



Faculty of Computing (23.1)

# **PUSL 2024 Software Engineering 2 (24/AU/M)**

## **Coursework Report**

ABC Cinema

Group G

|   | <b>Student ID (Plymouth)</b> | <b>Student Name (As appeared on DLE)</b> |
|---|------------------------------|--|
| 1 | 10953243                     | Bedde Samarasinghe (Leader)              |
| 2 | 10952466                     | Hewa Gamage                              |
| 3 | 10953326                     | Yuhas Gamage                             |
| 4 | 10952509                     | Amarasinghe Senevirathne                 |
| 5 | 10903095                     | Liyana Gunawardana                       |
| 6 | 10899703                     | Dasith Silva                             |
| 7 | 10952758                     | Hiddiyadura Abeynayake                   |
| 8 | 10820754                     | Herath Vimalaranga                       |

# Table of Content

|  |    |
|--|----|
| Table of Content                           | 2  |
| Introduction                               | 4  |
| Identified Processes in ABC Cinema         | 4  |
| Customer Processes :                       | 4  |
| Admin / Manager Processes :                | 5  |
| Use Case Diagram                           | 6  |
| Class Diagram                              | 7  |
| Sequence Diagram for Booking a Reservation | 8  |
| ER Diagram                                 | 9  |
| Samples of Developed System (Screenshots)  | 10 |
| Navigation Bar                             | 10 |
| Home Page                                  | 10 |
| Movies Page                                | 11 |
| Booking Page                               | 13 |
| Seat Selection Page                        | 15 |
| Payment Page                               | 17 |
| Payment Success Page                       | 20 |
| Feedbacks Page                             | 21 |
| Login Page                                 | 23 |
| User Register Page                         | 25 |
| Admin Login Page                           | 26 |
| Authentication Servlet                     | 27 |
| Admin Dashboard                            | 28 |
| Movie Management                           | 29 |
| Ticket Management                          | 31 |
| User Management                            | 32 |
| Feedback Management                        | 34 |

|                         |    |
|-------------------------|----|
| Admin Feedback Servlets | 35 |
| Challenges              | 36 |
| Conclusion              | 36 |
| References              | 36 |
| Work Breakdown Matrix   | 38 |
| GitHub Repository Link  | 38 |

# Introduction

As our coursework has mentioned we have created a web-application for ‘ABC Cinema’ using Java Servlets, JSP, HTML, CSS, JavaScripts, and MYSQL Server database. In this web-application we have upgraded ‘ABC Cinema’s obsolete website to the revolutionised web-application including : Online Ticket reservation of tickets avoiding the booking collisions, Online payment handling, Visualisation of the customer feedbacks, Email payment notification (payment confirmation) service, with interactive modern user interface. We have started developing, after the designing of all the UML diagrams (Use Case diagram, Class diagram, Sequence diagram, and the ER diagram) and in below all the diagrams are explained: how this web-application’s system is designed.

## Identified Processes in ABC Cinema

### Customer Processes :

- **Browsing movies and showtimes.**

By clicking ‘Movies’ in navigation bar, users can browse all the available movies, and upcoming movies.

- **Creating and managing user accounts.**

By clicking ‘Login’ button in navigation bar, users can login if they have existing account in ‘ABC Cinema’ or if a new user comes to the login page, they can click ‘Register here’ button to register to the ‘ABC Cinema’. After the registration user will redirected to the ‘Login’ page.

- **Booking seats and paying for tickets.**

Users can book tickets after the selection of a movie from ‘Movies’ page. Users can select show date, show time from available times, and Number of tickets. Then clicking ‘Continue to Seat Selection’ button, will navigate the user into the seat structure map to selecta seat from available seats. By clicking ‘Proceed to Payment’ button, user will redirect to payment completion page and user can add their Visa, Master or any Credit/Debit card to Complete the payment. Then, the payment receipt will received to their registered Email address.

- **Posting feedbacks.**

By clicking, ‘Feedbacks’ in navigation bar, Users can add a feedback with their rating.

- **Viewing booked tickets.**

Users can view their booked tickets by clicking ‘My Bookings’ in navigation bar.

## **Admin / Manager Processes :**

- **Adding / Editing and Deleting movie details.**

For a admin, We have developed an admin panel to add movies, edit movie details, and delete movies from the movie list. Those admin functions are updating real-time database’s ‘Movies’ table.

- **Ticket Management.**

In ticket management, admins can click ‘Add New Tickets’ button to add tickets to a movie declaring price, Quantity, and Status of the availability. An admins can use ‘Edit Ticket’ icon in Actions to edit the ticket details in a movie. As well as, admins can delete tickets using ‘Delete icon’ in ‘Actions’.

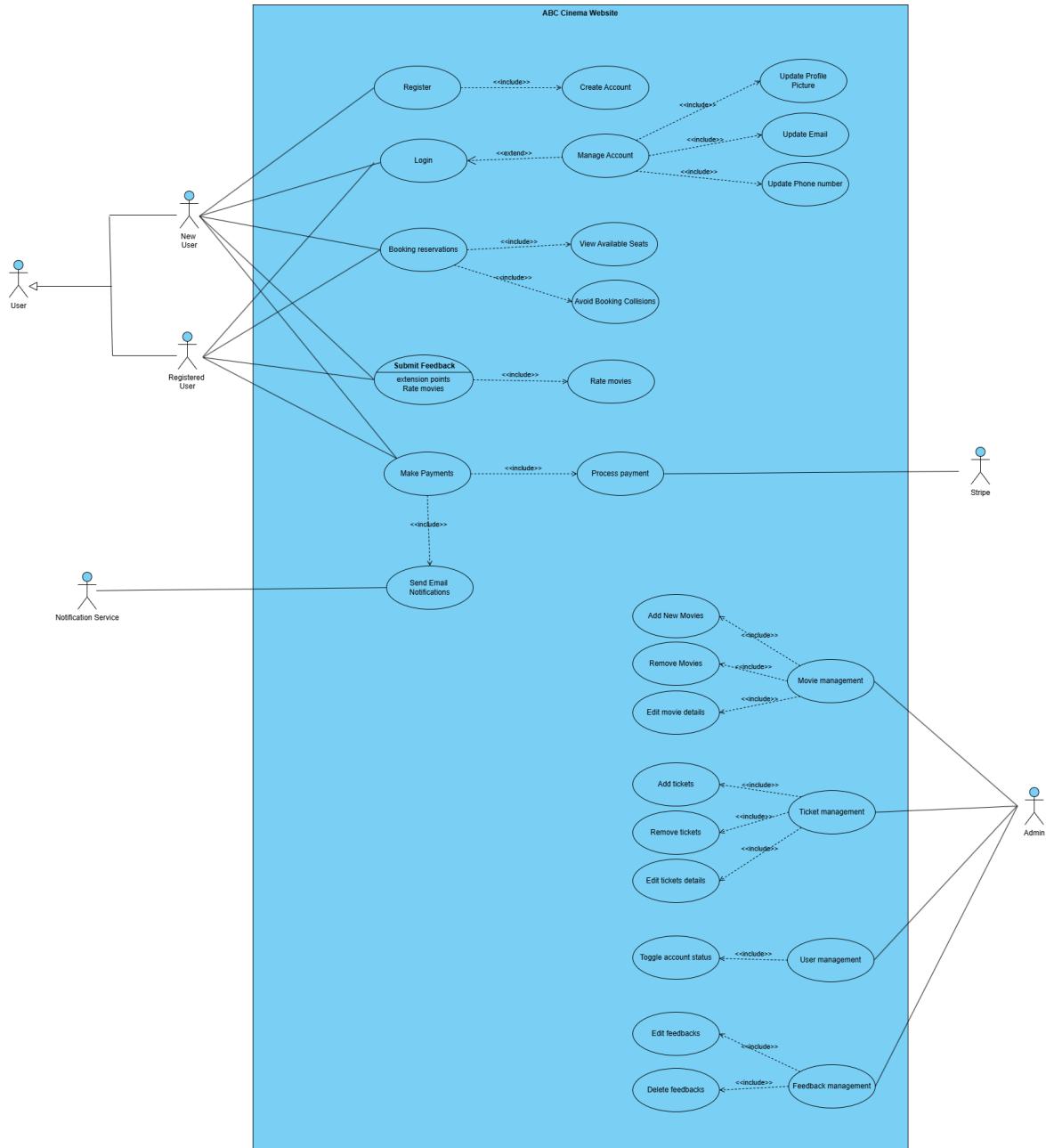
- **User Management**

In user management, admins can view all the registered users details except their passwords. An admin can change the user profiles ‘Status’ by clicking power button in ‘Actions’ column. When the user profile in ‘Inactive’ users cannot login to the system using their credentials until an admin change their status.

- **Feedback Management**

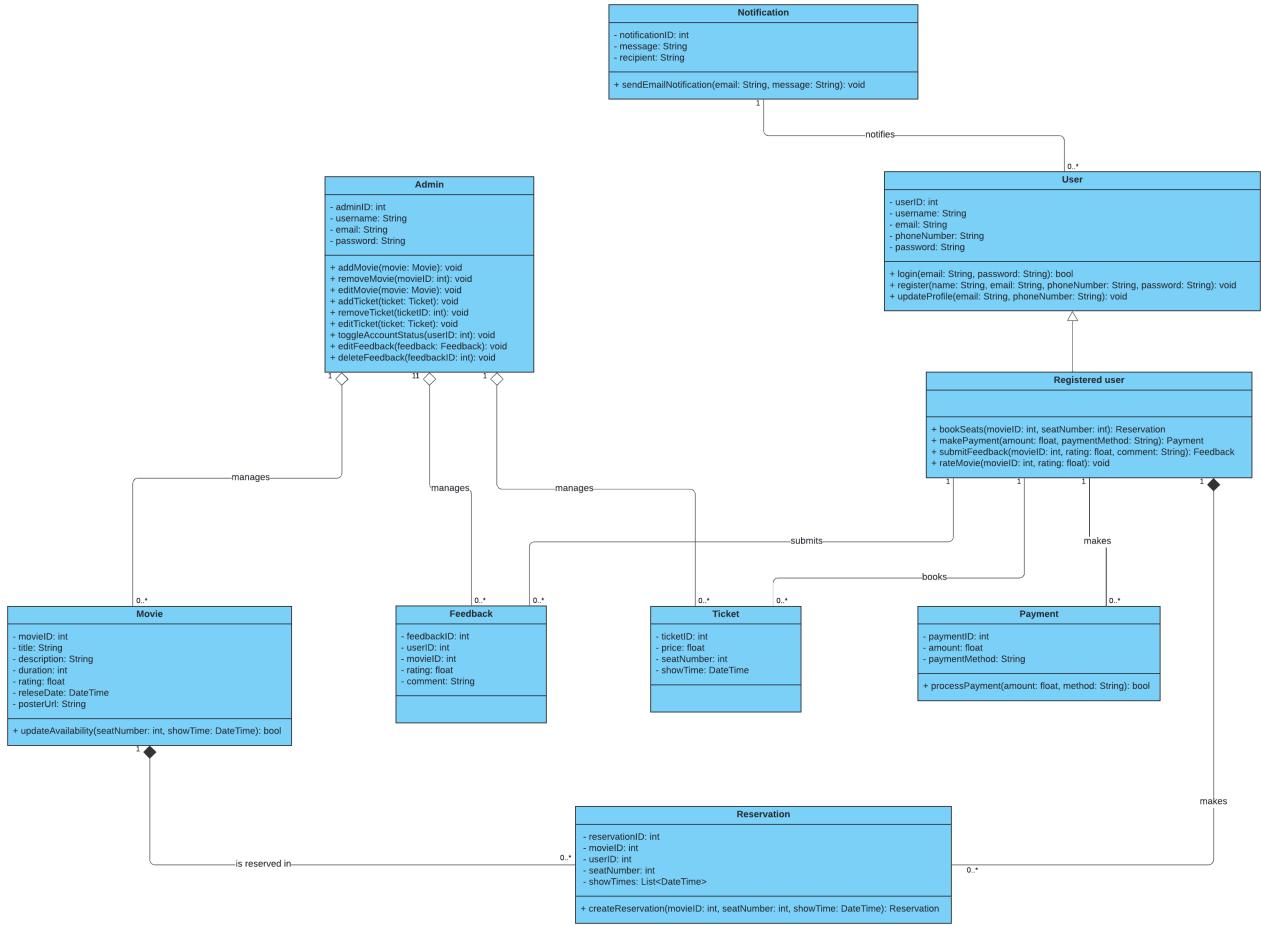
Admins can view all the user submitted feedbacks in ‘Feedback Management’ option and if there is a feedback that the ‘ABC Cinema’ want to delete or edit, we have added ‘Edit’ and ‘Delete’ option to the feedbacks.

# Use Case Diagram



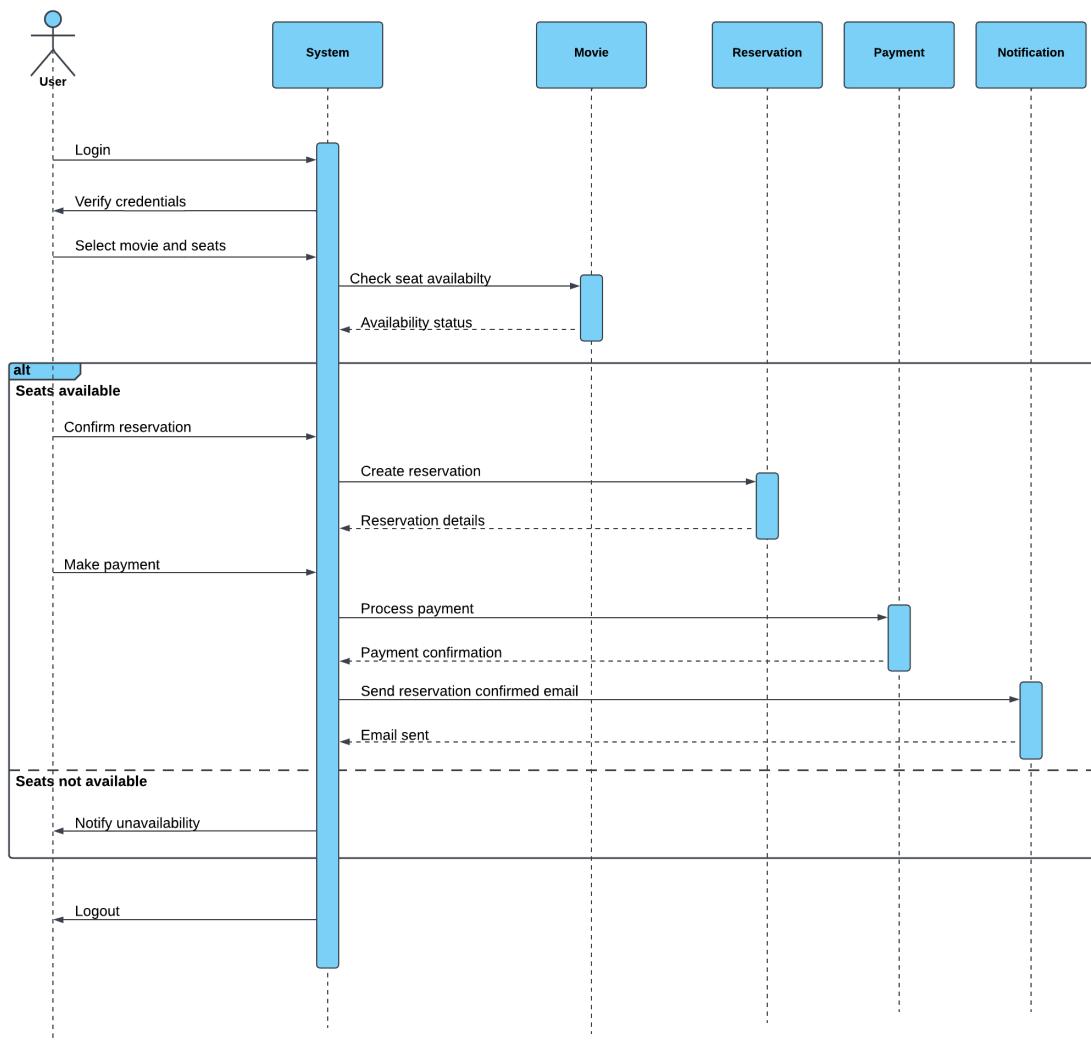
The Use Case Diagram represents core functionalities in our system. There are 4 actors. The admin manages core operations such as movies, theatres, showtimes, ticket reservations, and user feedback. Registered Users can browse movies, book tickets, make payments, provide feedback, and manage their profiles, while Guest Users may have limited access, such as browsing movies and showtimes. Notification service handle the sending notification parts and in our case Stripe is working as the payment gateway.

