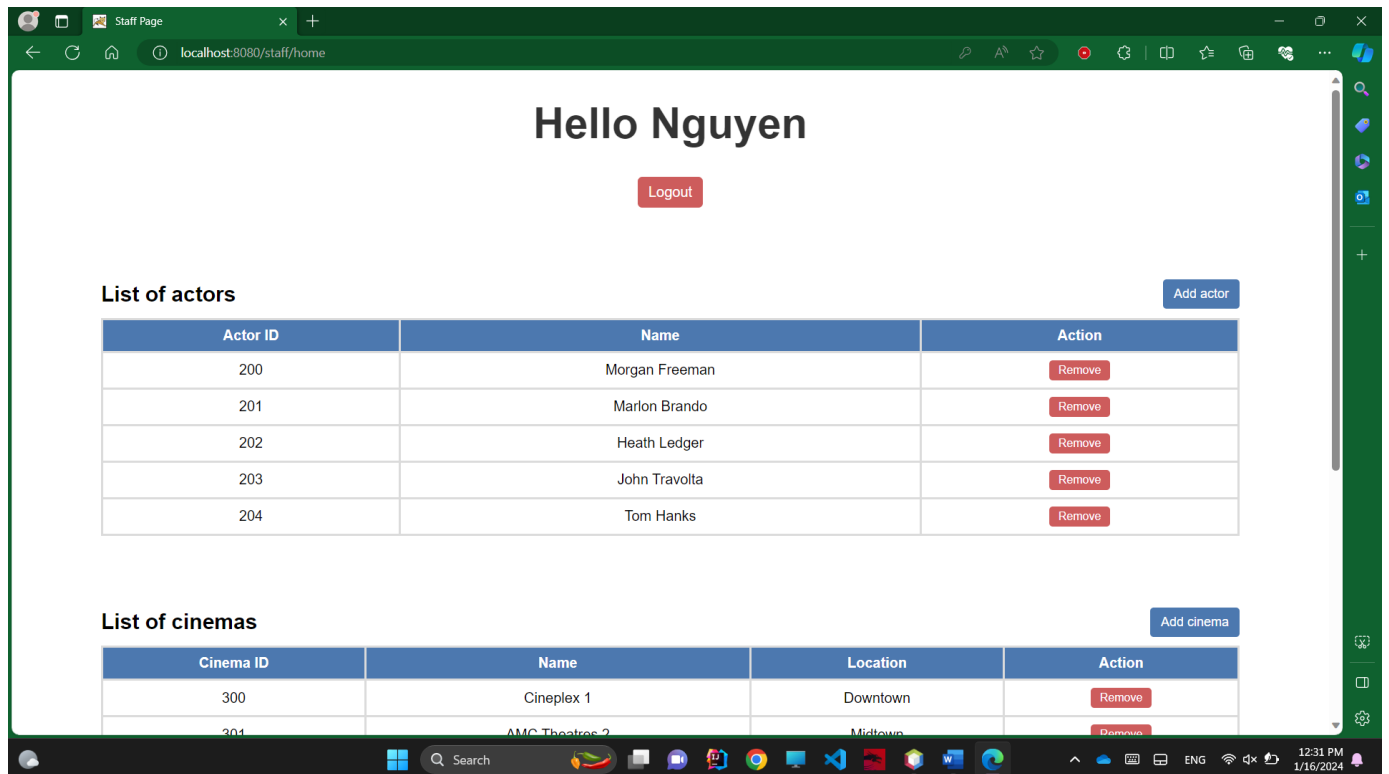


Figure 38. Remove staff page



*Figure 39. Staff main page*

**Add Movie**

Title:  Please fill out this field.