# Class Diagram



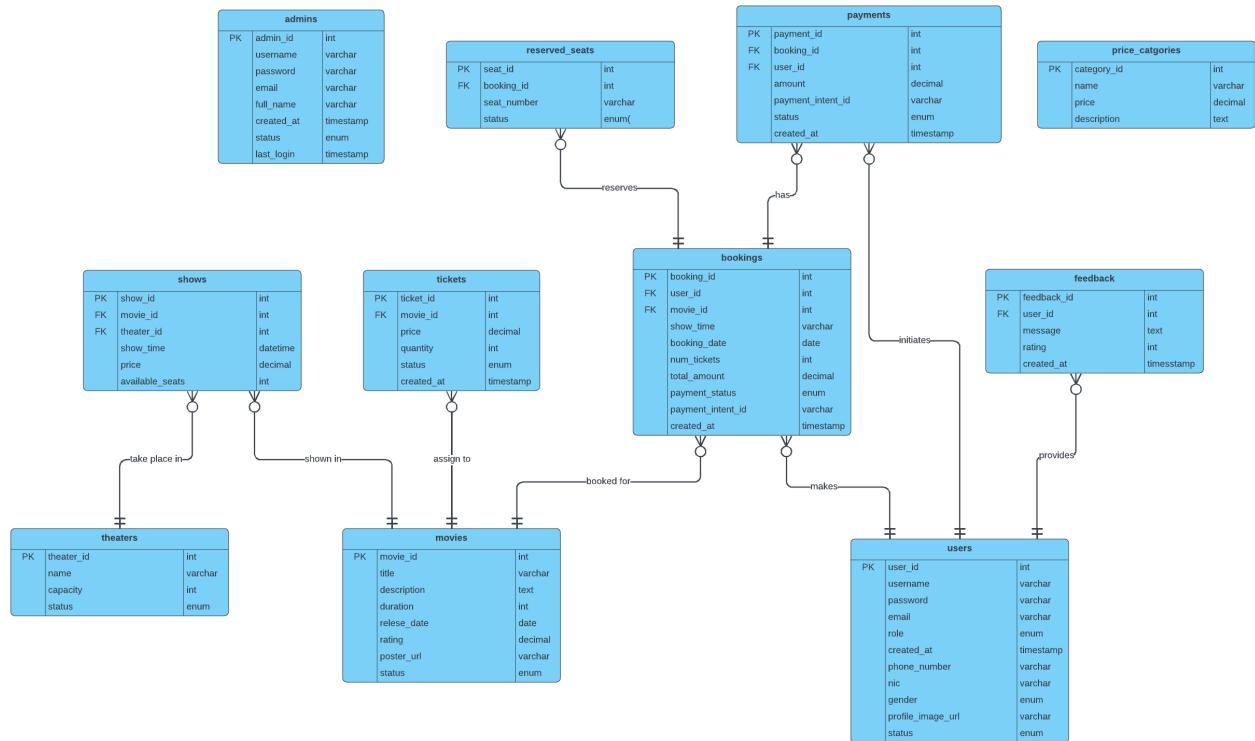
This class diagram represents the structure of your system. It's like a blueprint to the system. It identifies the key classes, which include Admin, User, Movie, Theater, Show, Booking, Payment, Ticket, ReservedSeat, and Feedback, each encapsulating relevant attributes and methods. System interactions and operations are managed by the Admin and User classes, with the Admin in charge of system control and the User in charge of reservations, payments, and feedback. ReservedSeat keeps track of assigned seating, while Booking, Payment, and Ticket classes manage reservations, payments, and ticketing procedures. The links between processes are established via relationships like dependencies and associations between these classes, which guarantee a coherent and useful system architecture.

# Sequence Diagram for Booking a Reservation



The user-system interaction during the movie ticket booking process is represented in the sequence diagram. The system verifies login information and seat availability once the user chooses a movie, showtime, and seats. The Payment Gateway (in our case Stripe) is used to handle the user's payment when seats availability has been confirmed. When the payment is successful, the system completes the reservation and emails the user a confirmation along with the ticket. This ensures a smooth and well-organized reservation procedure.

# ER Diagram



The database structure of our system, which includes things like admins, users, movies, theatres, shows, bookings, payments, tickets, reserved seats, and feedback, is represented in the ER Diagram. It describes the characteristics and connections between various entities, including the fact that Movies include details about the film, theaters store details about the movie theater, shows link Movies to Theaters for certain showtimes, and Admins and Users store login credentials. Reserved seats ensure seat availability, payments keep track of financial transactions, bookings monitor reservations, and feedback keeps track of user feedback. To ensure data consistency throughout the system, foreign keys connect Users, Movies, and Bookings.

# Samples of Developed System (Screenshots)

## Navigation Bar



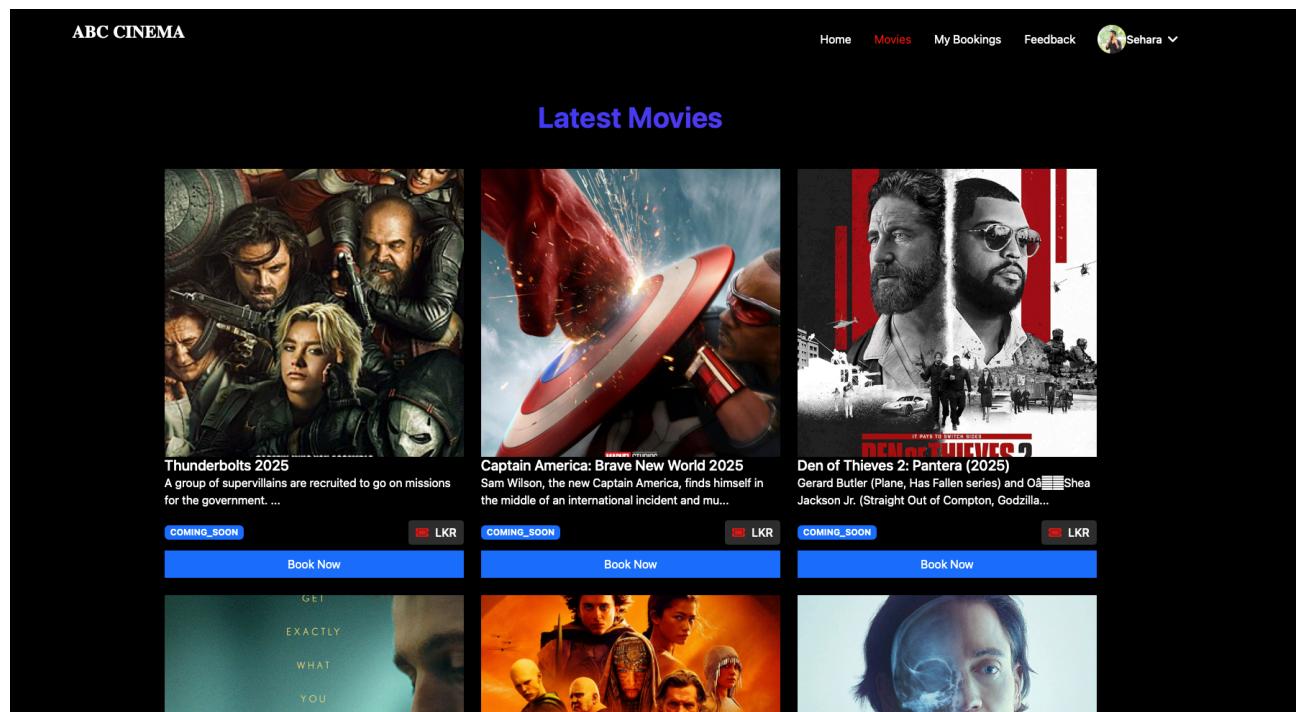
In our web application the navigation bar is designed to provide the users with clear access to the core functionalities in our system. The navigation bar includes sections such as home, movies, my bookings, feedback a login. After the user logs in to the system the login button in the navigation button replaces with profile image of the user along with the name of the user. Clicking on this shows a dropdown list including my profile, which allows users to edit their profiles, my bookings lead the users to my booking page and the logout button to log out from the system.

## Home Page

A screenshot of the main homepage of the ABC Cinema website. The top navigation bar is identical to the one shown above, with "ABC CINEMA" on the left and a menu with "Home" (red), "Movies", "My Bookings", "Feedback", and "Login". Below the navigation, there is a large promotional image of a movie theater interior with red seats and a grand screen. To the left of the image, the text "Experience the Magic of Cinema Today" is displayed in blue, with a smaller paragraph below it. At the bottom of the page, there is a call-to-action section with the text "Effortlessly Book Your Tickets and Reserve Your Seats Online" in blue, followed by three icons with corresponding descriptions: a book icon for "Simple Steps to Secure Your Spot for the Latest Movies", a person icon for "Enjoy a Seamless Experience with Our User-Friendly Interface", and a ticket icon for "Receive Instant Confirmation and Access Your Tickets Anytime".

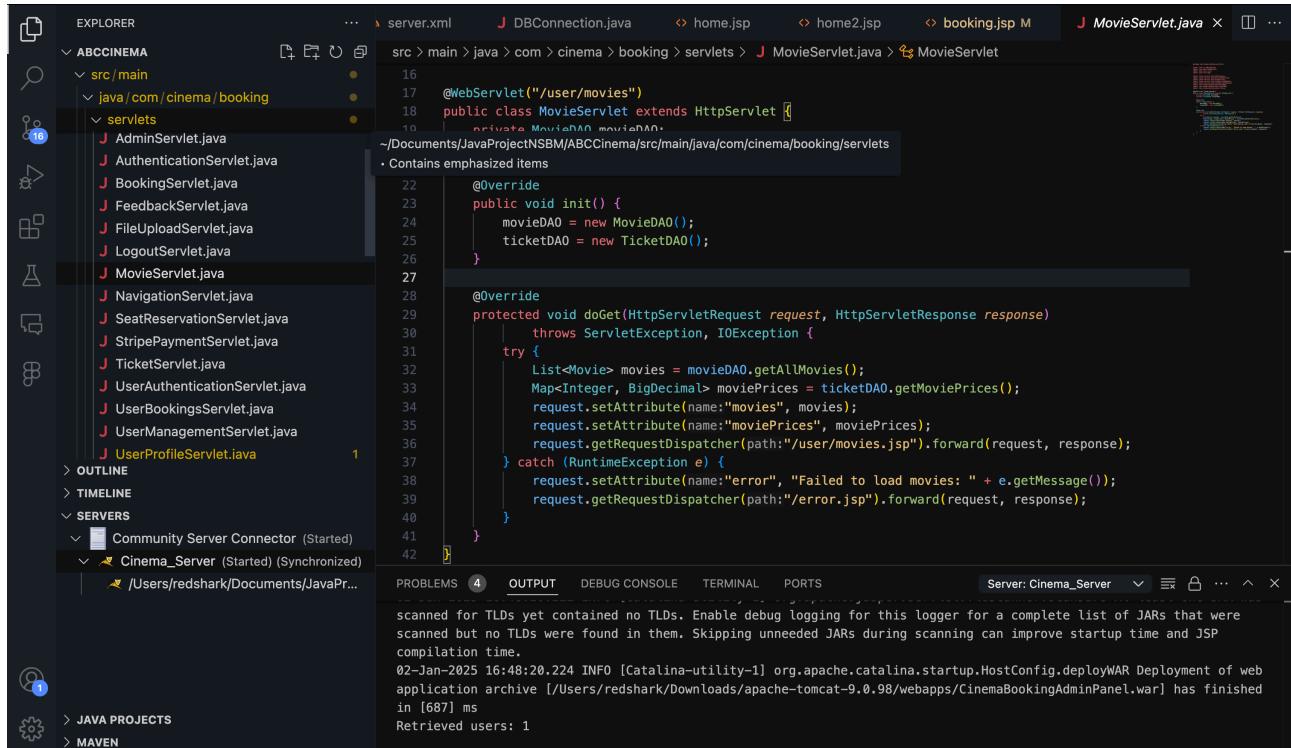
This acts as the main entry point of the website. This leads users to the landing page of our web application. This page includes descriptions of the cinema hall and details about the web applications functions like easy booking process. Unregistered users can view only this page and can register to the system using the login button and the book button. Unless they can't access the other pages.

## Movies Page



This page represents the movies catalog of our cinema hall. This includes upcoming movies and now showing movies. Users can book tickets according to their choice of movies on this page. Users have to login or register to the system to see this page. The book now button in the movie cards leads user to the ticket booking page.

## Movie Page Servlet



The screenshot shows an IDE interface with the following details:

- EXPLORER** panel on the left lists files under the project "ABCCINEMA":
  - src/main/java/com/cinema/booking
  - src/main/java/com/cinema/booking/servlets
  - AdminServlet.java
  - AuthenticationServlet.java
  - BookingServlet.java
  - FeedbackServlet.java
  - FileUploadServlet.java
  - LogoutServlet.java
  - MovieServlet.java
  - NavigationServlet.java
  - SeatReservationServlet.java
  - StripePaymentServlet.java
  - TicketServlet.java
  - UserAuthenticationServlet.java
  - UserBookingsServlet.java
  - UserManagementServlet.java
  - UserProfileServlet.java
- EDITOR** panel shows the code for **MovieServlet.java**:

```
16
17     @WebServlet("/user/movies")
18     public class MovieServlet extends HttpServlet {
19         private MovieDAO movieDAO;
20
21         @Override
22         public void init() {
23             movieDAO = new MovieDAO();
24             ticketDAO = new TicketDAO();
25         }
26
27         @Override
28         protected void doGet(HttpServletRequest request, HttpServletResponse response)
29             throws ServletException, IOException {
30             try {
31                 List<Movie> movies = movieDAO.getAllMovies();
32                 Map<Integer, BigDecimal> moviePrices = ticketDAO.getMoviePrices();
33                 request.setAttribute(name:"movies", movies);
34                 request.setAttribute(name:"moviePrices", moviePrices);
35                 request.getRequestDispatcher(path:"/user/movies.jsp").forward(request, response);
36             } catch (RuntimeException e) {
37                 request.setAttribute(name:"error", "Failed to load movies: " + e.getMessage());
38                 request.getRequestDispatcher(path:"/error.jsp").forward(request, response);
39             }
40         }
41     }
```
- PROBLEMS** tab in the bottom navigation bar shows 4 errors.
- OUTPUT** tab shows deployment logs:

```
scanned for TLDs yet contained no TLDs. Enable debug logging for this logger for a complete list of JARs that were scanned but no TLDs were found in them. Skipping unneeded JARs during scanning can improve startup time and JSP compilation time.
02-Jan-2025 16:48:20.224 INFO [Catalina-utility-1] org.apache.catalina.startup.HostConfig.deployWAR Deployment of web application archive [/Users/redshark/Downloads/apache-tomcat-9.0.98/webapps/CinemaBookingAdminPanel.war] has finished
in [687] ms
Retrieved users: 1
```

The **MovieServlet.java** class is a part of the cinema booking system and it is designed to handle requests related to Retrieving and displaying a list of movies available for booking.

## Movie Page SQL Queries

In above picture, that's the SQL query that creates the movies table and the below 'Result Grid' shows the all available movies that we have added in Web-application.

Local instance 3306 - Warning - not supported

Schemas

```

1 • CREATE TABLE `movies` (
2     `movie_id` int NOT NULL AUTO_INCREMENT,
3     `title` varchar(100) NOT NULL,
4     `description` text,
5     `duration` int DEFAULT NULL,
6     `release_date` date DEFAULT NULL,
7     `rating` decimal(2,1) DEFAULT NULL,
8     `poster_url` varchar(255) DEFAULT NULL,
9     `status` enum('NOW_SHOWING','COMING_SOON','ENDED') DEFAULT 'COMING_SOON',
10    PRIMARY KEY (`movie_id`)
11 ) ENGINE=InnoDB AUTO_INCREMENT=16 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
12 • SELECT * FROM cinema_db.movies;

```

Result Grid

| movie_id | title  | description  | duration | release_date | rating | poster_url  | status      |
|----------|--|--|----------|--------------|--------|---|-------------|
| 1        | Red One 2024                                 | After Santa Claus is kidnapped, the North Pole...    | 123      | 2024-11-15   | 4.8    | https://res.cloudinary.com/dzyk7ias/image/uplo... | NOW_SHOWING |
| 2        | Dune: Part Two 2024                          | Paul Atreides unites with the Fremen while on a...   | 166      | 2025-01-03   | 4.9    | https://res.cloudinary.com/dzyk7ias/image/uplo... | NOW_SHOWING |
| 6        | Captain America: Brave New World 2025        | Sam Wilson, the new Captain America, finds hi...     | 100      | 2025-02-14   | 0.0    | https://res.cloudinary.com/dzyk7ias/image/uplo... | COMING_SOON |
| 7        | Thunderbolts 2025                            | A group of supervillains are recruited to go on m... | 0        | 2025-05-02   | 0.0    | https://res.cloudinary.com/dzyk7ias/image/uplo... | COMING_SOON |
| 8        | Den of Thieves 2: Pantera (2025)             | Gerard Butler (Plane, Has Fallen series) and Ol...   | 124      | 2025-01-10   | 0.0    | https://res.cloudinary.com/dzyk7ias/image/uplo... | COMING_SOON |
| 9        | Survive 2024                                 | In SURVIVE, a woman and her loving husband calc...   | 90       | 2024-01-10   | 4.6    | https://res.cloudinary.com/dzyk7ias/image/uplo... | NOW_SHOWING |
| 10       | Plan 2024                                    | The plan of business magnate and Princeton stud...   | 109      | 2024-01-20   | 3.7    | https://res.cloudinary.com/dzyk7ias/image/uplo... | PENDING     |
| 11       | Black Diamond 2023                           | After moving into a secluded cabin, a young wo...    | 89       | 2023-01-10   | 4.5    | https://res.cloudinary.com/dzyk7ias/image/uplo... | ENDED       |
| 12       | Babyspin 2024                                | A high-powered CEO puts her career and family...     | 114      | 2025-01-10   | 4.9    | https://res.cloudinary.com/dzyk7ias/image/uplo... | COMING_SOON |
| 13       | The Loxax 2012 (Christmas special)           | A 12-year-old boy searches for the one thing tha...  | 86       | 2012-01-01   | 4.7    | https://res.cloudinary.com/dzyk7ias/image/uplo... | NOW_SHOWING |
| 14       | Don't Die: The Man Who Wants to Live... 2022 | Explores a man's quest for immortality and the i...  | 81       | 2025-01-01   | 4.8    | https://res.cloudinary.com/dzyk7ias/image/uplo... | NOW_SHOWING |
| 15       | Avatar: The Way of Water 2022                | Jake Sully lives with his newfound family formed...  | 112      | 2022-05-12   | 5.0    | https://res.cloudinary.com/dzyk7ias/image/uplo... | NOW_SHOWING |
| HULL     | HULL   | HULL   | HULL     | HULL         | HULL   | HULL  | HULL        |

Action Output

| Time       | Action                                       | Response           | Duration / Fetch Time   |
|------------|--|--------------------|-------------------------|
| 3 13:46:58 | SELECT * FROM cinema_db.users LIMIT 0, 1000  | 1 row(s) returned  | 0.000090 sec / 0.000... |
| 4 17:14:19 | SELECT * FROM cinema_db.movies LIMIT 0, 1000 | 12 row(s) returned | 0.00064 sec / 0.000...  |

Query Completed

## Booking Page

**CAPTAIN AMERICA: BRAVE NEW WORLD 2025**

🕒 0 mins   ★ 0.0/5



MARVEL STUDIOS  
**CAPTAIN AMERICA**  
IN THEATERS FEBRUARY 14

🕒 Duration: 0 mins   ★ Rating: 0.0/5

**Book Your Tickets**

Show Date:

Show Time:

Number of Tickets:

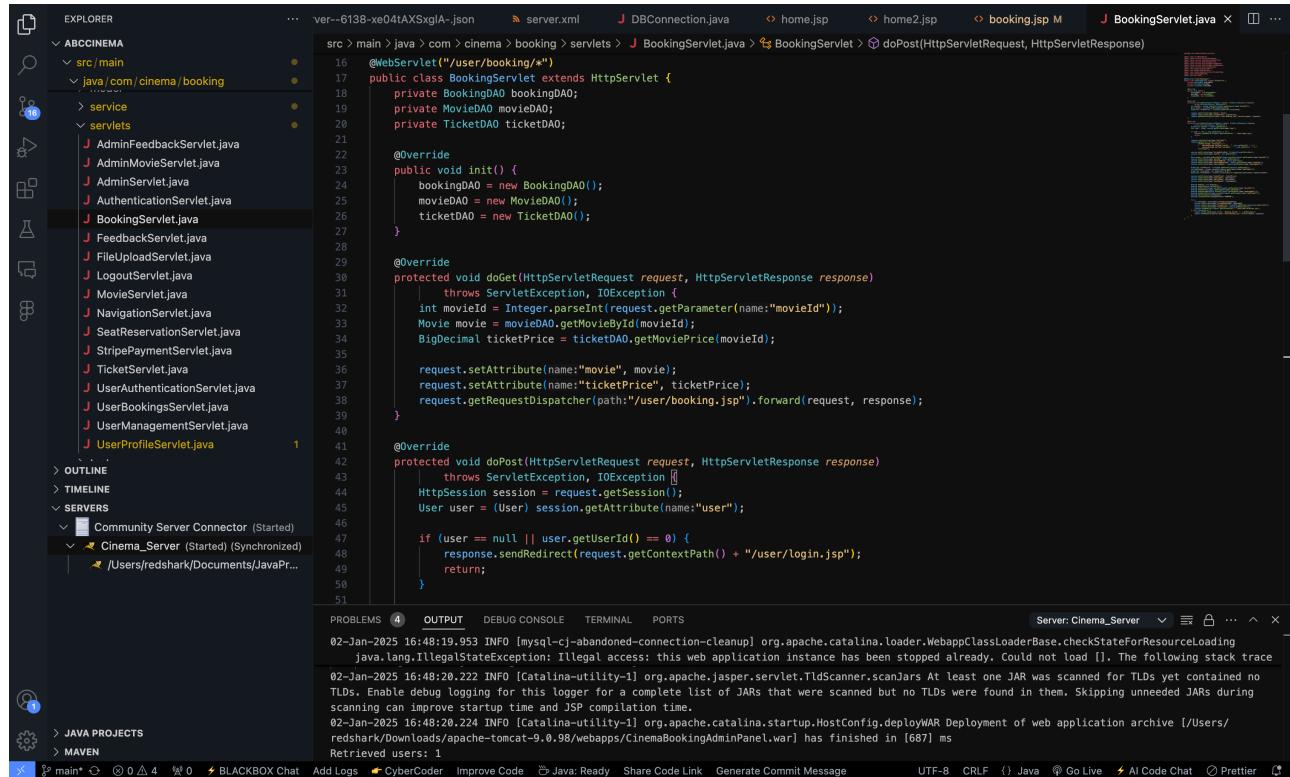
Ticket Price:

Total Amount:

**Continue to Seat Selection →**

This page allows users to book their ticket for the selected movie. In this page user can choose the show date, show time and number of tickets. This page shows the price of the tickets according to the user's input. Then clicking the Continue to Seat Selection button leads user to the seat's selection page.

## Booking Servlet



The screenshot shows the Eclipse IDE interface with the following details:

- EXPLORER View:** Shows the project structure under "ABCINEMA". The "src/main/java/com/cinema/booking" package contains several Java files: AdminFeedbackServlet.java, AdminMovieServlet.java, AdminServlet.java, AuthenticationServlet.java, BookingServlet.java (selected), FeedbackServlet.java, FileUploadServlet.java, LogoutServlet.java, MovieServlet.java, NavigationServlet.java, SeatReservationServlet.java, StripePaymentServlet.java, TicketServlet.java, UserAuthenticationServlet.java, UserBookingsServlet.java, UserManagementServlet.java, and UserProfileServlet.java.
- Servers View:** Shows a "Community Server Connector (Started)" and a "Cinema\_Server (Started) (Synchronized)" server instance.
- Java Projects View:** Shows the "JAVA PROJECTS" section with "MAVEN" selected.
- Code Editor:** Displays the "BookingServlet.java" code. The code is annotated with Javadoc-style comments and imports DAO classes for Booking, Movie, and Ticket.
- Terminal:** Shows the server logs for the Cinema\_Server instance, indicating successful deployment and startup.
- Bottom Bar:** Includes various Eclipse and Java-related icons and text like "BLACKBOX Chat", "Add Logs", "CyberCoder", "Improve Code", "Java: Ready", "Share Code Link", "Generate Commit Message", "UTF-8", "CRLF", "AI Code Chat", "Prettier", and "Retrieved users: 1".

The **BookingServlet** is a core component of the 'ABC Cinema' ticket booking system that handles user requests related to booking movies. It facilitates the process of selecting movies, choosing showtimes, specifying the number of tickets, and initiating the booking process.

# Booking SQL Queries

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Administration' is selected. Below it, the 'Schemas' tab is active, showing the 'cinema\_db' schema. Under 'Tables', the 'bookings' table is selected. The main pane displays the SQL code for creating the 'bookings' table:

```
1 * CREATE TABLE `bookings` (
2     `booking_id` int NOT NULL AUTO_INCREMENT,
3     `user_id` int NOT NULL,
4     `movie_id` int NOT NULL,
5     `show_time` varchar(50) NOT NULL,
6     `booking_date` date NOT NULL,
7     `num_tickets` int NOT NULL,
8     `total_amount` decimal(10,2) NOT NULL,
9     `payment_status` enum('PENDING','COMPLETED','FAILED') DEFAULT 'PENDING',
10    `payment_intent_id` varchar(255) DEFAULT NULL,
11    `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
12    PRIMARY KEY (`booking_id`),
13    UNIQUE KEY `payment_intent_id` (`payment_intent_id`),
14    KEY `user_id` (`user_id`),
15    KEY `movie_id` (`movie_id`),
16    CONSTRAINT `bookings_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`),
17    CONSTRAINT `bookings_ibfk_2` FOREIGN KEY (`movie_id`) REFERENCES `movies` (`movie_id`)
18 ) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
19 * SELECT * FROM cinema_db.bookings;
```

The result grid shows three rows of booking data:

| booking_id | user_id | movie_id | show_time        | booking_date | num_tickets | total_amount | payment_status | payment_intent_id           | created_at          |
|------------|---------|----------|------------------|--------------|-------------|--------------|----------------|-----------------------------|---------------------|
| 1          | 1       | 1        | 8:30 AM-10:30 AM | 2025-01-04   | 2           | 2000.00      | COMPLETED      | pi_3QcgeNAsGXnTq4j51Dap18Jl | 2025-01-02 10:27:25 |
| 2          | 1       | 1        | 4:30 PM-6:30 PM  | 2025-01-17   | 1           | 1000.00      | PENDING        | pi_3QclzoAeGXnTq4j51AdhkRd  | 2025-01-02 12:47:53 |
| 3          | 1       | 2        | 12:30 PM-2:30 PM | 2025-01-17   | 1           | 1000.00      | COMPLETED      | pi_3QclzoAeGXnTq4j51AdhkRd  | 2025-01-02 10:17:32 |

The Action Output panel shows two recent queries:

| Time       | Action   | Response           | Duration / Fetch Time  |
|------------|--|--------------------|------------------------|
| 4 17:14:19 | SELECT * FROM cinema_db.movies LIMIT 0, 1000   | 12 row(s) returned | 0.00064 sec / 0.000... |
| 5 17:21:20 | SELECT * FROM cinema_db.bookings LIMIT 0, 1000 | 3 row(s) returned  | 0.0011 sec / 0.0000... |

The above SQL query show the database table that holds booking details of the users.