Director:

Category:

Age Require:

Actors:

*Figure 40. Add movie page*

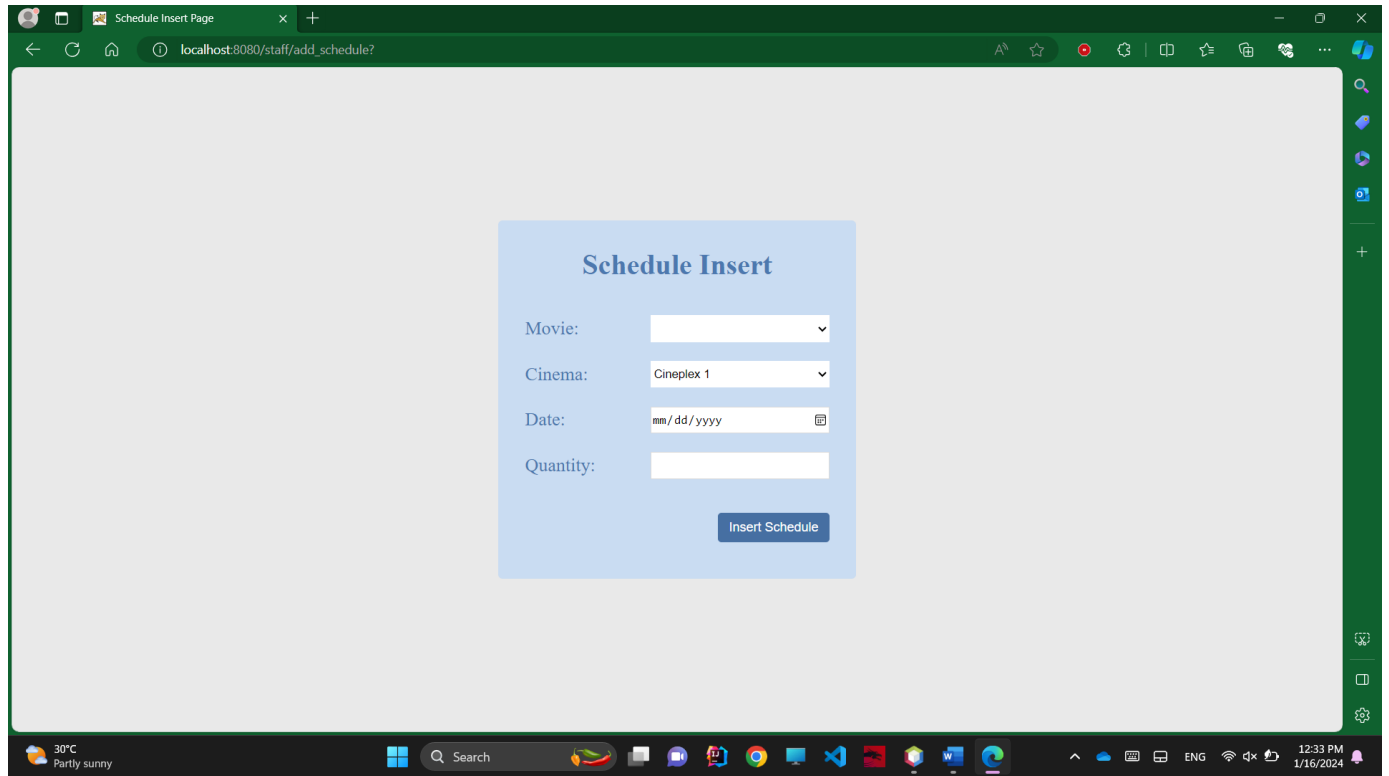


Figure 41. Add shedule page

Staff Page

localhost:8080/staff/home

Guest

List of actors

Add actor

Actor ID	Name	Action
200	Actor 1	Remove
201	Actor 2	Remove
202	Actor 3	Remove
203	Actor 4	Remove
204	Actor 5	Remove

List of cinemas

Add cinema

Cinema ID	Name	Location	Action
300	Cinema A	HCM	Remove
301	Cinema B	Hanoi	Remove
302	Cinema C	Thu Duc	Remove