## Seat Selection Page

The screenshot shows a seat selection interface. At the top, it says "Select Your Seats". Below that, it shows movie details: "Movie: Dune: Part Two 2024", "Show Time: 12:30 PM-2:30 PM", and "Date: 2025-01-17".

The main area is a grid of 8 rows and 8 columns of seats, labeled A1 through H8. Seat F4 is highlighted with a green background, indicating it is selected. Below the grid, there is a legend: a grey square for "Available", a blue square for "Selected", and a red square for "Occupied".

A "Booking Summary" box at the bottom contains the following information:

- Selected Seats: None
- Total Amount: 0.00 LKR

At the bottom right of the summary box is a "Proceed to Payment" button.

This page shows the users how the screen and the seats are in the cinema hall. And by clicking on the seat numbers the user can select the seat for only the number of seats that user have entered before. If a previously logged user has booked seats before the current user, those seats represent in red color and the current user cannot book those seats. This avoids the seats collision. After selecting the seats, the user can provide the payment by clicking the Proceed to Payment button.

## SeatReservation Servlet

```

server.xml          J DBConnection.java      home.jsp      home2.jsp      booking.jsp M      J BookingServlet.java      J SeatReservationServlet.java X      ...
src > main > java > com > cinema > booking > servlets > J SeatReservationServlet.java > {} com.cinema.booking.servlets

19  @WebServlet(urlPatterns = {
20      "/user/seats/reserved",
21      "/user/seats/store-session"
22  })
23  public class SeatReservationServlet extends HttpServlet {
24      private ReservedSeatDAO reservedSeatDAO;
25      private Gson gson;
26
27      @Override
28      public void init() {
29          reservedSeatDAO = new ReservedSeatDAO();
30          gson = new Gson();
31      }
32
33      @Override
34      protected void doGet(HttpServletRequest request, HttpServletResponse response)
35          throws ServletException, IOException {
36          int movieId = Integer.parseInt(request.getParameter("movieId"));
37          String showTime = request.getParameter("showTime");
38          java.sql.Date bookingDate = java.sql.Date.valueOf(request.getParameter("bookingDate"));
39
40          try {
41              List<String> reservedSeats = reservedSeatDAO.getReservedSeats(movieId, showTime, bookingDate);
42              response.setContentType(type:"application/json");
43              response.getWriter().write(gson.toJson(reservedSeats));
44          } catch (Exception e) {
45              response.setStatus(HttpServletRequest.SC_INTERNAL_SERVER_ERROR);
46              response.getWriter().write(gson.toJson("Error fetching reserved seats"));
47          }
48      }
49
50      @Override
51      protected void doPost(HttpServletRequest request, HttpServletResponse response)
52          throws ServletException, IOException {
53          String pathInfo = request.getServletPath();

```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS Server: Cinema\_Server

```

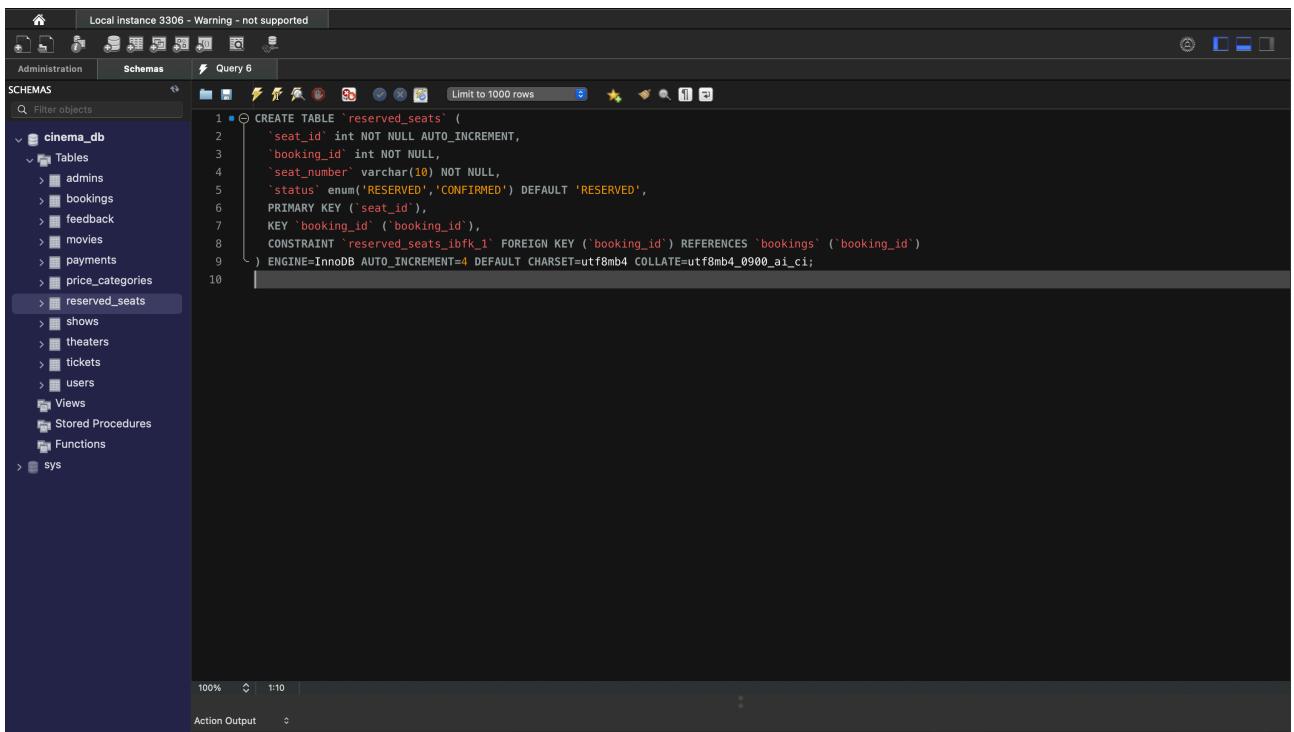
02-Jan-2025 16:48:19.953 INFO [mysql-cj-abandoned-connection-clean-up] org.apache.catalina.loader.WebappClassLoaderBase.checkStateForResourceLoading
java.lang.IllegalStateException: Illegal access: this web application instance has been stopped already. Could not load []. The following stack trace
02-Jan-2025 16:48:20.222 INFO [Catalina-utility-1] org.apache.jasper.servlet.TldScanner.scanJars At least one JAR was scanned for TLDs yet contained no
TLDs. Enable debug logging for this logger for a complete list of JARs that were scanned but no TLDs were found in them. Skipping unneeded JARs during
scanning can improve startup time and JSP compilation time.
02-Jan-2025 16:48:20.224 INFO [Catalina-utility-1] org.apache.catalina.startup.HostConfig.deployWAR Deployment of web application archive [/Users/
redshark/Downloads/apache-tomcat-9.0.98/webapps/CinemaBookingAdminPanel.war] has finished in [687] ms
Retrieved users: 1

```

Logs CyberCoder Improve Code Java: Ready Share Code Link Generate Commit Message UTF-8 CRLF {} Java Go Live AI Code Chat Prettier

The **SeatReservationServlet** is responsible for managing seat reservations in the cinema booking system. It provides functionalities to fetch reserved seats, reservation for new seats, and store seat reservation related data in the user session.

## Reserved Seats Table Query



The screenshot shows the MySQL Workbench interface with the 'cinema\_db' schema selected. In the central query editor, the following SQL code is displayed:

```
1 • CREATE TABLE `reserved_seats` (
2     `seat_id` int NOT NULL AUTO_INCREMENT,
3     `booking_id` int NOT NULL,
4     `seat_number` varchar(10) NOT NULL,
5     `status` enum('RESERVED','CONFIRMED') DEFAULT 'RESERVED',
6     PRIMARY KEY (`seat_id`),
7     KEY `booking_id` (`booking_id`),
8     CONSTRAINT `reserved_seats_ibfk_1` FOREIGN KEY (`booking_id`) REFERENCES `bookings` (`booking_id`)
9 ) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

This table holds the data about reserved seats and when user requests a seat reservation servlet checks if the availability from this table.

## Payment Page

The payment sandbox that we have used is **Stripe** and it provides test keys for testing purposes. In above page users can provide their credit/debit card details and complete the payment procedure.

## Complete Your Payment

⌚ Time remaining: 9:45

### 🎥 Order Details

Movie:  
Show Time:  
Date:

Dune: Part Two 2024  
12:30 PM-2:30 PM  
2025-01-17

### 💰 Price Details

Ticket Price:  
Number of Tickets:  
Service Fee:

1000.00 LKR  
1  
0.00 LKR

Total:

1000.00 LKR

### 💳 Payment Method

Credit/Debit Card

 Card number

Autofill [link](#)

Save card for future payments

 Pay Now

X Cancel Payment

## StripePayment Servlet

For the ABC Cinema web application, StripePaymentServlet handles Stripe payment processes. It processes both GET and POST requests, manages payment activities through various endpoints, and initialises required services. Creating payment intents, managing Stripe webhooks, logging unsuccessful payments, processing successful payments, and starting refunds are some of the essential features. It refreshes the database with booking and payment statuses and stores temporary information using session data. The servlet handles mistakes gently and guarantees safe and effective payment integration.

The screenshot shows a Java IDE interface with the following details:

- Project Structure:** The project is named "ABCCinema". The "StripePaymentServlet.java" file is open in the editor.
- Code Content:** The code is a Java servlet for handling Stripe payments. It includes imports for javax.servlet, javax.servlet.http, and javax.servlet.annotation. It defines a class `StripePaymentServlet` that extends `HttpServlet`. The class has private fields for `StripeService`, `PaymentService`, `BookingDAO`, and `StripeWebhookHandler`. The `init()` method initializes these services. The `doGet()` and `doPost()` methods handle requests for creating payment intents and saving payments respectively, using the initialized services.
- Terminal Output:** The terminal tab shows deployment logs for the "Cinema\_Server" server. It includes messages from Catalina, MySQL, and Tomcat, indicating the deployment of the "CinemaBookingAdminPanel.war" application and the retrieval of users.

## Payments Table Query

The screenshot shows the MySQL Workbench interface. On the left, the 'Schemas' tree view shows the 'cinema\_db' schema selected, with the 'payments' table highlighted. The main pane displays the SQL code for creating the 'payments' table:

```
CREATE TABLE `payments` (
  `payment_id` int NOT NULL AUTO_INCREMENT,
  `booking_id` int NOT NULL,
  `user_id` int NOT NULL,
  `amount` decimal(10,2) NOT NULL,
  `payment_intent_id` varchar(255) DEFAULT NULL,
  `status` enum('PENDING','COMPLETED','FAILED') DEFAULT 'PENDING',
  `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`payment_id`),
  UNIQUE KEY `payment_intent_id` (`payment_intent_id`),
  KEY `booking_id` (`booking_id`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `payments_ibfk_1` FOREIGN KEY (`booking_id`) REFERENCES `bookings` (`booking_id`),
  CONSTRAINT `payments_ibfk_2` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
SELECT * FROM cinema_db.payments;
```

Below the code, the 'Result Grid' shows the data in the 'payments' table:

| payment_id | booking_id | user_id | amount  | payment_intent_id          | status    | created_at          |
|------------|------------|---------|---------|----------------------------|-----------|---------------------|
| 1          | 1          | 1       | 2000.00 | pi_3QcgeNAeGXTq4j51Dap18Jl | COMPLETED | 2025-01-02 10:38:52 |
| 2          | 3          | 1       | 1000.00 | pi_3QclzqAeGXTq4j51Adhk0Rd | COMPLETED | 2025-01-02 16:21:23 |
| NULL       | NULL       | NULL    | NULL    | NULL                       | NULL      | NULL                |

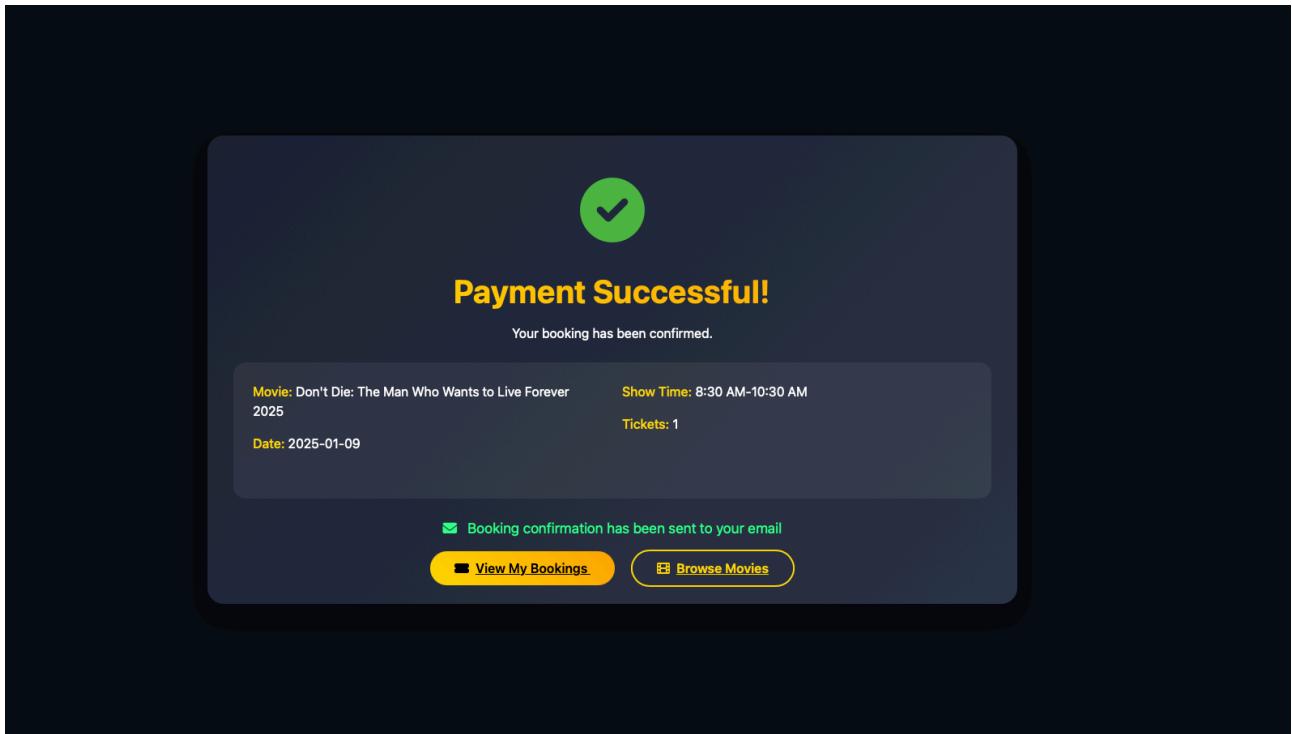
The 'Action Output' section at the bottom shows the history of queries:

| Action | Time     | Response                                       | Duration / Fetch Time                        |
|--------|----------|--|--|
| 5      | 17:21:20 | SELECT * FROM cinema_db.bookings LIMIT 0, 1000 | 3 row(s) returned<br>0.0011 sec / 0.00001... |
| 6      | 21:11:41 | SELECT * FROM cinema_db.payments LIMIT 0, 1000 | 2 row(s) returned<br>0.0013 sec / 0.00000... |

The above ‘Payments’ table holds the data that stored from the user payment sessions.

## Payment Success Page

The above page popups when the payment is completed and the Ticket Reservation receipt will be send to the user’s registered email address and by clicking on the ‘View my bookings’ users can directly view their bookings from Web-Application. And clicking on the ‘Browse Movies’ users will redirected to the movie page.



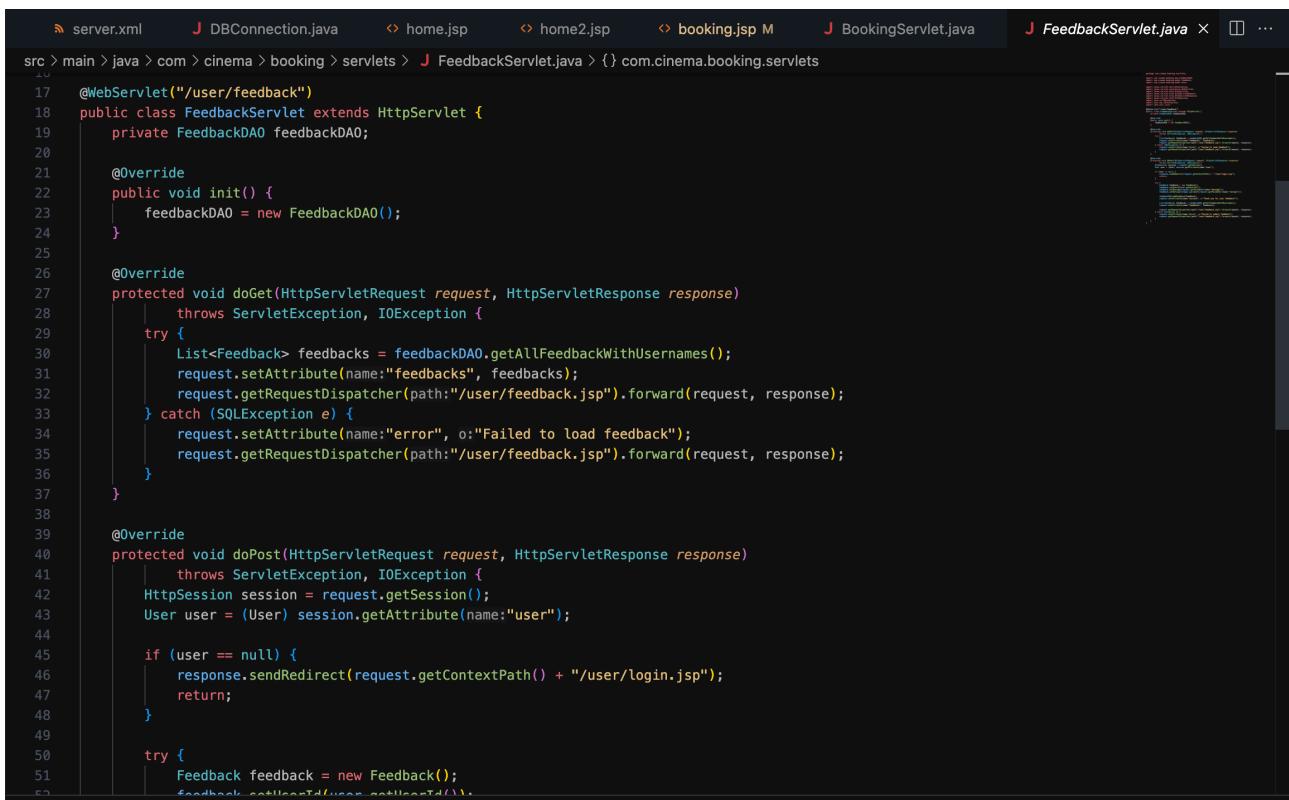
## Feedbacks Page

A screenshot of the ABC Cinema website's feedback section. At the top left is the logo "ABC CINEMA". The top right features a navigation bar with links for "Home", "Movies", "My Bookings", "Feedback" (which is highlighted in red), and a user profile for "Sehara". Below the navigation is a large dark card with the heading "Share Your Experience". It includes a "Rate Your Experience" section with five star icons, a "Your Feedback" section with a text input placeholder "Tell us about your experience...", and a blue "Submit Feedback" button. Below this card is a section titled "What Our Users Say" featuring a user review from "Sehara" dated "Jan 02, 2025 12:21". The review text is "Fantastic Experience !". To the right of the review is a row of five red star icons.

Users can click on 'Feedbacks' button in navigation bar and it will redirects user to the above page. First users have to click on star icons to rate the 'ABC Cinema' and they can post about their thoughts by writing them below and clicking the 'Submit Feedback' button. The submitted feedbacks are shown below the feedback card.

## Feedback Servlet

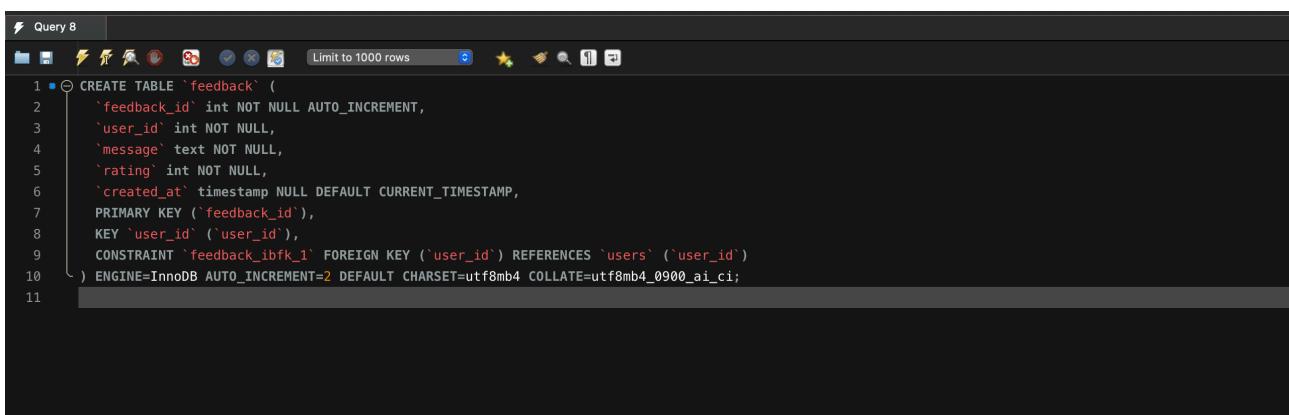
The below feedback servlet is used to display all the feedbacks in a user-friendly style using comments and associated usernames. And also the ‘Feedback Servlet’ captures users’ inputs and saved them to the database. This will only allows logged in users to submit a feedback.



A screenshot of a Java IDE showing the code for the FeedbackServlet.java file. The code is annotated with line numbers from 10 to 57. It includes imports for javax.servlet, javax.servlet.http, javax.servlet.annotation, com.cinema.booking.model, com.cinema.booking.dao, and com.cinema.booking.util. The class extends HttpServlet and implements Serializable. It has methods for doGet and doPost. The doGet method retrieves all feedbacks with associated usernames and forwards them to a JSP page. The doPost method checks if a user is logged in; if not, it redirects to the login page. If a user is logged in, it creates a new Feedback object and saves it to the database using the FeedbackDAO.

```
10 import javax.servlet.*;
11 import javax.servlet.http.*;
12 import javax.servlet.annotation.*;
13 import com.cinema.booking.model.*;
14 import com.cinema.booking.dao.*;
15 import com.cinema.booking.util.*;
16 
17 @WebServlet("/user/feedback")
18 public class FeedbackServlet extends HttpServlet {
19     private FeedbackDAO feedbackDAO;
20 
21     @Override
22     public void init() {
23         feedbackDAO = new FeedbackDAO();
24     }
25 
26     @Override
27     protected void doGet(HttpServletRequest request, HttpServletResponse response)
28             throws ServletException, IOException {
29         try {
30             List<Feedback> feedbacks = feedbackDAO.getAllFeedbackWithUsernames();
31             request.setAttribute(name:"feedbacks", feedbacks);
32             request.getRequestDispatcher(path:"/user/feedback.jsp").forward(request, response);
33         } catch (SQLException e) {
34             request.setAttribute(name:"error", o:"Failed to load feedback");
35             request.getRequestDispatcher(path:"/user/feedback.jsp").forward(request, response);
36         }
37     }
38 
39     @Override
40     protected void doPost(HttpServletRequest request, HttpServletResponse response)
41             throws ServletException, IOException {
42         HttpSession session = request.getSession();
43         User user = (User) session.getAttribute(name:"user");
44 
45         if (user == null) {
46             response.sendRedirect(request.getContextPath() + "/user/login.jsp");
47             return;
48         }
49 
50         try {
51             Feedback feedback = new Feedback();
52             feedback.setComment(request.getParameter("comment"));
53             feedback.setUser_id((int) session.getAttribute("user_id"));
54             feedback.setRating(Integer.parseInt(request.getParameter("rating")));
55             feedback.setCreated_at(new Date());
56             feedbackDAO.insertFeedback(feedback);
57         } catch (SQLException e) {
58             request.setAttribute(name:"error", o:"Failed to insert feedback");
59             request.getRequestDispatcher(path:"/user/feedback.jsp").forward(request, response);
60         }
61     }
62 }
```

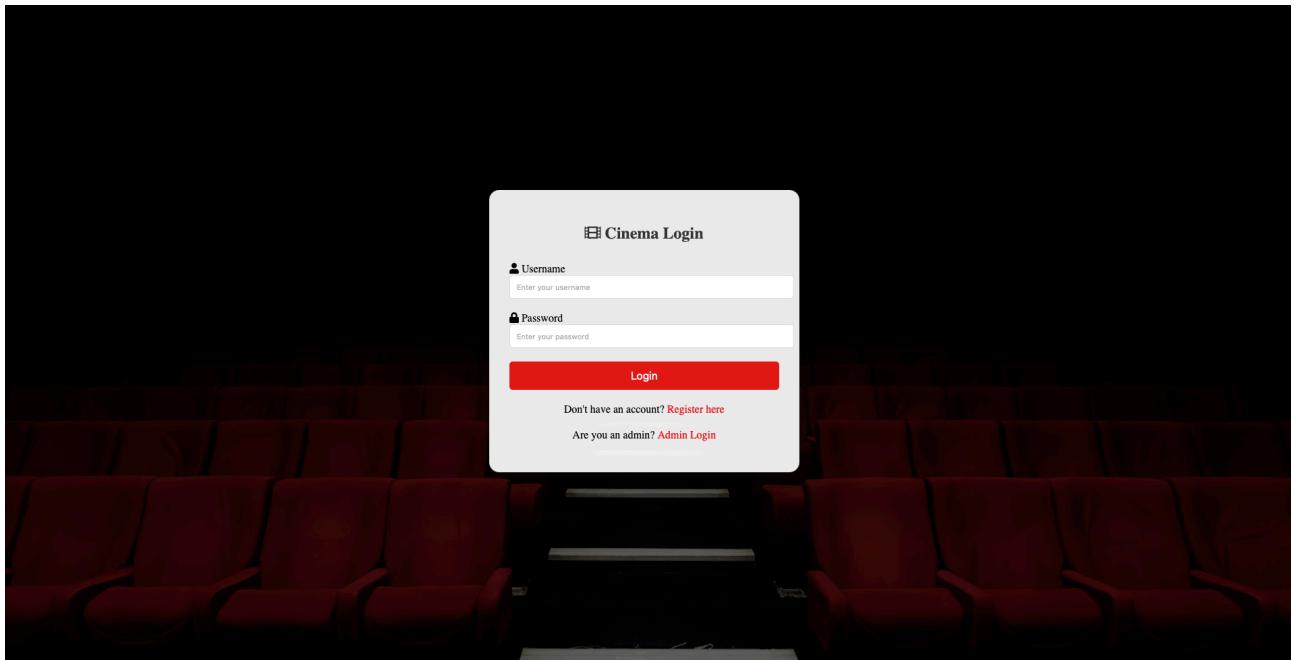
## Feedbacks Table Query



A screenshot of MySQL Workbench showing the SQL query for creating the feedback table. The query is annotated with line numbers from 1 to 11. It creates a table named 'feedback' with columns for feedback\_id (primary key, auto-increment), user\_id (not null), message (text, not null), rating (int, not null), and created\_at (timestamp, default current\_timestamp). It includes a foreign key constraint 'feedback\_ibfk\_1' linking user\_id to the users table. The table uses InnoDB engine with utf8mb4 charset and collation.