List of movies

Add movie

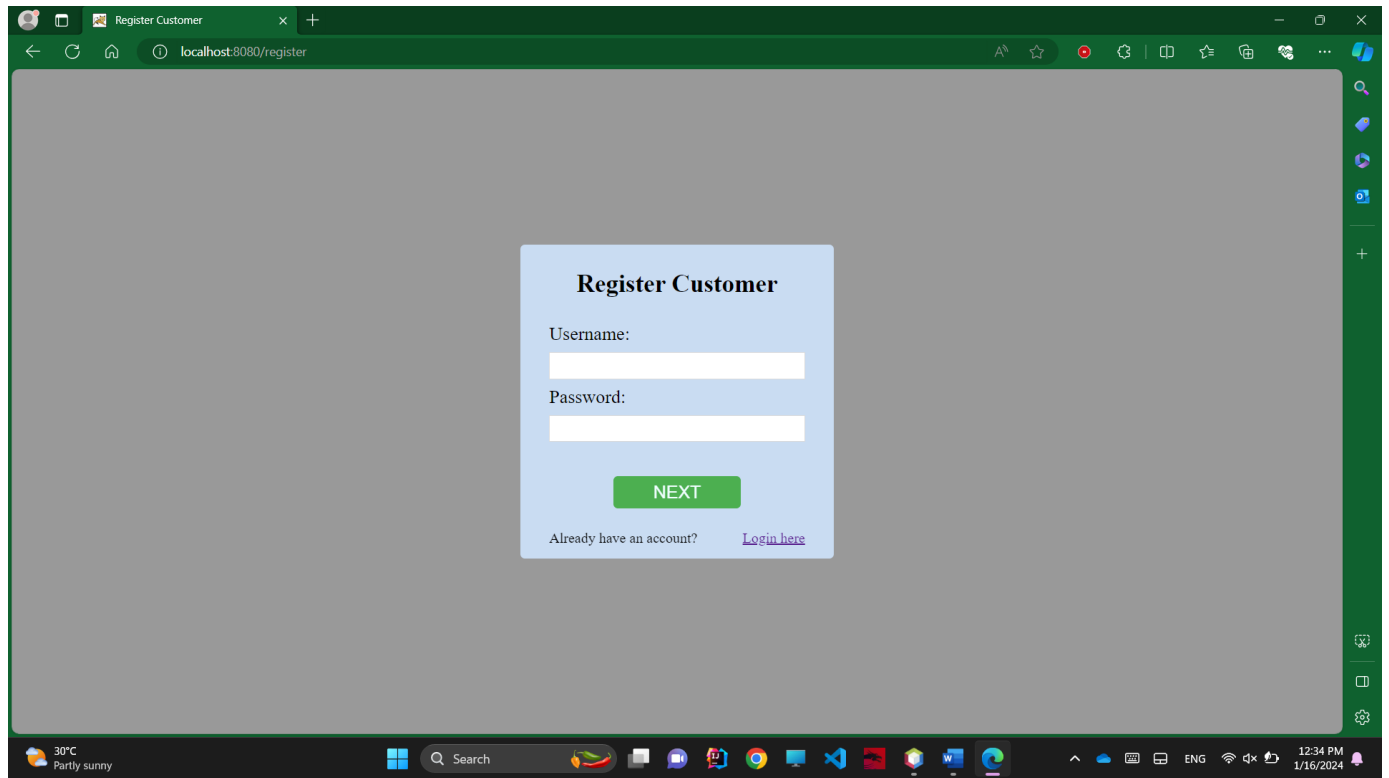
Movie ID	Title	Director	Category	Age Require	Actors	Action
100	Movie 1	Vinh	Action	18	[Actor 1, Actor 2]	Remove

List of schedules

Add schedule

Schedule ID	Movie	Cinema	Date	Quantity	Action
-------------	-------	--------	------	----------	--------

*Figure 42. Result after add movie and schedule*



The image shows a web browser window with a single tab titled 'Register Customer'. The address bar displays 'localhost:8080/register'. The main content area has a light gray background. Centered on the page is a light blue rectangular form with the title 'Register Customer' in bold black text. Below the title are two input fields: 'Username:' followed by a white text box, and 'Password:' followed by a white text box. A green button with the text 'NEXT' is positioned below the password field. At the bottom of the form, the text 'Already have an account?' is followed by a purple link labeled 'Login here'. The browser's taskbar at the bottom shows the Windows logo, a search bar, and various application icons. The system tray on the right indicates a temperature of 30°C, 'Partly sunny' weather, and the time 12:34 PM on 1/16/2024.

Register Customer

Username:

Password:

NEXT

Already have an account? [Login here](#)

*Figure 43. Register Page*

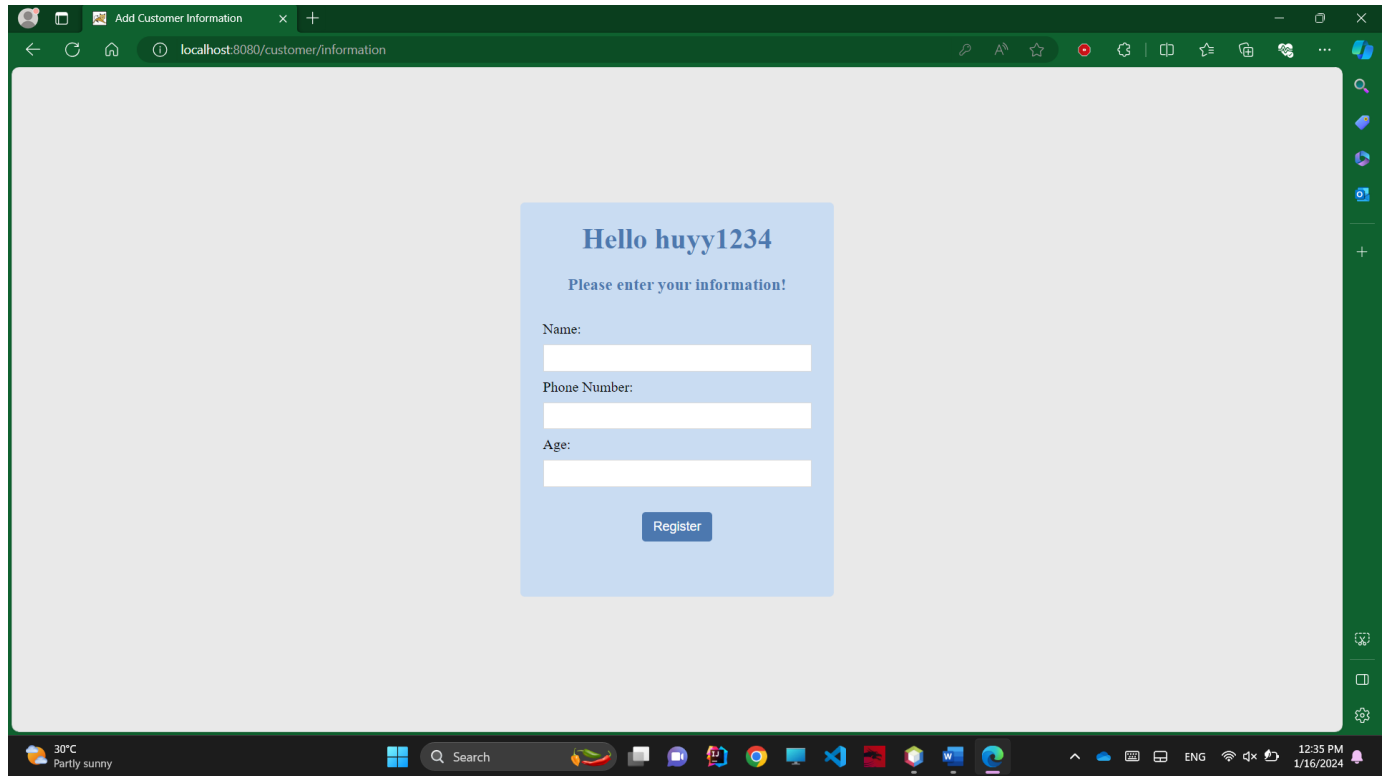


Figure 44. Add more information for registering customer

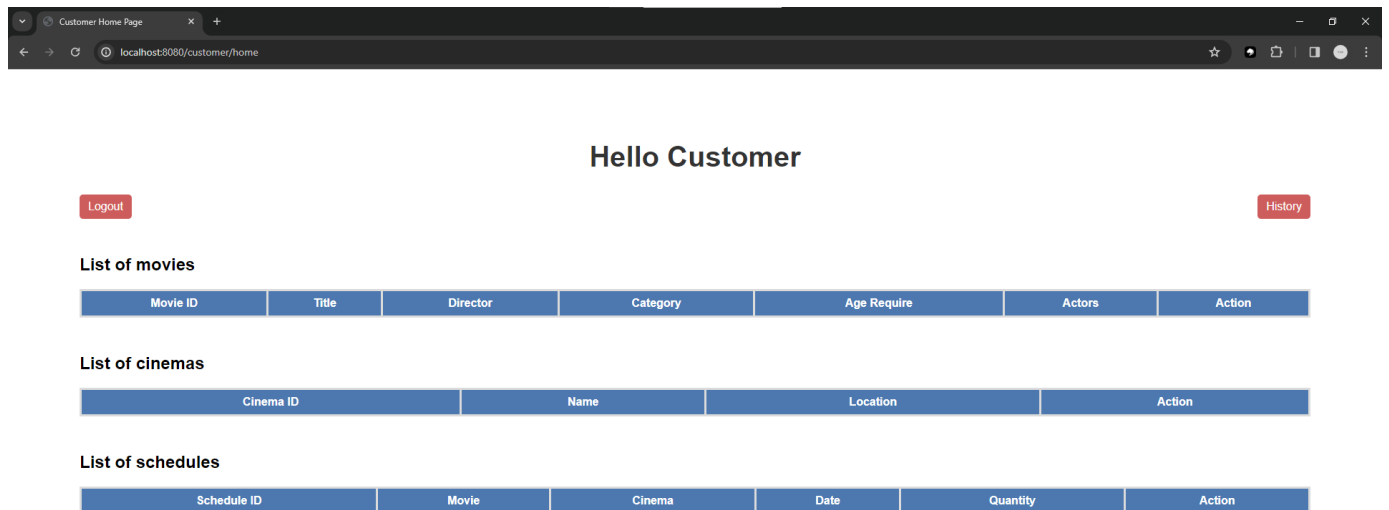


Figure 45. Customer main screen when no movie or schedule

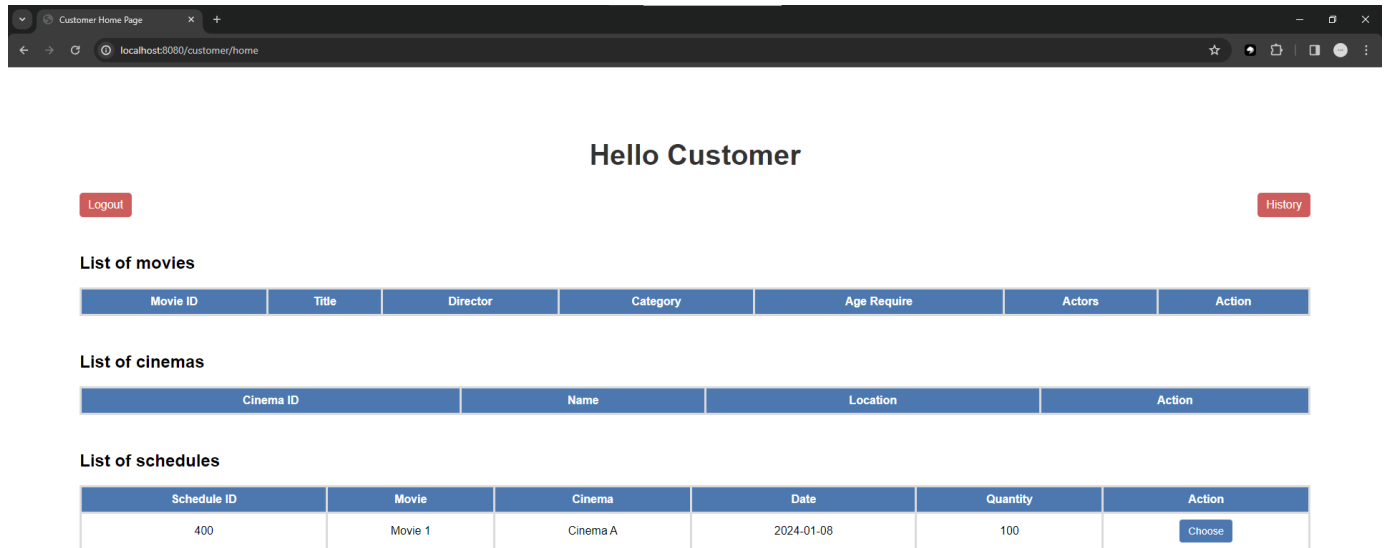


Figure 45. Customer main screen when added schedule

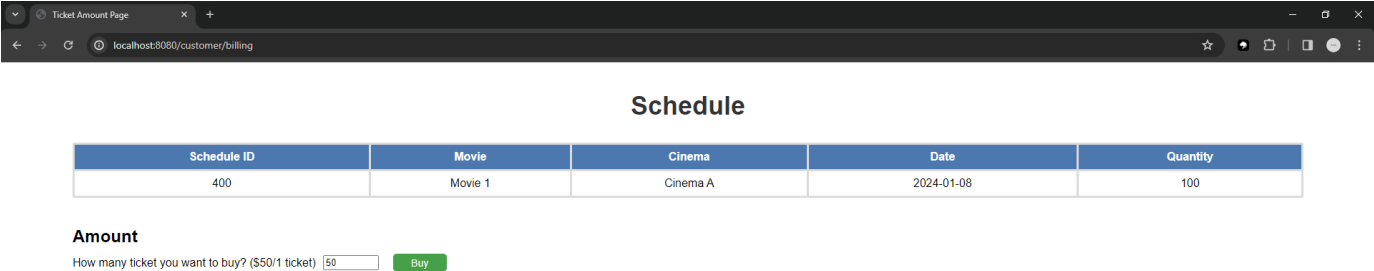
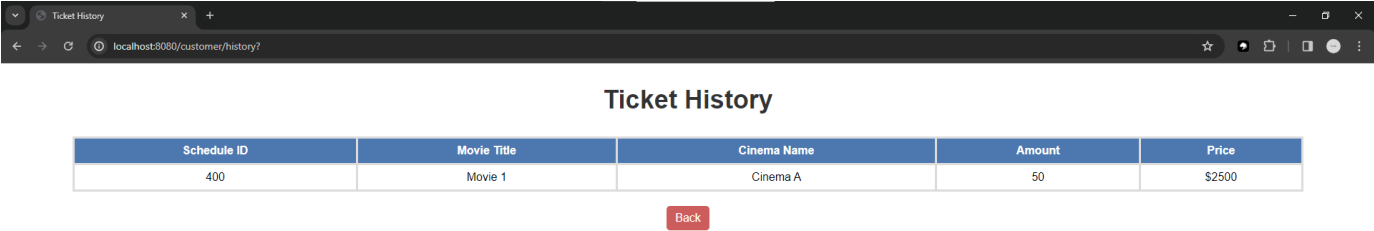


Figure 46. Booking screen



*Figure 47. History screen*



## **10. Conclusion**

In conclusion, the Movie System website successfully leverages a dynamic combination of CSS and JSP for an engaging and user-friendly interface, while utilizing servlets to seamlessly manage control logic. The website caters to a diverse user base, offering distinct functionalities for customers, staff, and managers. Customers enjoy a streamlined experience with the ability to effortlessly book tickets, while staff members benefit from the convenience of adding or removing movies, schedules, and cinema details. Additionally, the managerial role is empowered with the capability to efficiently manage staff by adding or removing members.

Overall, the Movie System website demonstrates a robust and well-integrated platform, ensuring a comprehensive and efficient cinema management experience for all stakeholders.

### **3.REFERENCE**

<https://www.youtube.com/watch?v=DzYyzmP4m5c>

<https://www.youtube.com/watch?v=OuBUUkQfBYM>

<https://phoenixnap.com/kb/install-tomcat-windows>

<https://www.w3schools.com/css/>

<https://www.w3schools.com/html/>

[https://developer.mozilla.org/en-US/docs/Learn/CSS/First\\_steps](https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps)

<https://www.itarian.com/ticketing-system/online-movie-ticketing-system.php>