```
1 CREATE TABLE `feedback` (
2     `feedback_id` int NOT NULL AUTO_INCREMENT,
3     `user_id` int NOT NULL,
4     `message` text NOT NULL,
5     `rating` int NOT NULL,
6     `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
7     PRIMARY KEY (`feedback_id`),
8     KEY `user_id` (`user_id`),
9     CONSTRAINT `feedback_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`)
10 ) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
11 
```

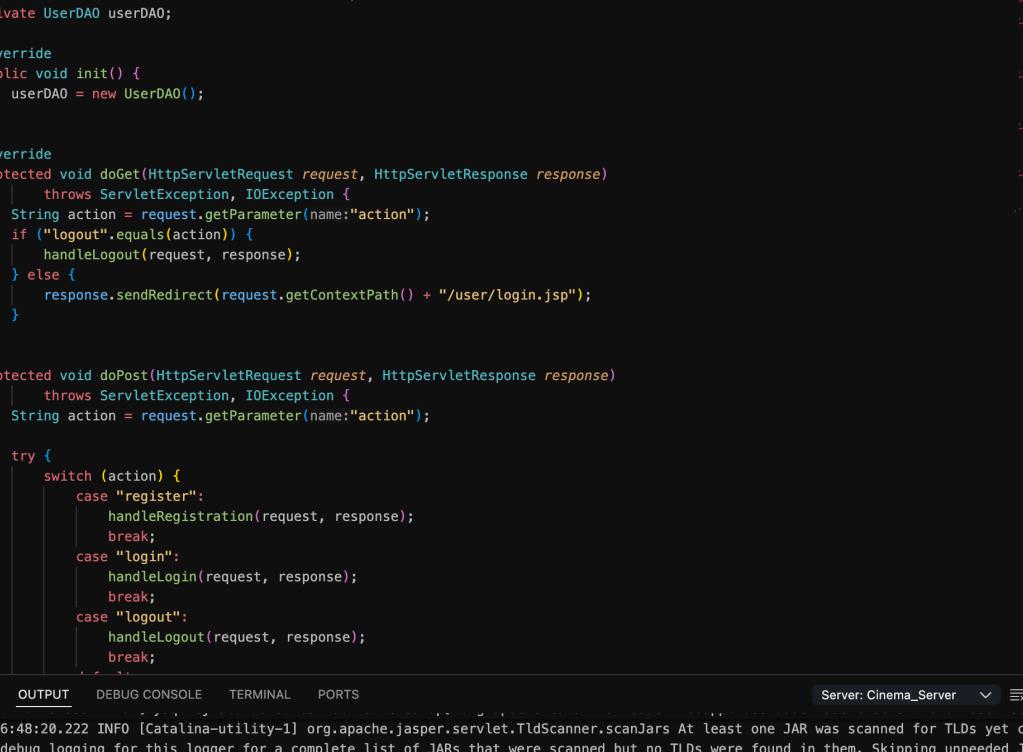
## Login Page



The above page is the login page of 'ABC Cinema' and users have to enter their registered username and password to login to the Web-Application and if a new user comes they have to click on 'Register here' link and it will redirect to the register page and after the registration they will redirect back to the login page. Then they can login using their credentials.

If an admin comes to the login page they have separate login interface for admin panel. Admin panel login page explanation can be seen below after the user login interface explanation.

## User Login handling Servlet



The screenshot shows a Java IDE interface with the following details:

- File Structure:** The left sidebar shows the project structure: server.xml, DBConnection.java, home.jsp, home2.jsp, booking.jsp, BookingServlet.java, and UserAuthenticationServlet.java.
- UserAuthenticationServlet.java Content:** The main code editor displays the implementation of the `UserAuthenticationServlet`. It includes methods for handling GET and POST requests, managing user authentication (register, login, logout), and retrieving users from a DAO.
- Output Tab:** The bottom-left tab bar shows the "OUTPUT" tab is selected. The output window displays logs from the Catalina server and terminal logs.
- Server Status:** The bottom right shows the server status as "Cinema\_Server".

```
server.xml J DBConnection.java < home.jsp < home2.jsp < booking.jsp M J BookingServlet.java J UserAuthenticationServlet.java X
```

```
src > main > java > com > cinema > booking > servlets > J UserAuthenticationServlet.java > {} com.cinema.booking.servlets
12
13     @WebServlet("/user/auth")
14     public class UserAuthenticationServlet extends HttpServlet {
15         private UserDao userDao;
16
17         @Override
18         public void init() {
19             userDao = new UserDao();
20         }
21
22         @Override
23         protected void doGet(HttpServletRequest request, HttpServletResponse response)
24             throws ServletException, IOException {
25             String action = request.getParameter(name:"action");
26             if ("logout".equals(action)) {
27                 handleLogout(request, response);
28             } else {
29                 response.sendRedirect(request.getContextPath() + "/user/login.jsp");
30             }
31         }
32
33         protected void doPost(HttpServletRequest request, HttpServletResponse response)
34             throws ServletException, IOException {
35             String action = request.getParameter(name:"action");
36
37             try {
38                 switch (action) {
39                     case "register":
40                         handleRegistration(request, response);
41                         break;
42                     case "login":
43                         handleLogin(request, response);
44                         break;
45                     case "logout":
46                         handleLogout(request, response);
47                         break;
48                 }
49             } catch (Exception e) {
50                 e.printStackTrace();
51             }
52         }
53     }
54 }
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS Server: Cinema\_Server

```
02-Jan-2025 16:48:20.222 INFO [Catalina-utility-1] org.apache.jasper.servlet.TldScanner.scanJars At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for this logger for a complete list of JARs that were scanned but no TLDs were found in them. Skipping unneeded JARs during scanning can improve startup time and JSP compilation time.
02-Jan-2025 16:48:20.224 INFO [Catalina-utility-1] org.apache.catalina.startup.HostConfig.deployWAR Deployment of web application archive [/Users/redshark/Downloads/apache-tomcat-9.0.98/webapps/CinemaBookingAdminPanel.war] has finished in [687] ms
Retrieved users: 1
Session userId: 1
```

For the web application, the UserAuthenticationServlet manages user authentication tasks including registration, login, and logout. The action parameter determines the steps it takes while processing HTTP GET and POST requests. Depending on the activity, the doPost method leads to the registration, login, or logout methods. The UserDAO class is used to gather and store user data for registration. If credentials are good, the servlet creates a session after authenticating the user for login. The response contains headers to avoid caching and a JSON redirect for a seamless transition in the event of a logout, which invalidates the session. Setting error messages and sending requests to the relevant JSP pages are how errors are handled.

## Users Table Query

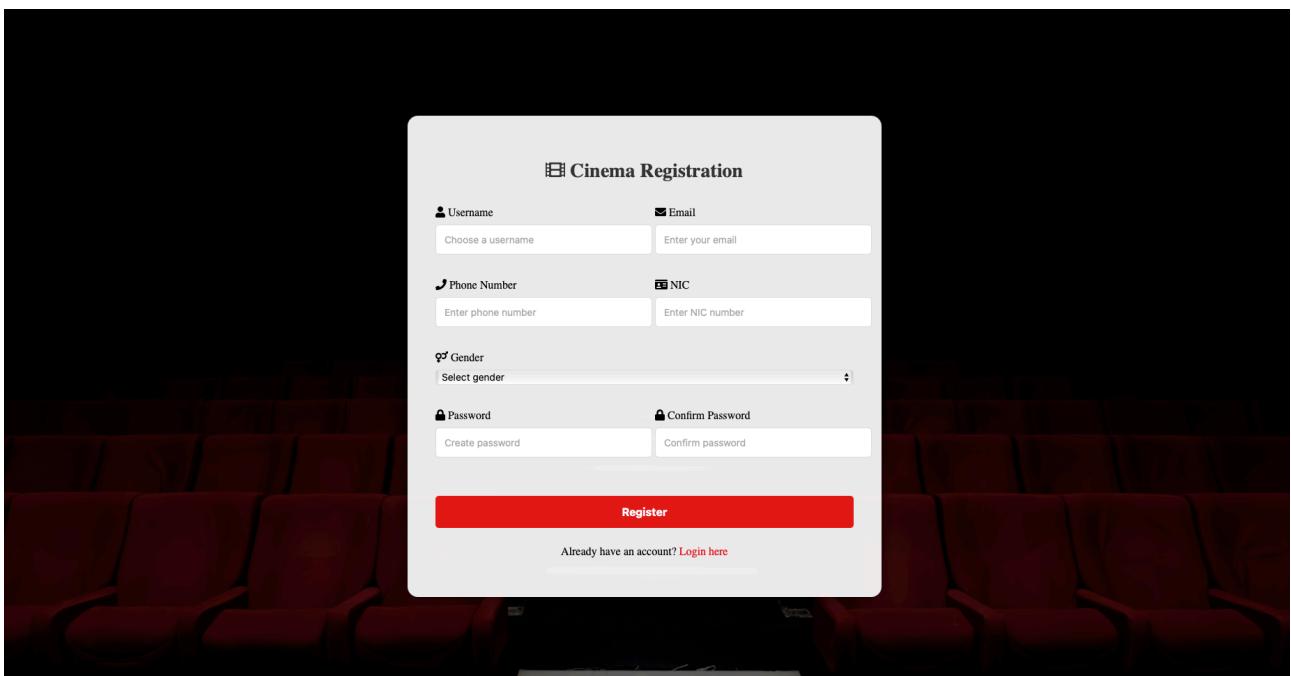
The below ‘users’ table stores all users details and the users’ passwords are hashed because of security reasons.

```

1 CREATE TABLE `users` (
2     `user_id` int NOT NULL AUTO_INCREMENT,
3     `username` varchar(50) NOT NULL,
4     `password` varchar(255) NOT NULL,
5     `email` varchar(100) NOT NULL,
6     `role` enum('ADMIN','USER') DEFAULT 'USER',
7     `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
8     `phone_number` varchar(15) NOT NULL,
9     `nic` varchar(10) NOT NULL,
10    `gender` enum('MALE','FEMALE','OTHER') NOT NULL,
11    `profile_image_url` varchar(255) DEFAULT NULL,
12    `status` enum('active','inactive') DEFAULT 'active',
13    PRIMARY KEY (`user_id`),
14    UNIQUE KEY `username` (`username`),
15    UNIQUE KEY `email` (`email`),
16    UNIQUE KEY `nic` (`nic`)
17 ) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
18 SELECT * FROM cinema_db.users;

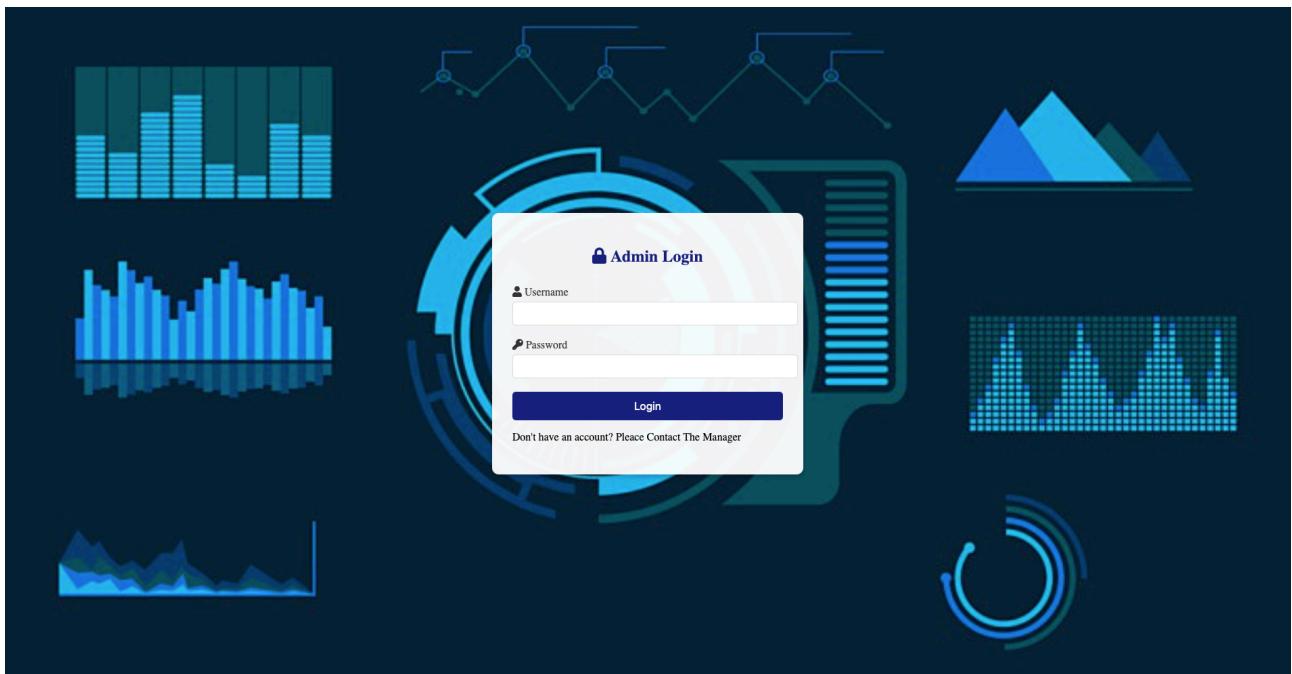
```

## User Register Page



In the above user registration page, new coming users can create an account of ‘ABC Cinema’. At first user have to enter specific user name and their email. Then the users have to enter their local telephone number and National Identity Card number. Then, from the gender dropdown menu they have to select the gender and enter secure password for web application. After that by clicking the Register button, if given the informations are valid, the user will redirect to the login page.

## Admin Login Page



In the above admin login page, admin's have to log in to the system via the given credentials that have given by the company (ABC Cinema). The 'Login' button will redirects to the 'Admin Dashboard' that will explained below.

# Authentication Servlet

The screenshot shows a Java IDE interface with the following details:

- Code Editor:** The main window displays the `AuthenticationServlet.java` file. The code implements a `HttpServlet` for handling admin authentication requests. It includes methods for `init()`, `doPost(HttpServletRequest request, HttpServletResponse response)`, and `handleLogin(HttpServletRequest request, HttpServletResponse response)`. It uses an `AdminDAO` to handle database operations.
- Terminal:** Below the editor is a terminal window showing the build process of a WAR file. The output includes:
  - [INFO] Processing war project
  - [INFO] Copying webapp resources [/Users/redshark/Documents/JavaProjectNSBM/ABCCinema/src/main/webapp]
  - [INFO] Building war: [/Users/redshark/Documents/JavaProjectNSBM/ABCCinema/target/CinemaBookingAdminPanel.war]
  - [INFO] [INFO] BUILD SUCCESS
  - [INFO] [INFO] -----
  - [INFO] [INFO] Total time: 1.045 s
  - [INFO] [INFO] Finished at: 2025-01-02T16:48:09+05:30
  - [INFO] [INFO] -----
- Status Bar:** The bottom of the IDE shows tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active.

User authentication for the admin section of the ABC Cinema web application is managed by the `AuthenticationServlet`. It handles requests for registration, login, and logout. The action (login, register, or logout) is identified in the `doPost` method, which then sends the request to the relevant handler. In order to log in, it uses the `AdminDAO` to authenticate the admin credentials. If it is successful, a session is started. It gathers admin information, registers the admin with `AdminDAO`, and sends to the login page if registration is needed. It sends to the login page and invalidates the session for logout. The servlet forwards error messages to the appropriate JSP pages in order to handle exceptions, such as database failures. The `destroy` method also cleans up resources.

# Admin Dashboard

The screenshot shows the Admin Dashboard for ABC Cinema. On the left, a sidebar menu lists: Dashboard, Movie Management, Ticket Management, User Management, Feedback Management, and Booking Management. Below the sidebar is a 'Logout' button. The main content area features a banner with movie stills, followed by four summary cards: Total Tickets (1,234), Active Users (567), Movies (89), and Revenue (\$12,345). A 'Recent Bookings' section shows a booking for John Doe. Another section displays 'Theater Status' for Theater 1, which is at 75% capacity, now playing Dune Part II, and the next show is at 7:30 PM. A 'Revenue Analytics' section is partially visible at the bottom.

We have our admin dashboard, that the admin can do following operations:

- Movie Management
- Ticket Management
- User Management
- Feedback Management
- Booking Management

# Movie Management

ABC Cinema

- Dashboard
- Movie Management
- Ticket Management
- User Management
- Feedback Management
- Booking Management

Logout

Total Movies 12

+ Add New Movie

Thunderbolts 2025  
COMING\_SOON  
⌚ 0 mins  
★ 0.0/5.0

Captain America: Brave New World 2025  
COMING\_SOON  
⌚ 0 mins  
★ 0.0/5.0

Den of Thieves 2: Pantera (2025)  
COMING\_SOON  
⌚ 124 mins  
★ 0.0/5.0

Babygirl 2024  
COMING\_SOON  
⌚ 114 mins  
★ 4.9/5.0

Dune: Part Two 2024  
NOW\_SHOWING  
⌚ 166 mins  
★ 4.9/5.0

A NETFLIX DOCUMENTARY  
DON'T DIE

DAWNEE JOHNSON CHRIS EVANS LUCY LIU MIA SIMMONS  
RED ONE  
ONLY IN THEATERS NOVEMBER 15

PIITIM

SURVIVE  
THE WORLD HAS GONE SIDEWAYS

BLACK

In the movie management page, admins can add new movies, edit movies, and delete available movies.

Thunderbolts 2025

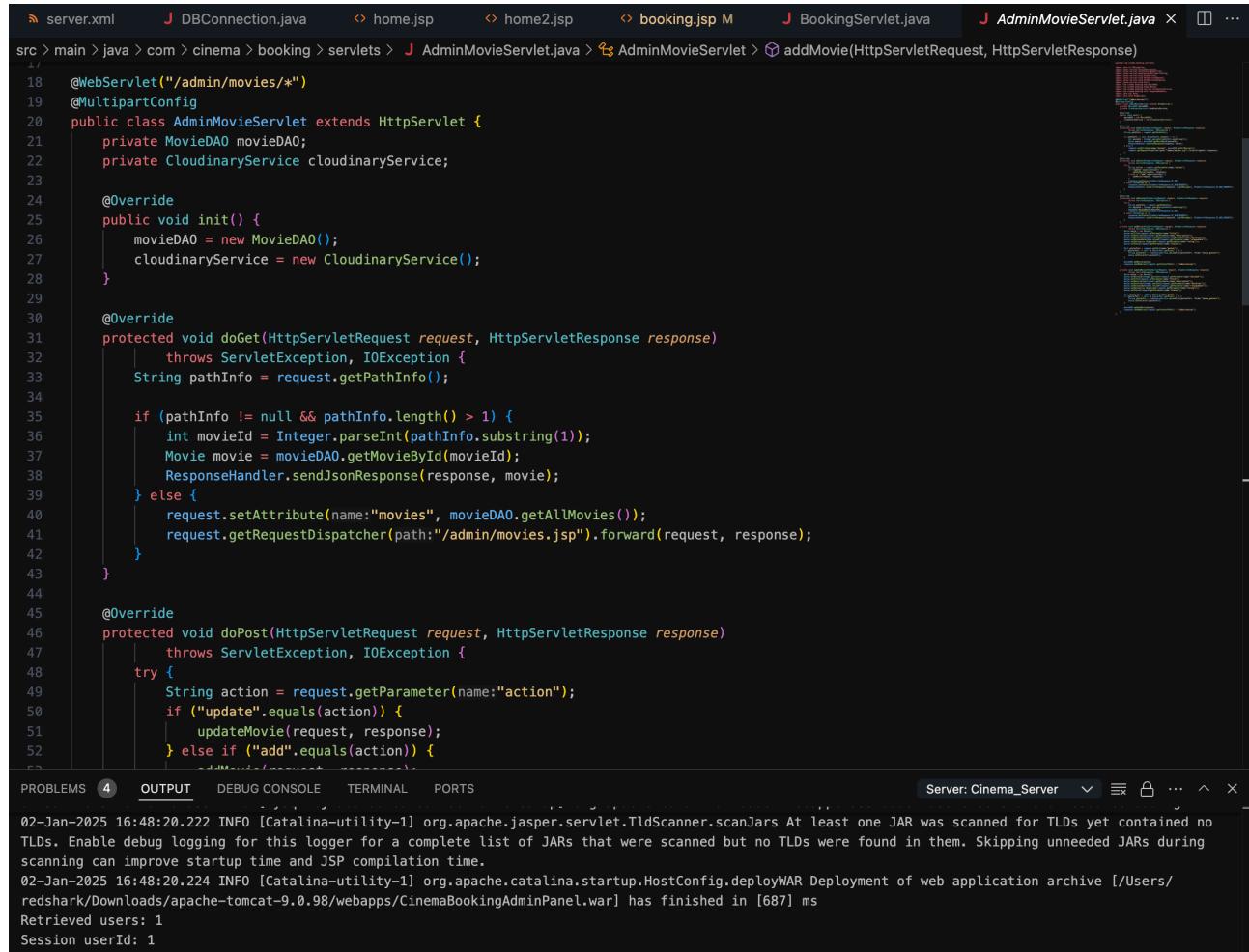
COMING\_SOON

⌚ 0 mins  
★ 0.0/5.0

As this sample movie card, admins can use edit button to view the editing panel and by clicking the delete icon, admins can delete the movies.

In the editing panel, admins can change the title, description, duration, rating, release date, status, and movie poster.

## Admin Movie Servlet



```
server.xml J DBConnection.java < home.jsp < home2.jsp < booking.jsp M J BookingServlet.java J AdminMovieServlet.java X ...  
src > main > java > com > cinema > booking > servlets > J AdminMovieServlet.java > AdminMovieServlet > addMovie(HttpServletRequest, HttpServletResponse)  
1/  
18 @WebServlet("/admin/movies/*")  
19 @MultipartConfig  
20 public class AdminMovieServlet extends HttpServlet {  
21     private MovieDAO movieDAO;  
22     private CloudinaryService cloudinaryService;  
23  
24     @Override  
25     public void init() {  
26         movieDAO = new MovieDAO();  
27         cloudinaryService = new CloudinaryService();  
28     }  
29  
30     @Override  
31     protected void doGet(HttpServletRequest request, HttpServletResponse response)  
32         throws ServletException, IOException {  
33         String pathInfo = request.getPathInfo();  
34  
35         if (pathInfo != null && pathInfo.length() > 1) {  
36             int movieId = Integer.parseInt(pathInfo.substring(1));  
37             Movie movie = movieDAO.getMovieById(movieId);  
38             ResponseHandler.sendJsonResponse(response, movie);  
39         } else {  
40             request.setAttribute(name:"movies", movieDAO.getAllMovies());  
41             request.getRequestDispatcher(path:"/admin/movies.jsp").forward(request, response);  
42         }  
43     }  
44  
45     @Override  
46     protected void doPost(HttpServletRequest request, HttpServletResponse response)  
47         throws ServletException, IOException {  
48         try {  
49             String action = request.getParameter(name:"action");  
50             if ("update".equals(action)) {  
51                 updateMovie(request, response);  
52             } else if ("add".equals(action)) {  
53                 addMovie(request, response);  
54             }  
55         } catch (Exception e) {  
56             e.printStackTrace();  
57         }  
58     }  
59  
59 }  
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS Server: Cinema_Server ... ^ x  
02-Jan-2025 16:48:20.222 INFO [Catalina-utility-1] org.apache.jasper.servlet.TldScanner.scanJars At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for this logger for a complete list of JARs that were scanned but no TLDs were found in them. Skipping unneeded JARs during scanning can improve startup time and JSP compilation time.  
02-Jan-2025 16:48:20.224 INFO [Catalina-utility-1] org.apache.catalina.startup.HostConfig.deployWAR Deployment of web application archive [/Users/redshark/Downloads/apache-tomcat-9.0.98/webapps/CinemaBookingAdminPanel.war] has finished in [687] ms  
Retrieved users: 1  
Session userId: 1
```

Movie management for the web application's admin section is managed by the AdminMovieServlet. It allows to view, add, edit, and remove movie details. The doGet method either lists all movies for display or uses its ID to obtain a single movie. The doPost method handles movie additions or modifications by collecting form data, using the CloudinaryService to handle file uploads (such movie posters), and storing the movie details in the database. Movies can be deleted by their ID using the doDelete method. The servlet utilises the ResponseHandler to transmit responses and communicates with the MovieDAO to carry out database operations. Additionally, it handles file uploads via multipart configuration.

# Ticket Management

The screenshot shows the Admin Ticket Management interface for ABC Cinema. On the left, a dark sidebar menu includes options like Dashboard, Movie Management, Ticket Management (which is selected and highlighted in blue), User Management, Feedback Management, and Booking Management. At the bottom of the sidebar is a Logout link. The main content area features three summary boxes: 'Available Tickets' (450), 'Sold Today' (127), and 'Reserved' (45). Below these are buttons for '+ Add New Tickets', 'Modify Pricing', and 'Remove Tickets'. A table titled 'Current Tickets' lists six movie entries with columns for Ticket ID, Movie, Show Time, Price, Qty, Status, and Actions. Each row includes a green 'AVAILABLE' status indicator and two small blue and red action icons.

The above Admin Ticket Management page allows admins to add new tickets, edit tickets and delete tickets.

A modal dialog box titled 'Add New Ticket' is displayed over a background table of current tickets. The dialog has fields for 'Movie:' (set to 'Thunderbolts 2025 (COMING\_SOON)'), 'Price:' (empty input field), 'Quantity:' (empty input field), and 'Status:' (dropdown set to 'Available'). At the bottom is a large blue 'Add Ticket' button. The background table lists current tickets with columns for Ticket ID, Movie, Show Time, Price, Qty, and Price. The table data is as follows:

| Ticket ID | Movie   | Show Time | Price   | Qty | Price   |
|-----------|---|-----------|---------|-----|---------|
| #1        | Red One 2024                                      |           | 1000.00 | 100 | 1000.00 |
| #2        | Dune: Part Two 2024                               |           | 1000.00 | 200 | 1000.00 |
| #3        | Survive 2024                                      |           | 1000.00 | 150 | 1000.00 |
| #4        | The Lorax 2012 (Christmas special)                |           | 500.00  | 500 | 500.00  |
| #5        | Don't Die: The Man Who Wants to Live Forever 2025 |           | 800.00  | 200 | 800.00  |
| #6        | Avatar: The Way of Water 2022                     |           | 1000.00 | 300 | 1000.00 |

The ‘Add New Tickets’ button will open the above page and admins can add tickets to the available movies. Admins have to select the movie from the dropdown menu and after entering price and quantity of tickets, they can change the status (availability) of selected movie.

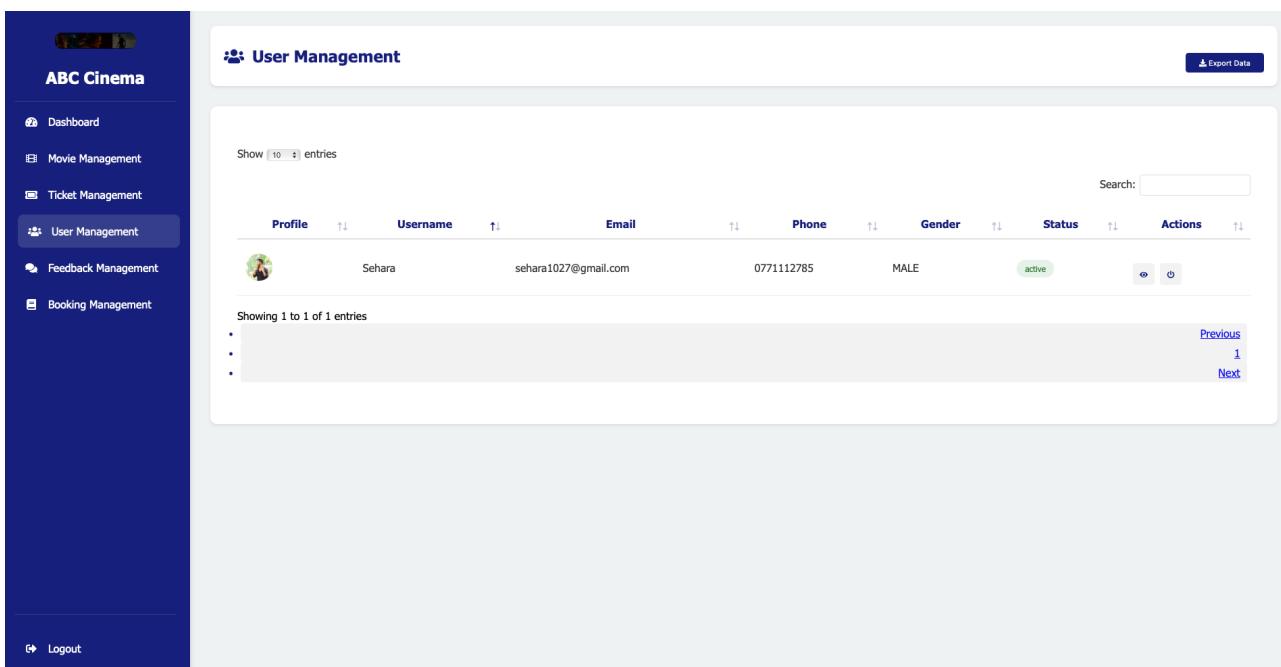
## Current Tickets

| Ticket ID | Movie        | Show Time | Price   | Qty | Status    | Actions   |
|-----------|--------------|-----------|---------|-----|-----------|---|
| #1        | Red One 2024 |           | 1000.00 | 100 | AVAILABLE |   |

By clicking the ‘Edit Icon’ in the actions column admins can edit the added tickets’ details.

By clicking the ‘Delete Icon’ in the actions column admins can delete the added tickets.

## User Management



| Profile   | Username | Email                | Phone      | Gender | Status | Actions   |
|---|----------|----------------------|------------|--------|--------|---|
|  | Sehara   | sehara1027@gmail.com | 0771112785 | MALE   | active |   |

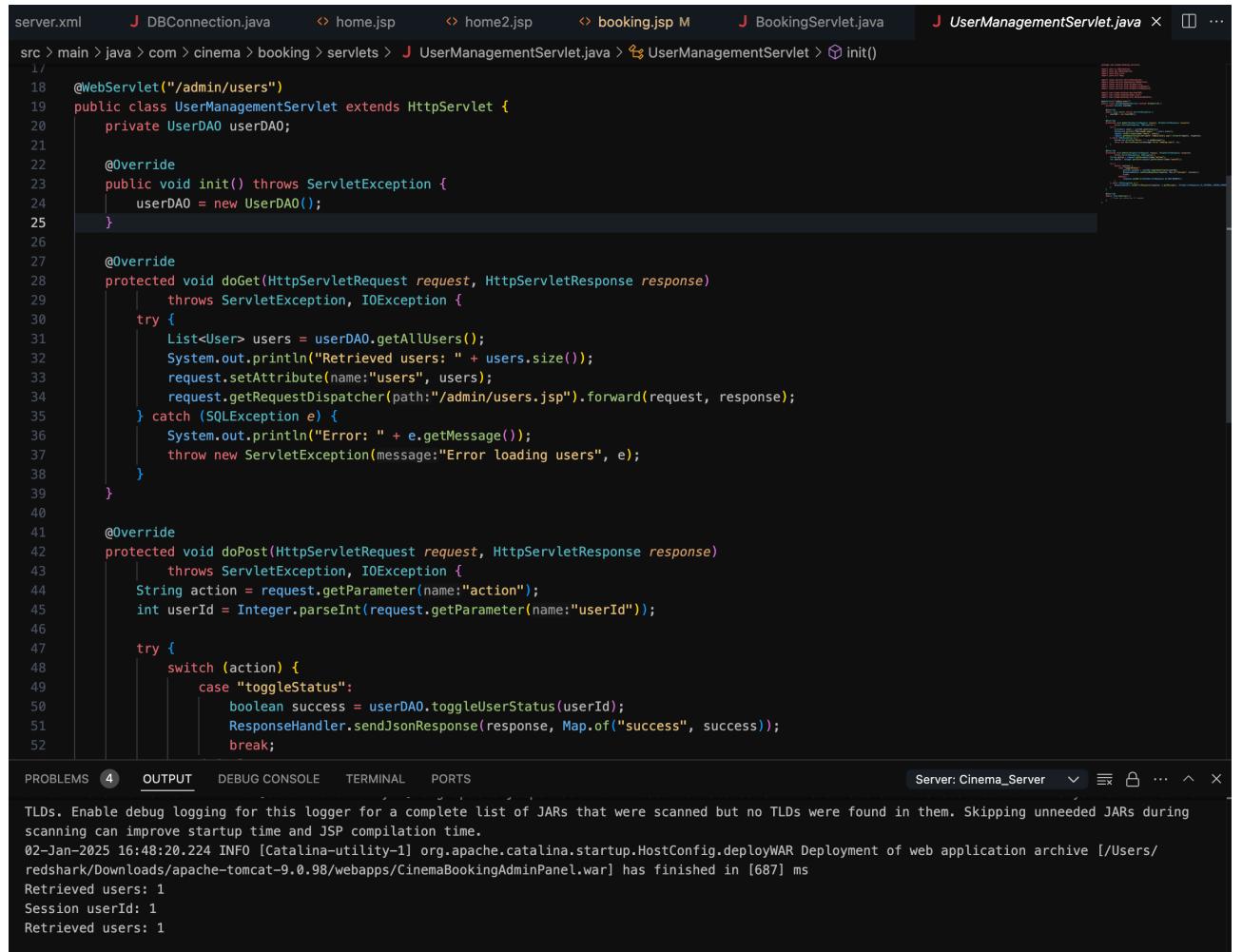
Showing 1 to 1 of 1 entries

Previous [1](#) Next

In the above admin user management page, will allow admins to view all the registered users details except the login password, because of security features. And by clicking the ‘Power button’ in actions column, admins can toggle the status of the users’ accounts.

If the user’s status is active, user can login to their to the ABC Cinema and if admin made the status inactive, users can not login to the web application.

## User Management Servlet



```

server.xml      J DBConnection.java      home.jsp      home2.jsp      booking.jsp M      J BookingServlet.java      J UserManagementServlet.java X      ...
src > main > java > com > cinema > booking > servlets > J UserManagementServlet.java > UserManagementServlet > init()
1/
18 @WebServlet("/admin/users")
19 public class UserManagementServlet extends HttpServlet {
20     private UserDAO userDAO;
21
22     @Override
23     public void init() throws ServletException {
24         userDAO = new UserDAO();
25     }
26
27     @Override
28     protected void doGet(HttpServletRequest request, HttpServletResponse response)
29             throws ServletException, IOException {
30         try {
31             List<User> users = userDAO.getAllUsers();
32             System.out.println("Retrieved users: " + users.size());
33             request.setAttribute(name:"users", users);
34             request.getRequestDispatcher(path:"/admin/users.jsp").forward(request, response);
35         } catch (SQLException e) {
36             System.out.println("Error: " + e.getMessage());
37             throw new ServletException(message:"Error loading users", e);
38         }
39     }
40
41     @Override
42     protected void doPost(HttpServletRequest request, HttpServletResponse response)
43             throws ServletException, IOException {
44         String action = request.getParameter(name:"action");
45         int userId = Integer.parseInt(request.getParameter(name:"userId"));
46
47         try {
48             switch (action) {
49                 case "toggleStatus":
50                     boolean success = userDAO.toggleUserStatus(userId);
51                     ResponseHandler.sendJsonResponse(response, Map.of("success", success));
52                     break;
53             }
54         }
55     }
56 }

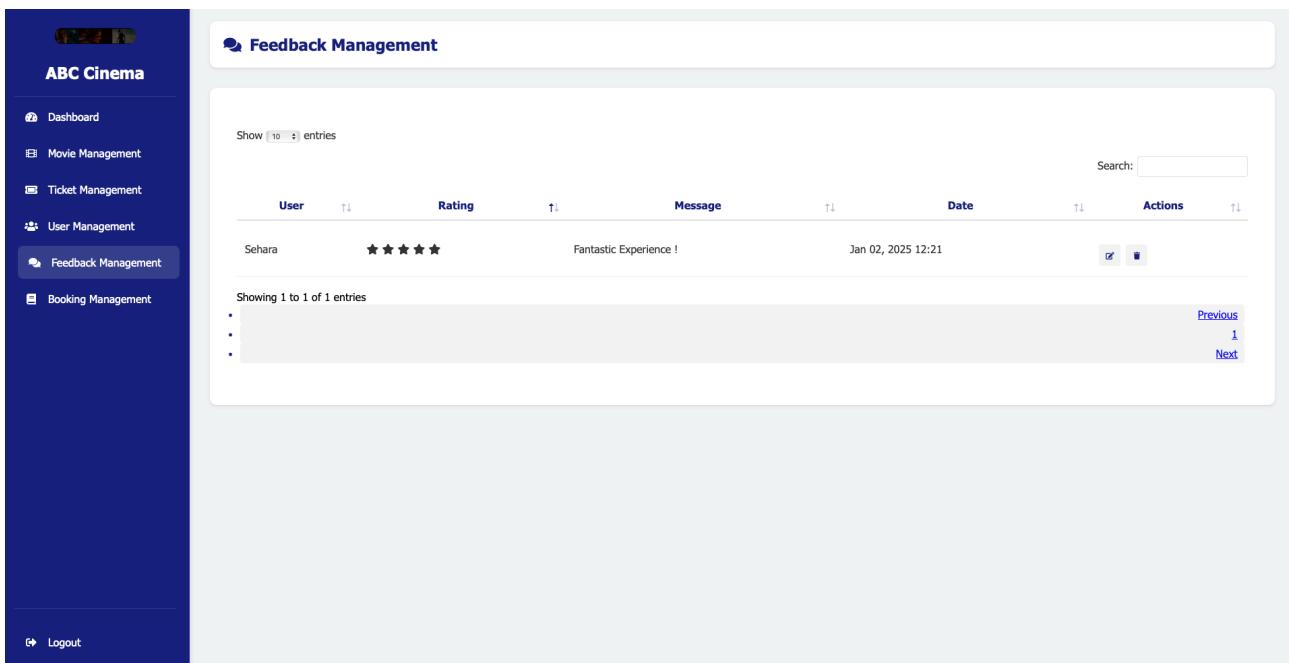
```

PROBLEMS 4    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    Server: Cinema\_Server

TLDs. Enable debug logging for this logger for a complete list of JARs that were scanned but no TLDs were found in them. Skipping unneeded JARs during scanning can improve startup time and JSP compilation time.  
02-Jan-2025 16:48:20.224 INFO [Catalina-utility-1] org.apache.catalina.startup.HostConfig.deployWAR Deployment of web application archive [/Users/redshark/Downloads/apache-tomcat-9.0.98/webapps/CinemaBookingAdminPanel.war] has finished in [687] ms  
Retrieved users: 1  
Session userId: 1  
Retrieved users: 1

In the web application's admin part, the UserManagementServlet is in charge of overseeing user-related operations. The doGet function sends the user data to the users after retrieving a list of every user from the UserDao.jsp page to be shown. It raises a ServletException if there is a problem fetching users. Toggling a user's status is handled by the doPost method, which calls the toggleUserStatus method of the UserDao after receiving an action and a user ID from the request. Success or failure is indicated via a JSON response that is returned. The client receives an error response if there are any problems with the database action.

## Feedback Management



The screenshot shows the 'Feedback Management' section of the ABC Cinema Admin interface. The left sidebar contains navigation links for Dashboard, Movie Management, Ticket Management, User Management, Feedback Management (which is currently selected and highlighted in blue), and Booking Management. The main content area has a header 'Feedback Management' with a back arrow icon. Below it is a table with columns: User, Rating, Message, Date, and Actions. A single row is visible, showing a user named 'Sehara' with a 5-star rating, a message 'Fantastic Experience !', a date 'Jan 02, 2025 12:21', and two icons in the Actions column (Edit and Delete). At the bottom of the table, it says 'Showing 1 to 1 of 1 entries'. Navigation links at the bottom right include 'Previous', a page number '1', and 'Next'.

| User   | Rating | Message                | Date               | Actions |
|--------|--------|------------------------|--------------------|---------|
| Sehara | ★★★★★  | Fantastic Experience ! | Jan 02, 2025 12:21 |         |

In the above feedback management page, admins can view all the submitted feedbacks by users as well as their ratings for managerial decisions. By clicking the 'Edit icon' admins can edit the feedbacks if users posts inappropriate feedbacks. And from the delete icon admins can delete the feedbacks.

# Admin Feedback Servlets

The screenshot shows a Java IDE interface with the following details:

- EXPLORER** view: Shows the project structure under "ABCINEMA". The "src/main/java/com/cinema/booking/services" package contains several servlet files: AdminFeedbackServlet.java, AdminMovieServlet.java, AdminServlet.java, AuthenticationServlet.java, BookingServlet.java, FeedbackServlet.java, FileUploadServlet.java, LogoutServlet.java, MovieServlet.java, NavigationServlet.java, SeatReservationServlet.java, StripePaymentServlet.java, TicketServlet.java, UserAuthenticationServlet.java, UserBookingsServlet.java, UserManagementServlet.java, and UserProfileServlet.java.
- EDITOR** view: Displays the code for AdminFeedbackServlet.java. The code implements the HttpServlet interface and handles GET and POST requests. It uses a FeedbackDAO object to interact with the database. The doGet method retrieves feedback by ID or all feedback entries. The doPost method manages updates and deletions of feedback entries.
- TERMINAL** view: Shows the build logs for the war project. It includes messages like "[INFO] Processing war project", "[INFO] Copying webapp resources", "[INFO] Building war", "[INFO] BUILD SUCCESS", and "[INFO] Total time: 1.045 s".
- STATUS BAR**: Shows the current terminal is zsh, and other status indicators like Java Ready, Share Code Link, and Generate Commit Message.

The AdminFeedbackServlet handles feedback management for the admin section of the web application. Both GET and POST requests are supported. The doGet function can either get every feedback entries and show them on the feedback management page, or it can retrieve specific feedback depending on its ID. Depending on the action indicated in the request, the doPost method manages updating and removing feedback. In the case of "update," a feedback entry is modified, whereas "delete" eliminates a feedback. The servlet utilises a ResponseHandler to provide JSON replies with the results after interacting with the FeedbackDAO to execute database operations.

## **Challenges**

- Had to migrate from glassfish 7.0.15v and netbeans IDE to Visual Studio Code and Tomcat 9.0.98 because of port conflicts of group members.
- Had to clean the http and shutdown ports using nmap after scanning all ports of the system because testings of several servers has filled the most of the common ports.
- Had to fix merge conflicts of the GitHub to maintain the repository.

## **Conclusion**

From this report, We have explained all the functionalities and how they will work in backend of ‘ABC Cinema’ Web-Application

## **References**

Visual Paradigm (2024). *UML Class Diagram Tutorial*. [online] Visual-paradigm.com. Available at: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>.

www.javatpoint.com. (n.d.). *Learn Servlet Tutorial - javatpoint*. [online] Available at: <https://www.javatpoint.com/servlet-tutorial>.

Stripe.com. (2024). *Manage access and API keys*. [online] Available at: <https://docs.stripe.com/sandboxes/dashboard/manage-access> [Accessed 2 Jan. 2025].

Cloudinary.com. (2024). *Cloudinary Java SDK Quick Start / Cloudinary*. [online] Available at: [https://cloudinary.com/documentation/java\\_quickstart](https://cloudinary.com/documentation/java_quickstart) [Accessed 2 Jan. 2025].

tomcat.apache.org. (n.d.). *Apache Tomcat 9 (9.0.75) - Documentation Index*. [online] Available at: <https://tomcat.apache.org/tomcat-9.0-doc/index.html>.

Glassfish.org. (2018). *Eclipse GlassFish Application Deployment Guide, Release 7*. [online] Available at: <https://glassfish.org/docs/latest/application-deployment-guide.html> [Accessed 2 Jan. 2025].

code.visualstudio.com. (n.d.). *Getting Started with Java in Visual Studio Code*. [online] Available at: <https://code.visualstudio.com/docs/java/java-tutorial>.

maven.apache.org. (n.d.). *Maven – Configuring Apache Maven*. [online] Available at: <https://maven.apache.org/configure.html>.

Simplilearn (2020). *Java Servlets Tutorial / Java JSP Tutorial / Java Server-Side Programming For Beginners /Simplilearn*. [online] YouTube. Available at: <https://www.youtube.com/watch?v=7AIjcZMo-V4> [Accessed 2 Jan. 2025].

Git (n.d.). *Git - Reference*. [online] git-scm.com. Available at: <https://git-scm.com/docs>.

# Work Breakdown Matrix

| Student ID (Plymouth) | Student Name             | Contributed Tasks   |
|-----------------------|--------------------------|---|
| 10953243              | Bedde Samarasinghe       | UI / UX Design<br>Front-End Development<br>Report Writing<br>Ticket Management<br>Movie Management<br>Payment Handling    |
| 10952466              | Hewa Gamage              | UML Diagrams<br>Front-End Development<br>Report Writing<br>Ticket Management<br>Movie Management<br>Notification Handling |
| 10953326              | Yuhas Gamage             | Testing and Debugging<br>User Management  |
| 10952509              | Amarasinghe Senevirathne | Admin Login and Register<br>Movie Management  |
| 10903095              | Liyana Gunawardana       | Booking Management<br>Admin Dashboard   |
| 10899703              | Dasith Silva             | Admin Dashboard<br>Booking Management   |
| 10820754              | Herath Vimalaranga       | Feedback Management   |
| 10952758              | Hiddiyadura Abeynayake   | Feedback Management<br>Ticket Management<br>Admin Dashboard   |

## GitHub Repository Link

All the projects' files, database backup file and this report is available in below GitHub repository:

<https://github.com/RedsharkG/ABCCinema.git>

